Future internet metrology : characterization, quantification and prediction of web browsing quality

Antoine Saverimoutou

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Antoine SAVERIMOUTOU

Métrologie de l'Internet du futur : Caractérisation, Quantification et Prédiction de la Qualité de la Navigation Web

Thèse présentée et soutenue à Brest, le 15 mai 2020
Unité de recherche : Lab STICC
Thèse N° : 2020IMTA0186

Rapporteurs avant soutenance:

M. Raouf BOUTABA	PROFESSEUR	Université de Waterloo
Mme. Cristel PELSSER	PROFESSEURE	Université de Strasbourg

Composition du Jury :

Président:

M. Toufik AHMED	PROFESSEUR	Bordeaux INP

Examinateurs:

M. Raouf BOUTABA	PROFESSEUR	Université de Waterloo
M. Pedro CASAS	CHERCHEUR SENIOR	Austrian Institute of Technology GmbH
M. Olivier FOURMAUX	MAITRE DE CONFERENCES	Sorbonne Université
Mme. Cristel PELSSER	PROFESSEURE	Université de Strasbourg

Dir. de thèse:

Mme. Sandrine VATON	PROFESSEURE	IMT Atlantique Brest

Co-directeur de thèse:

M. Bertrand MATHIEU	CHERCHEUR SENIOR	Orange Labs

This thesis has been conducted in collaboration with
Orange Labs and IMT Atlantique

I dedicate this thesis
to my
family

Acknowledgements

Many individuals helped me into the completion of this thesis both technically and morally. I would like to mention a few of them. I am deeply grateful to Dr. Bertrand Mathieu, my thesis advisor at Orange Labs, who guided me throughout my work. I am thankful to him for believing in me and my research, for his patience, continuous hard work to keep my thesis work in the right direction and his kind and pedagogical guidance. Without his implication, this work would never have been possible.

I am thankful and grateful to my thesis director, Prof. Sandrine Vaton, who supported me and provided guidance during these three years. Her support and guidance has helped me into digging further my research work.

I am thankful to my thesis reporters, Prof. Cristel Pelsser and Prof. Raouf Boutaba for accepting to review my work and spare their precious time. I am also grateful to my thesis examiners, Dr. Toufik Ahmed, Dr. Pedro Casas and Dr. Olivier Fourmaux to spare their time and be part of my jury.

I would like to thank Orange Labs which allowed me to go beyond my limits and support my work. I am thankful to my head of research, Dr. Stéphane Tuffin, who provided precious advices where every meeting was friendly and beneficial. I am grateful to my team leader, Mr. Jean-Marc Corolleur, who provided continuous assistance and useful advices regarding my work and presentations. All members of the team were kind and always ready to provide assistance during these three years. I am grateful to all members of the team who provided constructive feedback during my pre-defense. Specially, I would like to thank Mr. Patrick Anat with whom I worked in collaboration. Working with him was fruitful and I learnt a lot from him. I would like to thank Mr. Maxime Laye for his kindness and useful feedback on my work.

I would like to thank IMT Atlantique which provided guidance and support during these three years. I would like to thank members of my research entity, LAB STICC, for providing me useful feedback during my presentations. Specially, I would like to thank Mrs. Armelle Lannuzel for her kindness and I am grateful for her administrative assistance.

I would like to express my gratitude to my colleagues and friends from Orange and IMT Atlantique for their support. Your constructive discussions have been very useful in the completion of this work. I would like to thank my colleague and friend, Dr. Romuald Corbel, for his support, advices and help. I am grateful to my colleague and friend, Miss Clara Catanese, for her support and jokes which have been very helpful in times of stress.

I would like to thank all members of my family, especially my mother who has allowed me to stay away from her for so long when pursuing my goals. I am grateful to all of you for your support, motivation and encouragement throughout this long period of time. Your positive attitude and recommendations helped me a lot.

Table of Contents

List of Figures

List of tables

Introduction

OVER the last decade, global Internet traffic has increased by more than 40%[1] and Internet users represent so far 57%[2] of the global population. Our society has been evolving to this digital trend thanks to the wide range of services e.g education, recreational, administrative purposes, etc., delivered to end-users through different communicating devices such as desktops, laptops, smart phones or tablets. These devices have gained over time increased computing power and battery lifetime coupled to higher bandwidths with the emergence of new network access technologies such as cable or dedicated Fiber. Consequently, the Web traffic has been increasing where end-users can browse more than 1.7 Billion different websites offering content of different flavors.

Accessing the Web is mainly performed through Web browsers, e.g Google Chrome, Firefox, Opera, etc., to display web pages of different websites and the process as a whole is commonly defined as Web browsing. The latter is a fast-paced and changing domain where a plethora of applications and services are nowadays accessed through them. The Web was originally designed to deliver static contents but has evolved towards dynamic web pages offering to end-users a built-in environment for education, gaming, video streaming or social networking. Web browsing has been driving the digital transformation of our society where people prefer to remain in touch globally and have access to their favorite services either at home, office, or even when travelling.

Web browsing has a complex eco-system since a wide set of actors are involved, from the end-user device, type of web browser, network access, the Internet protocol through which content is delivered, the different remote web servers delivering content and finally the websites themselves [1]. End-user devices evolve constantly, embarked with new functionalities and network operators have been offering network access with higher throughputs. Large service companies, e.g Google, Mozilla, Apple, etc., have been following the trend by enabling Virtual Reality, Virtual Machines or IoT (Internet of Things) applications embedded in their web browsers . Privacy is also promoted through private web browsing [2] or secured connections [3] [4].

Content needed to render web pages is downloaded through the HTTP (HyperText Transfer Protocol) protocol. The content was originally delivered over the HTTP/1.1 (Version 1.1 of the HyperText Transfer Protocol) protocol [5], but can nowadays be downloaded through new transport protocols such as HTTP/2 (Version 2 of the HyperText Transfer Protocol) [6], standardized in 2015, or QUIC (Quick UDP Internet Connections) [7], which is paving its way to standardization as HTTP/3. Service providers have also increased their fingerprint in the whole process by deploying different cache servers [8], CDNs (Content Delivery Networks) [9] and proxy-based architectures [10] in order to deliver content closest to end-users.

Websites are all different [11] and can be regrouped upon the type of content intended to end-users through categories. These different categories, e.g *News*, *Entertainment*, *Education*, etc., are put forth by different organizations such as Alexa[3], QuantCast[4] or Web

[1]http://hostingfacts.com/internet-facts-stats
[2] https://wearesocial.com/global-digital-report-2019
[3]https://www.alexa.com/topsites/category
[4]https://help.quantcast.com/hc/en-us/articles/115014006128-Interest-Category-Definitions

Filter[5]. Once regrouped by categories, these web pages are on average composed of the same types of objects, e.g scripts, images, videos, etc. [12].

The number of actors involved into the Web browsing eco-system is diverse and they have all evolved over the past years with the sole aim to enhance web browsing quality following different studies regarding end-users' web browsing patterns [13] [14] [15] [16] [17]. To meet this goal new Internet protocols embarked with new mechanisms, namely HTTP/2 [18] [19] or QUIC [20] [21] were introduced to deliver contents faster and promote end-users' privacy. New caching mechanisms [22] [23] and networking mechanisms [24] have also been enhanced to reduce the download times of content. New Web technologies [25] [26] promote delivery. New types of content, e.g WebP or WebM sponsored by *Google*, promoting quality and at the same time reducing the size of transferred bytes have also emerged. Although all these actors have evolved over time, Web browsing still suffers from performance issues [27] [28] since new Internet protocols are being slowly adopted, caching and networking mechanisms are not scaling to increased demand in real-time or new Web technologies not available through different web browsers.

Problem Statement

Web browsing perceived quality [29] [30] [31] is most of the time correlated to the time needed to load a web page (or parts of it). An increase in web page load timings can impact end-users' behavior which can lead to web page abandonment and thus a potential decrease in a website revenue (e.g for an *e-commerce* website, a delay of 1 second in web page loading can lead up to 2.5 Million dollars of revenue loss[6] per year). When the observed loading times are degraded, i.e a specific web page loading time is higher compared to past web browsing activity, there is a need to identify which web browsing factor contributes the most to this degradation.

Identifying the parameters contributing the most to web browsing quality is two-fold. Firstly it can help web developers to optimize web pages' loading times and strengthen customer experience which is the key to retention and loyalty or encourage service companies to adopt new Web technologies. Secondly, these parameters can be used as benchmarking indicators to identify on-the-fly degradations during the web pages' loading process. As an example, if the degradations are linked to the network (in particular internal network of an operator), network operators can anticipate and put forth different mechanisms, e.g modify traffic routing policies, deploy caches in their network, etc., to ensure optimal Quality of Service (QoS).

Regarding the study of web browsing, current approaches in the literature pay particular attention to the following questions:

- To what extent can end-users' devices, e.g computing power, impact web browsing perceived quality?
- Which web browsers with different embarked functionalities can improve web page loading times?
- To what extent new Internet protocols can enhance web browsing quality?
- How can new Internet protocols enhance web browsing experience when watching videos (mainly on mobile applications)?
- What is the impact of degraded web browsing quality on real end-users' behavior?

[5]https://fortiguard.com/webfilter/categories
[6]Neil Patel Digital Report on negative effects of web page loading times

All these research works have contributed in having an overview of the complexity of web browsing and underlying impact on perceived quality but do face certain limits:

1. Web browsers are upgraded on a regular basis to offer enhanced security features to promote privacy or have their processing engines improved. Furthermore, end-users may upgrade their device to gain in computational power. Actual research studies can be biased since most of the time only a limited type of web browsers with a time-frozen version, are considered. Perceived web pages' loading times should be measured following different web browsers and their different versions over long periods of time to better understand web browsing quality changes.
2. When browsing a specific website, the content is downloaded from different web servers and even when requesting the contents to be delivered via the latest Internet protocol (which are sometimes not implemented in all web browsers), some web servers might not implement it. More focus should be made on the demand and supply rate of Internet protocols (e.g when using a Firefox web browser, what is the distribution rate of content in HTTP/2 and HTTP/1.1) and hence the underlying impact on web browsing quality.
3. Web servers have different policies and web developers might favor the delivery of content to end-users via caches or CDNs whose main purpose is to deliver content closest to end-users. More emphasis on the type of web servers delivering contents and underlying policies should be taken into account to better understand perceived quality.
4. Web browsers, pages and technologies evolve on a regular basis, i.e on average every 20 days for web browsers, every 10 mins for web pages and every day for web technologies. The different web metrics, put forth by standardization bodies, meant to measure web pages' loading times are updated on average every 3 years and might not be implemented by all web browsers. Furthermore, these web metrics might not measure finely loading times with the introduction of new web technologies, e.g Progressive Web Applications (PWA). In order not to rely on the type or version of web browsers, new web metrics making use of web browsers' networking logs should be privileged.
5. In order to better study the impact of web browsing on end-users' behavior, but also its evolutions over time, measurements should be performed on long time periods since research works are often conducted over an average period of 3 months. The use of different web browsers and versions over long periods of time can help to better identify when and which web browser impacts the most web browsing quality.
6. Web servers are located all over the globe and might belong to different Autonomous Systems, and the network path (ascending and descending) to access them might change on an hourly or daily basis. The network path taken by packets to reach web servers (requests) and reach end-users (responses) must also be studied.
7. Different tools are used in the literature to measure web browsing quality but do not reproduce most of the time an end-user environment, i.e real desktops or laptops, real web browsers embarked with different ad blockers or probes connected to residential network access. New tools being end-user representative are needed to finely qualify and quantify web browsing quality.
8. Understanding the web browsing eco-system is important but degraded quality must be identified in real time (its precise cause among the different web browsing actors) so that web developers can enhance their web pages, service providers can deliver content from different web servers or network operators being proactive by re-routing their internal traffic.

In light of these limits, the web eco-system must be studied as a whole and not unit-wise, e.g the impact of Internet protocols, web browsers or service providers disjointedly. New measurement tools being end-user representative should be used to better quality QoE. New

web metrics making abstraction of web browsers' types and versions should be privileged when measuring web browsing quality by making full use of networking information. With the wide set of factors such as Internet protocols, service providers and companies, network state, etc., involved in the web browsing process, it is important to identify when and how each actor can enhance or decrease perceived quality. With more than 1.7 Billion websites which can be visited by end-users, predicting the underlying web browsing quality from the profiled content can help service providers to offer better QoS. Web browsing quality may also fluctuate at different times of the day for different websites and new solutions being able to cope with these fluctuations and provide efficient quality predictions are needed.

Thesis Contribution

In order to address the above-mentioned problems we tackled the study of web browsing quality as per its whole eco-system where we focus on homepages of websites in 4 ways. Firstly we have developed a new automated user-representative web browsing measurement tool where measurements of specific mostly visited websites are represented in real-time on a public monitoring website. Secondly we have defined a new web metric making abstraction of web browsers' types and versions in order to cope with web metrics which have not evolved accordingly to newly introduced Web technologies. Thirdly, from our tool measurements, we have identified through statistical techniques the parameters which can enhance or decrease web browsing quality. Last but not the least; we have applied machine learning techniques on measurements performed over large time spans to identify the sets of rules (parameters' thresholds) quantifying web browsing quality. These identified sets of rules have been applied on never assessed websites to predict the perceived QoE upon the rendering of the visible portion of a web page where the error prediction rate is mean, i.e 9.6%. When focusing on quality prediction for an entire web page, a larger number of factors are implicated and the construction of our rules-based model has been tuned for websites where fluctuations happen regularly. For some websites the prediction correctness for the entire web page perceived quality can be increased up to 25%.

A wide set of tools have been put forth by different research studies or service companies over the past years to measure web browsing quality of a fixed number of websites. While these tools instrument web browsers which may have time-frozen versions, different modules of these web browsers might purposely be de-activated which do not reflect end-users' environment. Furthermore, these tools offer limited functionalities and do not allow the thorough study of the Web browsing eco-system. We have thus designed, developed and deployed at different geographic locations a new tool, Web View. The latter makes use of real web browsers (and corresponding versions since more than 2.5 years) connected to residential access networks and measures a wide set of websites (Top 10,000 Alexa websites) 24/7/365. The main aim of our tool is to perform measurements in a user-representative manner and consider the Web browsing eco-system as a whole. Our tool is upgraded on a regular basis and we have conducted more than 18 Trillion distinct measurements over 2.5 years where each measurement offers 84 different quality indicators (loading times, identification of web servers delivering content, Internet protocol through which content is downloaded, upstream path taken by network packets, etc.). In order to keep track of web pages loading times through time, we have also introduced a public visualization website (https://webview.orange.com).

Web technologies and web browsers evolve on a regular basis and actual web metrics deliver loading times being different since large service companies integrate them in their web browsers differently. From our tool Web View, we have identified during the first 6 months of its deployment that available web metrics had several inefficiencies. In order to have

uniform and fine-grained loading times, we have defined a new web metric, the Time for Full Visual Rendering (TFVR) which makes use of web browsers' networking information to calculate the time to load the visible portion of a web page. Our web metric is independent of the type and version of web browsers and helps to identify which contents (types and sizes) are downloaded over different Internet protocols.

From collected web browsing measurements, we have used statistical techniques to identify the parameters which are involved in the Web browsing delivery, as well as their impact on Web browsing quality. We have been able to identify that new Internet protocols are being deployed at a low pace and thus contribute in the decrease of perceived QoE. Other parameters such as the usage of Content Delivery Networks can also leverage web browsing quality.

We have then focused our study on the parameters which can contribute to different web browsing quality ranges, commonly called *satisfaction degrees*, i.e *good*, *fair*, *moderate*, *worse* and *poor*. Our former statistical analysis has been confirmed by decision trees which have enriched these parameters with sets of rules, along with their thresholds, to qualify a particular satisfaction degree, e.g a *poor* perceived quality for a European end-user can represented by a website where more than 300 objects are downloaded over an ADSL (Asymmetric Digital Subscriber Line) network access through HTTP/1.1.

From the different identified rules for different satisfaction degrees, we have introduced a model which can predict web browsing quality for the visible portion of websites at first glance (without scrolling). The quality correctness prediction rate is 94.17% for the Top 10,000 Alexa websites measured on a regular basis. The accuracy of our model has been validated on the Top 10,000-15,000 Alexa websites never assessed before with a quality prediction error rate of 9.6%. Web browsing quality is usually qualified in literature upon the entire web page loading time where a larger number of factors are implicated (compared to the visible portion loading process). When the former identified rules-based model is applied for web pages having an important number of fluctuations in the entire web page loading times, the web browsing quality correctness prediction is low, e.g 74.92% for the web page *baidu.com*. To increase the accuracy of our model oriented towards the entire web page quality prediction, we have used a Hierarchical Dirichlet Process Hidden Markov Models (HDP-HMM) with Gaussian Mixture Model (GMM) emissions distributions which helps to finely identify these fluctuations and enrich our rules-based model. The application of this enriched rules-based model has helped in increasing entire web pages' quality prediction correctness up to 25%. This model thanks to the use of HDP-HMM with GMM emissions distributions has proven to be more versatile, i.e providing good quality predictions for the overall web page, over any website even if fluctuations in loading times are known to happen regularly.

When focusing on the quality prediction of the visible surface area of Web pages, our rules-based model can be applied on any website and underlying correctness in quality prediction is good. When focusing on the quality prediction for the entire Web page, our rules-based model can be applied on any web page if and only if fluctuations in loading times are mean. But if important fluctuations happen, a specific set of rules for every web page is needed.

Document Structure

The structure of the remainder of this document is organized as follows. In the first part in Chapter I, we present extensive literature reviews on the wide set of actors involved in the web browsing process, i.e their evolutions and promised outcomes. We discuss the

different types of application layer protocols, web servers' (origin web servers, caches and CDNs) policies and the different tools used to better understand web browsing quality. We then go through the different available web metrics put forth by standardization bodies and service companies to measure web pages' loading times.

In the second part, we go through our contributions. Chapter II presents our automated web browsing user-representative tool, Web View; its overall architecture together with the measurement parameters (inputs and outputs). We then discuss how we circumvent the limitations of actual web browsing tools through its evaluation and how our measurements are automatically represented on a public visualization website, which helps in assessing the evolution of the Web eco-system. Chapter III presents our web metric, the TFVR, its design and offered capabilities. Chapter IV presents the analysis of measurements performed by our tool upon specific conditions in order to identify which parameters are involved into the Web browsing delivery process. We also assess how these factors can increase or impact Web browsing quality. Chapter V presents how we constructed our rules-based model to predict perceived QoE regarding the visible portion of web pages. We also highlight how we used new segmentation solutions to take into account quality fluctuations to finely predict entire web pages' perceived quality. At the end, through Chapter VI, we give a general conclusion and future research directions.

Part I

Literature Review

Chapter 1

An Overview of Web Browsing Eco-system

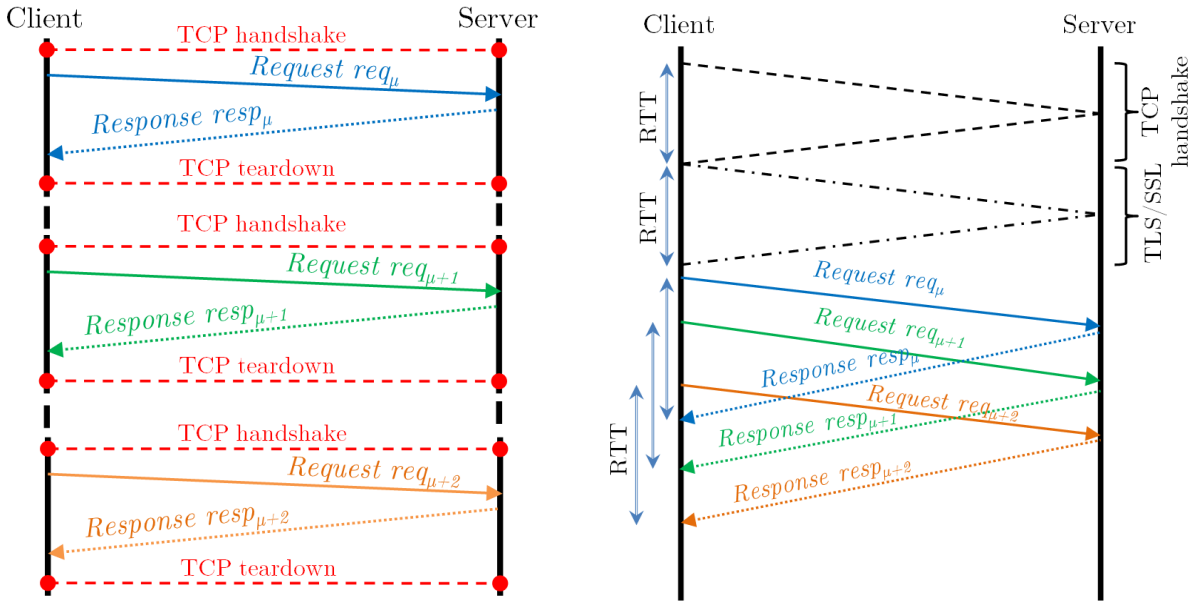
Contents

In this chapter, we review the different actors of the Web eco-system involved into the web browsing process. We go through an introduction of these different actors' specifications, together with their different evolutions in order to enhance web browsing quality, followed by research studies meant to quantify how they contribute to web browsing.

1.1 Application layer protocols

The Internet protocol is meant for addressing host interfaces; through datagrams encapsulate data and are routed from a source host interface to a destination source interface across different IP networks. Each datagram has two distinct components: a header (source IP address, destination IP address and metadata to route and deliver the datagram) and a payload which is the data to be transported. In order to provide connection-oriented links and datagram services between hosts, two modular architectures consisting of the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) at the Transport layer and the Internet protocol at the Internet layer.

To allow different end-users to access different web servers all over the globe, a stateless protocol, HTTP (Hypertext Transfer Protocol) was introduced in 1991. The protocol has gone through major evolutions over the last decade.



(a) HTTP/1.0 (b) HTTP/1.1

Fig. 1: HTTP/1.0 and HTTP/1.1 request-response mechanism

HTTP/1

Despite the popularity of HTTP, HTTP/1.0 [32] over TCP was introduced in 1996. This first version of HTTP suffered from several inefficiencies. When a client contacted a remote web server, the connection was automatically closed after a single request/response pair. The response following a specific request had to be explicitly received before sending another one as shown in Fig. 1(a); the request $req_{\mu+1}$ could not be sent before receiving the response $resp_{\mu}$ which leads to an accumulation of Round-Trip-Times (RTTs) and increased the end-to-end delay.

As a remedy, HTTP/1.1 [5] over TCP was standardized in 1999. The main goal of this new version of HTTP was to provide a keep-alive-mechanism where a connection could be re-used for more than one request. Twenty years after its introduction, HTTP/1.1 is still widely used where it allows a client to send a new request (two parallel connections following [5] in real-life scenarios but could also be up to six) before receiving the response of the previous one which reduces end-to-end delay considerably. However, a remote web server must respond to a request, req_{μ}, before processing the request $req_{\mu+1}$ as shown in Fig. 1(b). Furthermore, if the response of a request is blocked, e.g req_{μ}, all following requests are also blocked, i.e $req_{\mu+1}$ and $req_{\mu+2}$. Thus no response, i.e $resp_{\mu}$, $resp_{\mu+1}$ and $resp_{\mu+2}$, is made to the client, which is commonly known as head-of-line (HOL) blocking.

HTTP/1.1 as a whole generates more latency, i.e higher RTT values in web pages' loading times and creates head-of-line blocking, which prevents following requests to be received by the web server and reply accordingly to the client. Furthermore a HTTP/1.1-enabled web server only replies to user requests and is not able to push content.

HTTP/2

The version 2 of the HTTP protocol [6], usually denoted by HTTP/2 over TLS/TCP, was standardized in 2015. HTTP/2 was derived from the earlier experimental SPDY protocol promoted by *Google*. The main objective of HTTP/2 is to reduce web pages loading times by breaking down the HTTP protocol communication into an exchange of binary-encoded frames. These frames enable full request and response multiplexing, by allowing the client and server to break down an HTTP message into several independent frames, interleave them and then reassemble them on the other end. This particularity is commonly known as multiplexing multiple requests over a single TCP connection.

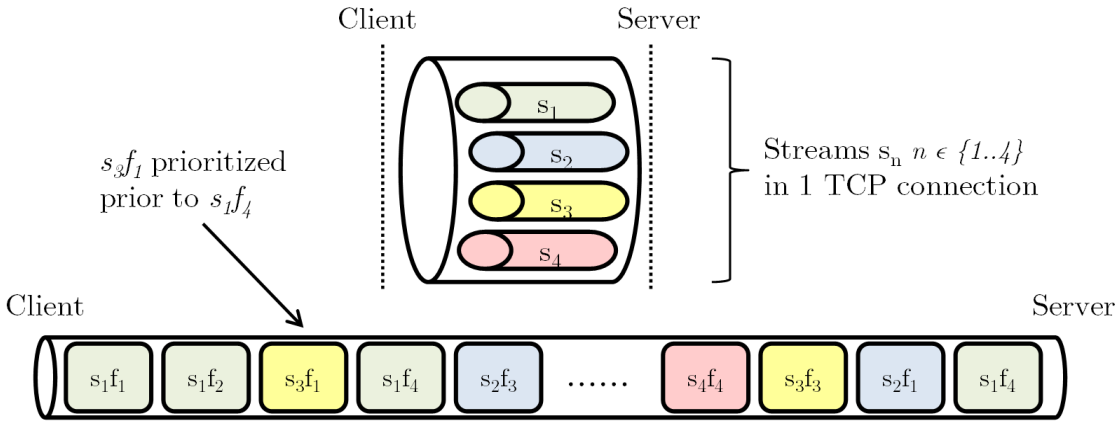


Fig. 2: HTTP/2 stream multiplexing and prioritizing

HTTP/2 also allows stream prioritizing by indicating the importance of a stream, s_n, compared to a stream s_{n+2} on the same connection. In addition to the traditional numerical priority of the *weight* of the stream, an HTTP/2 stream can also indicate the notion of *dependencies*. As illustrated in Fig. 2, the different frames, f_j with $j \in \{1..5\}$ following their stream s_n with $n \in \{1..4\}$ are rearranged following their dependencies and weight when the priority feature is activated. An additional particularity of HTTP/2 is the flow control mechanism, being directional (each receiver may choose to set its window size that it desires for each stream and entire connection), credit-based (each receiver advertises its initial connection and stream flow control window) and hop-by-hop (an intermediary can use it to control resource use). The flow control mechanism main goal is to prevent the sender from overwhelming the receiver with data it may not want or be able to process.

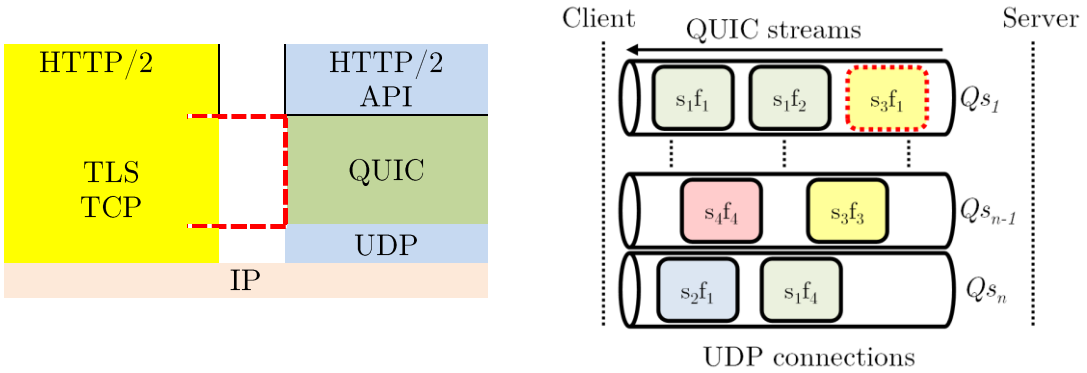
Another powerful feature of HTTP/2 is the ability of a web server to push additional resources to the client. Thus the client, i.e the web browser, does not have to request each one explicitly. In order to reduce overhead and improve performance, HTTP/2 compresses request and response header metadata.

Several studies [19] [33] [34] [35] [36] have been conducted in order to assess if the HTTP/2 protocol can enhance web browsing quality, thanks to its offered mechanisms. The studies clearly show that if the responses are delivered in HTTP/2 (not all web servers implement HTTP/2), in overall the download time of content is reduced, compared to HTTP/1.1. HTTP/2 as a whole allows a more efficient use of network resources and reduces perceived latency through header compression, multiplexing of requests over a single TCP connection and server push abilities. HTTP/2 delivering contents over TCP still contributes to higher RTT and creates head-of-line blocking.

QUIC (HTTP/3)

QUIC (Quick UDP Internet Connections) [7] protocol is a transport protocol being promoted by *Google* since 2016 and usually denoted by HTTP/2 over QUIC/UDP. From 2016 until now (2019), its corresponding version has moved from version 1 to 46 along with 23 different drafts. Since May 2019, the QUIC protocol is being discussed at the IETF (Internet Engineering Task Force) to be standardized as HTTP/3 (HTTP protocol over QUIC). The QUIC protocol is available in all chromium-based web browsers, e.g Google-Chrome, Brave or Microsoft Edge 2019. The version 46 is delivered into 2 flavors: GQUIC (Google QUIC) and IQUIC (IETF QUIC).

The QUIC protocol solves a quite number of transport-layer and application-layer problems where little or no change from web developers is needed. QUIC, being a self-contained protocol helps in reducing the RTT (*min.* 0-RTT, *max.* 1-RTT) thanks to UDP, compared to TCP (*min.* 1-RTT, *max.* 3-RTT). With QUIC, the 1-RTT is achieved when a client connects to a web server the first time and 0-RTT by using the cached credentials from



(a) HTTP/2 over TLS/TCP versus HTTP/2 over QUIC/UDP

(b) QUIC multiplexing

Fig. 3: HTTP/2 over QUIC/UDP

the previous connection. As depicted in Fig. 3(a), the QUIC protocol helps in pushing network functions to the application layer. This favors connection migration when a client changes its IP address or port (for TCP all connections become automatically invalid) and thus prevents any in-flight requests to a web server. As identified previously in Fig. 2, if the HTPP/2 frame s_1f_4 was lost, the next frames (and streams) were not transmitted to the client until the web server re-emitted the packet. The QUIC protocol palliates to HOL blocking effects, as shown in Fig. 3(b) by:

1. For a QUIC stream Qs_1 who loses an HTTP/2 frame s_3f_1, only this QUIC stream will be impacted, the other streams (Qs_2, ..., Qs_n) will be delivered to the client.
2. When packet s_3f_1 is lost, QUIC is embarked with a Forward Error Correction (FEC) which is meant to recover from lost packets without waiting for a re-transmission.

Several studies have been conducted to better understand how QUIC can enhance web browsing [37] [38] but also [20] [21] [39] [40] in order to assess if the QUIC protocol outperforms the HTTP/2 protocol. The advantages of the QUIC protocol are in general hard to assess since the protocol is in a development phase and mainly *Google* servers implement it. When content can be downloaded through QUIC, a small enhancement of the delivery time is noticeable in environments of low bandwidth, e.g ADSL networks, thanks to its packet recovery mechanism. The QUIC protocol was mainly designed to favor content delivery time but recent studies [41] go even further by showing that with the very slow adoption of the protocol, QUIC's promised quality enhancement is hard to quantify.

The QUIC protocol embarks its own transport security to connections, the QUIC-Crypto protocol until version 44 (The Transport Layer Security version 1.3 is being discussed at the IETF to be used as the transport security mechanism in the future).

QUIC-Crypto

The QUIC-Crypto[7] protocol is part of QUIC that provides transport security [42] to a connection [43], where two session keys are used. To meet the 1-RTT, the parties first agree on an initial session key during the *Initial Key Agreement* phase, which can be used to exchange data until the final exchange key is set, detailed as follows:

- *Initial Key Agreement*: Each party sets its initial key material, i_k, which is used for encryption and decryption,
- *Initial Data Exchange*: Client C and Server S exchange their initial data, encrypted and authenticated,
- *Key Agreement*: Consists of one message. S generates a new Diffie-Hellman (DH) [44] value and sends its public DH value to C. The latter verifies authenticity of the

[7]https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g/edit

server's new *DH* public value and both parties at this point derive the session key material, s_k,

- *Data Exchange*: Consists of two packets. *C* and *S* use the session key material, s_k, to encrypt and authenticate their remaining data.

Transport Layer Security

Transport layer Security (TLS) is a cryptographic protocol designed to provide communications' security. The main goal of the TLS protocol is to provide privacy and data integrity between different computer applications.

Transport Layer Security version 1.2

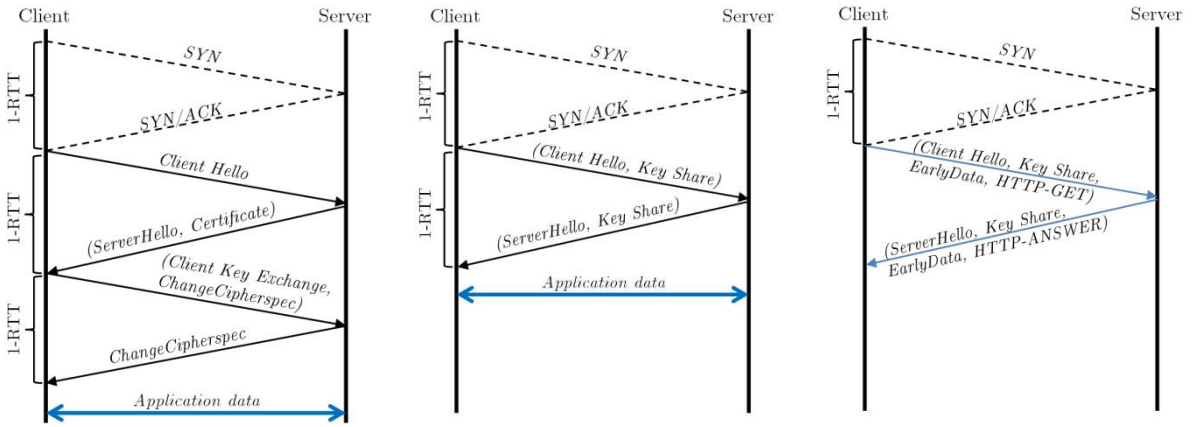
TLS version 1.2 [45] was introduced in 2008 which allows the client or server to specify which hashes and signature algorithms they accept. It provides cryptographic security, interoperability, extensibility and relative efficiency on two different levels:

1. TLS Record protocol: This protocol negotiates a private, reliable connection between the client and the server, where symmetric cryptography keys are used to ensure a private connection. The connection is secured through the use of hash functions generated by using a Message Authentication Code (MAC).
2. TLS negotiation protocol: The protocol allows authenticated communication to start between the client and the server. The handshake uses asymmetric encryption, where two separate keys are used; a public key used for encryption and a private key for decryption, to produce a newly-created shared key. The session then uses this freshly produced shared-key to perform a symmetric encryption, which yields to a feasible secure connection.

Transport Layer Security version 1.3

TLS version 1.3 [46] was standardized in 2018 and in contrast to TLS 1.2, it provides additional privacy for data exchange by encrypting more of the negotiation handshake [47] to protect it from eavesdroppers. The enhancement is two-fold: protect the identities of the participants (source and destination end points) and impede traffic analysis.

TLS 1.3 also enables forward secrecy by default, thus the compromise of long-term secrets used in the protocol does not allow the decryption of data communicated when those long term secrets were in use. This allows improving security of current communications even if the future ones are compromised. TLS 1.3 also reduces latency by reducing the corresponding RTT during its handshake exchange process. Although TLS 1.3 is not widely deployed on web servers [48], research works [49] prove that TLS 1.3 protects better the privacy of its end-users.



(a) TLS 1.2 (new session) (b) TLS 1.3 (new session) (c) TLS 1.3 (resumed session)

Fig. 4: TLS 1.2 versus TLS 1.3 over TCP

	HTTP protocol	New session	Resumed session
HTTP/2	HTTP/2 over TLS 1.2 / TCP	3-RTT	2-RTT
	HTTP/2 over TLS 1.3 / TCP	2-RTT	1-RTT
QUIC	HTTP/2 over QUIC / UDP	1-RTT	0-RTT
	HTTP/2 over TLS 1.3 / UDP	1-RTT	0-RTT

Table 1. Latency following different Internet protocols

The Fig. 4 shows the different handshake messages and corresponding RTT of TLS 1.2 versus TLS 1.3 over TCP. The usage of TCP always includes the SYN-SYN/ACK and a 1-RTT. The TLS 1.2, depicted by Fig. 4(a), adds 2-RTT for a new connection (additional 1-RTT for a resumed session), which sums up to a minimum of 2-RTT and maximum of 3-RTT. TLS 1.3 on the other hand reduces the overall RTT by 1. For a new connection, shown in Fig. 4(b), the latency sums up to 2-RTT and for a resumed session, through Fig. 4(c), we have a latency of 1-RTT.

As identified in Table 1, HTTP/2 over TCP contributes the most to the latency mainly due to the SYN-SYN/ACK, which UDP reduces. QUIC relies on TLS 1.3 for authentication and negotiation of parameters critical to security and performance. TLS Handshake and Alert messages are carried directly over the QUIC transport where instead of having a strict layering, these two protocols are co-dependent. QUIC makes use of the TLS handshake and TLS uses the reliability, ordered delivery and record layer provided by QUIC.

Protocols	Study	Literature	Takeaways	
			Pros	Cons
HTTP/1.0	Mechanism and efficiency	[32] [50] [51]	- Allows communication between a client and web server	- High RTT - Connection closed after single request / response - Impacts delivery time
HTTP/1.1	Mechanism and efficiency	[5] [51] [52]	- Send new request before receiving response of the previous one which considerably - Six parallel connections	- High RTT - Impacts delivery time - Unsecured
HTTP/2	Mechanism, adoption and efficiency	[6] [19] [52] [33] [35] [34] [35] [36] [53] [54] [20]	- Reduces content delivery time (versus HTTP/1) - Stream prioritization - Flow control mechanism - Header compression - Server push capabilities - End-to-end secured data exchange	- Not widely deployed - TCP contributes to higher RTT
QUIC (HTTP/3)	Mechanism, adoption and efficiency	[7] [37] [38] [20] [21] [39] [40] [55] [37] [41]	- Reduced latency thanks to UDP - Web QoE enhanced in low throughput environments	- Mainly deployed on *Google* web servers - Hard to assess gain in delivery time due to deployment rate
Application layer protocols security	TLS 1.2 / 1.3	[45] [46] [47] [48] [49]	- TLS 1.3 reduces overall RTT - End-to-end data integrity	- TLS 1.3 not widely deployed

Table 2. Application layer protocols overview

Takeaways: Application layer protocols have been evolving relentlessly over the past years where the main goal has been to deliver content faster to end-users. Table 2 presents the Internet protocols architectures as well as research works performed to verify how these new Internet protocols can enhance web browsing quality. While HTTP/1.1 has brought keep-alive mechanisms and thus help in reducing end-to-end delay, HTTP/2 allows multiplexing of multiple requests over single TCP connections, stream prioritizing or server push. Although standardized in 2015, the HTTP/2 protocol is not implemented by all web servers, but when available helps in reducing the time to download content compared to HTTP/1.1. The QUIC protocol is in its standardization phase towards HTTP/3 with the use of TLS 1.3 and its main goal is to deliver content over UDP and thus reduce latency. Since the protocol is still into a development phase, its outcomes are hard to assess as the protocol is mainly implemented on *Google* web servers. Through time, the QUIC protocol main goal has moved from "being able to deliver content faster" to "promote end-users' privacy". We have also identified that the HTTP/2 over QUIC/UDP provides greater security compared to HTTP/2 over TLS/TCP [56]. Although TLS 1.3 has been standardized in 2018, it has proven its robustness but its adoption rate is still very low.

1.2 Web servers

When visiting a website, a main HTML (HyperText Markup Language) file is downloaded and firstly processed by the web browser. Following the tag *<a href>*, additional resources such as images, style sheets, JavaScript, etc., are downloaded from different web servers. While the HTML file is always downloaded from the origin server, the other contents might be downloaded from *Non-Origin* web servers. A *Non-Origin* web server is denoted as a web server having its authoritative DNS (Domain Name System) different from the one of the origin web server (and conversely *Same-Origin* web servers).

When a European end-user visits the website *https://www.bbc.com*, a main web page is downloaded from the origin web server *bbc.com* which is hosted by *Amazon Cloudfront*. When the HTML is processed, additional content is downloaded from *Non-Origin* web servers such as *bid.g.doubleclick.net* (*Google*) or *t.effectivemeasure.net* (*Fastly*). These *Non-Origin* web servers are located all over the globe e.g *Amazon Cloudfront* in Germany or United States. Following the geographic location of an end-user, content might be delivered by web servers being closest to her. These web servers are commonly named caches or Content Delivery networks (CDNs).

Content Delivery Networks

A CDN is a globally distributed network of web servers whose main purpose is to provide content faster. The content is replicated and stored throughout the CDN so that an end-user can access the data being stored geographically close to him.

As depicted in Fig. 5, we have two origin web servers (O_1: *domain1.com* and O_2: *domain2.com*). The different CDNs (C_1, C_2, $C\text{-}I_3$, C_4 and C_5) will retrieve or receive data from the origin server, O_1, and end-users (U_1, U_2, U_3, U_4, U_5) will download the corresponding data from these CDNs.

CDNs are deployed by a wide number of service providers, e.g *Akamai*, *Fastly*, *Amazon Cloudfront*, etc., from all around the globe and have different architectures:

1. Push CDNs: The data is automatically pushed by the origin web server to the corresponding CDN. As illustrated in Fig. 5, the origin web server, O_1, pushes the content to the CDN, C_1 and the end-user, U_1, can retrieve the latest updated content.
2. Pull CDNs: Content stored into a CDN infrastructure has a corresponding Time-To-Live (TTL). Upon expiration, the content may either be automatically pulled from the origin web server or the CDN will wait for a first request from an end-user to

retrieve the content in order to save bandwidth. As shown in Fig. 5, CDNs {C_2, C_4, C_5} pull the content from the origin web server O_1.

3. CDN Infrastructures: CDNs may also be composed of different nodes (storage, pull or push) and a main control node. As shown in Fig. 5, $C\text{-}I_3$ is a CDN infrastructure.

Fig. 6 depicts the CDN infrastructure, $C\text{-}I_3$. When an end-user requests content, the entry point is the control node. The control node relays the request to the set of delivery nodes, $S_{D1\text{-}3}$ with a load balancing feature. One of the delivery nodes, for example S_{D2} verifies if the content is available in the storage nodes, $S_{B1\text{-}3}$. If a storage node has the content, for example S_{B3}, the content is handed to S_{D2}, relayed to the control node which delivers the content to the end-user. If the content is not available in storage nodes, the pull/push nodes, for example S_{A1} which is a pull CDN makes a request to the origin web server. The content received by S_{A1} is replicated on the different storage nodes, handed to the delivery nodes, control node and finally to the end-user's web browser.

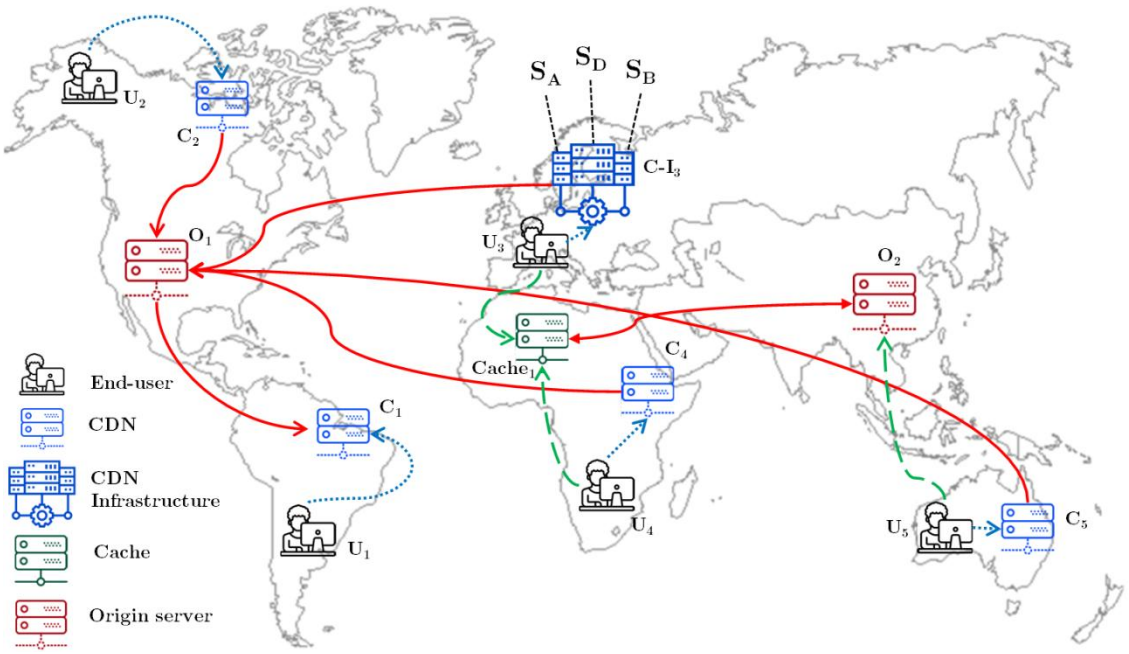


Fig. 5: Content Delivery Networks and Caches

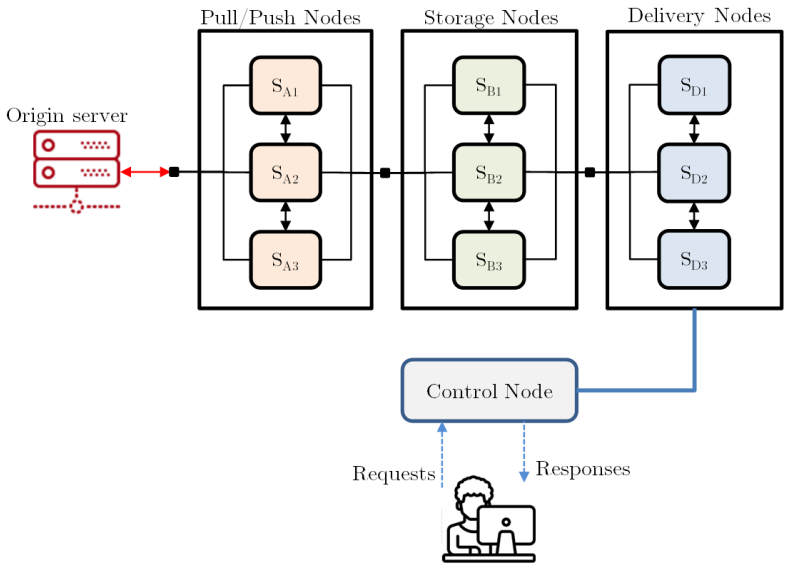


Fig. 6: CDN Infrastructure mechanism

Web Caches

Web caches follow the root functionalities of CDNs, where content is delivered close to end-users, reducing latency and optimizing content delivery. Web caches differ from CDNs by delivering most of the time static contents and their main aim is to optimize the data flow between the client and server, i.e a cache is a replica of the origin web server. Web caches are primarily used by Internet Service Providers, e.g *China Telecom*, backbone providers, e.g *China169-backbone*, large Intranets and enterprises, e.g *Microsoft-corp-msn-as-block*. Web caches are also used in different systems such as search-engines, web proxies and forward caches. Among the mostly used web cache software are *Nginx*, *Varnish Cache*, *Squid*, *Polipo*, etc.

As shown in Fig. 5, the web cache, $Cache_1$, located in Africa will store the contents of the origin server, O_2, which is located in Asia. The end-users are mainly redirected to web caches thanks to their DNS records. While the end-users U_3 and U_4 will retrieve content from $Cache_1$, the end-user, U_5, when requesting an object from *domain2.com*, will be automatically redirected to the origin web server, O_2, instead of $Cache_1$, due to its geographic position.

Takeaways: When delivering large scale websites to global audience, CDNs can reduce latency, accelerate websites loadings and reduce bandwidth consumption [9] [23] [57] [58] [59] [60]. In other research works [61] [62], authors assess the potential cache performance for CDNs and traditional web delivery. CDNs have been largely used in the past to deliver static contents but with the constant evolution of the Web to deliver dynamic contents, a wide range of content is nowadays delivered by CDNs [63]. Web caches [64] on the other hand deliver most of the time static contents and are primarily used by Internet Service Providers, backbone providers, large Intranets and enterprises.

1.3 Web Pages

Access to the Web is mainly done through web browsers where a main web page is firstly downloaded from the origin web server. Depending on the content needed to fully render the web page, additional content from other web servers is downloaded. During this process, this main web page is downloaded in bytes, converted to characters, tokens, nodes and finally object models. These object models are commonly named Document Object Model (DOM) and Cascading Style Sheet Object Model (CSSOM) which are both independent data structures.

Document Object Model

When accessing a website, a plain HTML file is firstly downloaded. These raw bytes, Fig. 7 – Step 1 of HTML are translated following specific encodings to individual characters, Fig. 7 – Step 2. Following the different strings of characters, they are converted into distinct tokens as per the W3C HTML5 standard[8], identified in Fig. 7 – Step 3. These tokens are then converted into nodes (*objects*) as shown in Fig. 7 – Step 4, which define their distinct set of properties and rules. Finally, the DOM is constructed, Fig. 7 – Step 5. Since the HTML markup defines relationships between different tags, the different created objects are linked in a tree data structure which helps to identify precisely the parent-child relationships. Web pages may sometimes be incorrectly written and are defined as *not valid HTML document*. The web browser will in this case automatically correct any missing element e.g missing the *<head>* and *<body>* element. When JavaScript is used, additional content might be downloaded and the DOM tree will be rebuilt accordingly.

[8]https://html.spec.whatwg.org/multipage/

CSS Object Model

The CSSOM on the other hand is meant to build a corresponding tree but focused on the different tags' styles e.g the font type and color. The CSSOM is built whenever a CSS style sheet is met in the HTML file. Building the CSSOM follows the same principle as the DOM, i.e convert the raw bytes to characters, tokens, nodes and finally the CSSOM depicted in Fig. 8. When JavaScript is used to download additional CSS style sheets, the CSSOM automatically rebuilds itself.

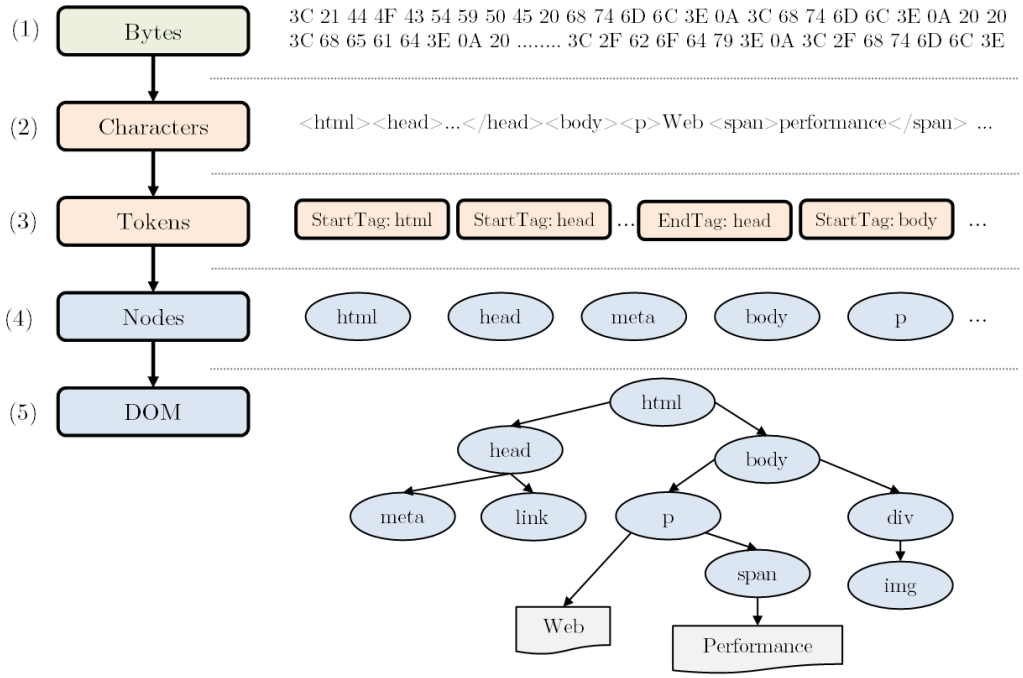


Fig. 7: Building the DOM of a web page

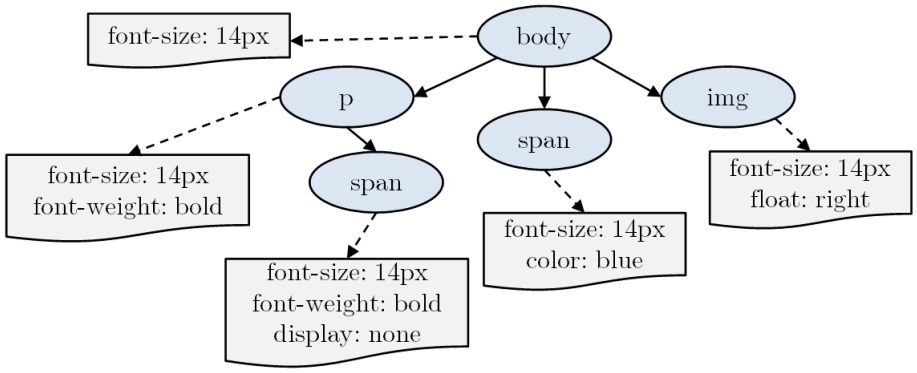


Fig. 8: Building the CSSOM of a web page

Takeaways: The DOM and CSSOM are both built separately; the web browser will combine both trees and render the output in the web browser window. Web browsers are developed by different service companies and have different policies and processing algorithms. As an example, the number of events triggered when rendering a web page during web browsing sessions can range from 10 to 1000 even if only 7 objects are downloaded and rendered in the web browser, e.g *wikipedia.org.* DOM and CSSOM processing is mandatory

in order to graphically render web pages. While JavaScript is greatly used by web developers, the DOM tree can be rebuilt several times until providing the final visual representation of a web page.

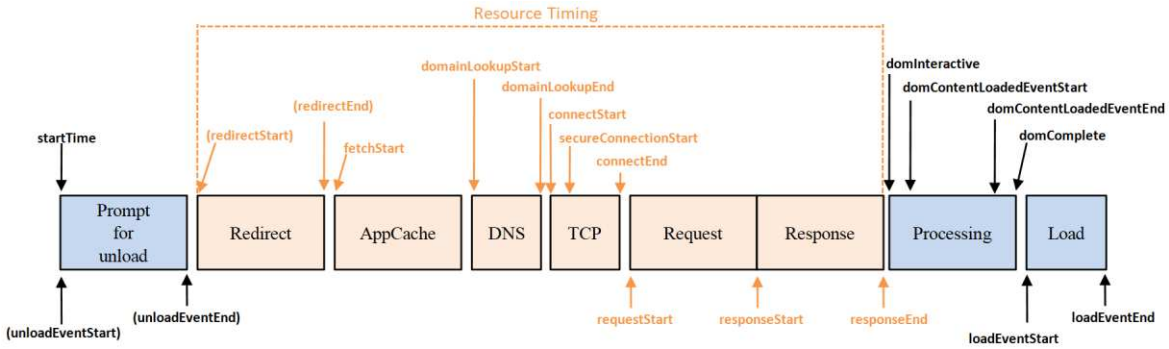


Fig. 9: Navigation Timing API

1.4 Web performance metrics

In order to bring uniform benchmarking indicators, standardization bodies such as the W3C (World Wide Web Consortium), in collaboration with a wide set of large service companies have defined a set of web metrics to measure web pages' loading time.

Navigation Timing API

The Navigation Timing API[9] [65] exposes the Page Load Time (PLT) which is the time between the start of a web page request and the moment it is entirely loaded. The Navigation Timing API is put forth by the W3C and offers different timings which outline the loading process of a web page. As identified by Fig. 9, the PLT-W3C is calculated by summing up different loading phases for every downloaded object (regrouped by connections). The Table 3 illustrates the different timings offered by the Navigation Timing.

Attribute	Definition
Prompt for unload	Discharge any timing calculations and get ready for a web navigation start
Redirect	The time to redirect end-user's request following his geographic position e.g An end-user in France will be redirected to *https://www.google.fr* when requesting *https://www.google.com*
AppCache	The time to retrieve contents from the end-user's web browser cache (if any)
DNS	The time to obtain DNS records
TCP	The time needed to negotiate for the first time with remote web servers (TLS negotiation or security keys exchange)
Request	The time to send requests to remote web servers
Response	The time needed to download content from web servers
Processing	The time needed by the web browser to build the DOM and CSSOM
onLoad	The time to load the DOM and CSSOM visually in the web browser window

Table 3: Navigation Timing API quality indicators

[9]https://www.w3.org/TR/navigation-timing-2/

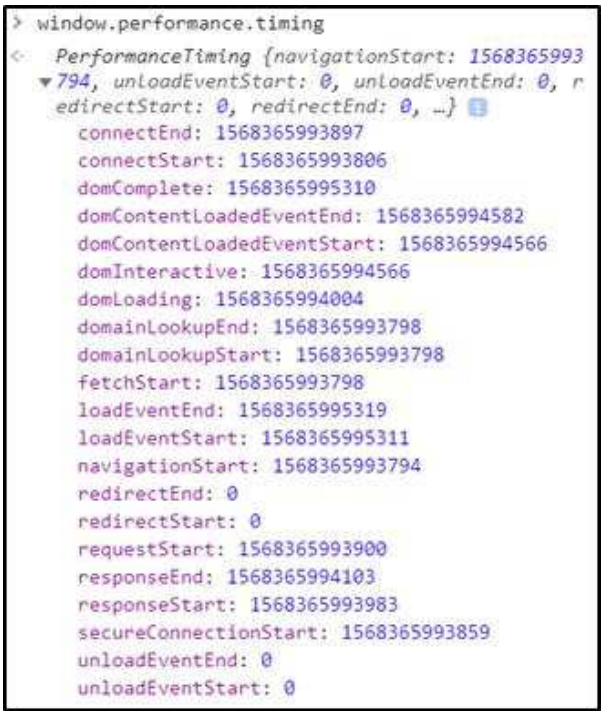


Fig. 10: Performance Timing

The PLT is implemented by default by the different on-market web browsers. Its corresponding value and is exposed through the console via the `window.performance.timing` function as shown in Fig. 10. Following the Fig. 9, the different loading times can be calculated. When browsing the homepage of *https://www.google.com* the exposed PLT-W3C is 1525 ms (`loadEventEnd - navigationStart`) where an overall number of 19 objects are requested in 83 ms and downloaded in 120ms (regrouped by `{IP_address, port}`). The web browser's processing time is 744 ms (`domComplete - domInteractive`), which contributes to 49% of the PLT-W3C value. The Navigation Timing through its design provides information on overall timings needed in order to request or download content i.e 19 different objects are processed by the web browser in 744 ms. The PLT-W3C obtained through the Navigation Timing API although being in a *"draft version"* since several years is nowadays the de-facto web metric used to measure web pages' loading times.

Fig. 11: Resource Timing API indicators

Resource Timing API

The Resource Timing[10] [66] is meant to provide information upon the downloaded resources unit-wise by exposing the different loading times such as the transport protocol used to request and receive a particular object, the size or type of downloaded object and some low level networking information.

The Resource Timing values are exposed in the web browser's console via the function `window.performance.getEntriesByType('resource')` as shown in Fig. 11 when web browsing the website *https://www.google.com*. For every downloaded object needed for the web page to be entirely rendered to the end-user, timings are offered. The 8th downloaded object (indicated by `8: PerformanceResourceTiming`) is an image of size 4396 bytes and is downloaded in 1.61 ms in HTTP/2. The total time to obtain this object is 23.9 ms (connection establishment, request and response). The Resource Timing API does provide additional information on every downloaded object but is subject to how the distant web server is configured. If a web server has its `Time-Allow-Origin`[11] set to False mainly due to security policies, only the type, size of the object and the Internet protocol through which it is delivered will be available. We will discuss in detail this inefficiency in section 3.1.1.

Paint Timing API

The Paint Timing API[12] [67] exposes four loading times prior to the web navigation start, namely the First Paint (FP), First Contentful Paint (FCP), First Meaningful Paint (FMP) and Time To Interactive (TTI), all meant to measure web pages' loading progression through time.

These four loading times indicators are meant to:

- The FP exposes the time when a first pixel is rendered in the web browser's window by excluding the default background paint.
- The FCP exposes the time when the web browser first renders any text, image, or non-white canvas which is the time when an end-user can start consuming web content.
- The FMP exposes the time after which the visible portion of the web browser window (without scrolling) has rendered all text and image backgrounds.
- The TTI exposes the time when the visible portion of the web page without scrolling is fully loaded (not taking into account advertisements) and when the end-user can start interacting with the web page.

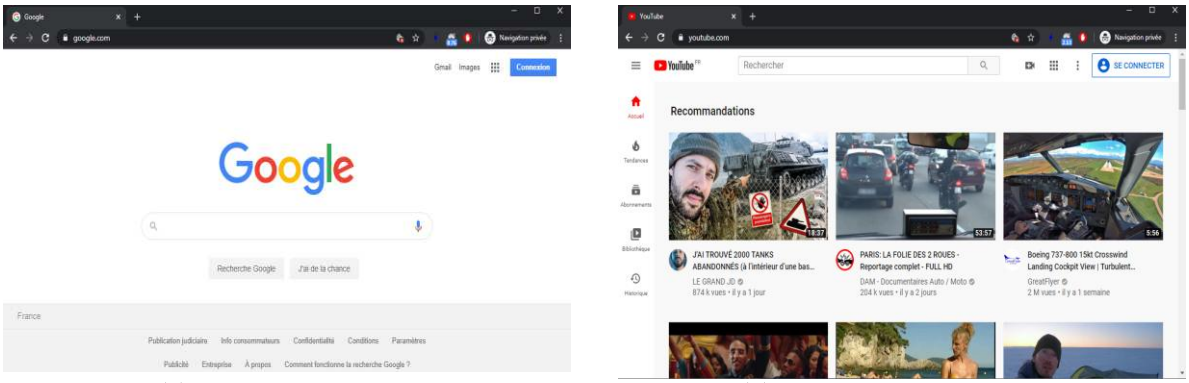
SpeedIndex

The SpeedIndex [68] provides a score to measure the visual progression of a web page rendering process. As an example, the SpeedIndex (SI) of the *Google* homepage depicted in Fig. 12(a), *https://www.google.com*, is 560 which reflect the visible portion of the homepage with minimal content. On the other hand, the SI of the homepage of *Youtube* depicted in Fig 12(b), *https://www.youtube.com*, is 5600 which indicates a visible portion of the homepage with a larger amount of content. Its calculation by default is performed by firstly making a video of the loading process. The video is then subdivided into several chunks, $\{C_{k1}, ...,C_{ki}\}$ and every chunk is compared to the previous one (C_{ki} versus C_{ki-1}) where the appearance of new pixels is assessed. The chunk C_{ki} defines the moment where no new change of pixels will happen. Nevertheless this technique bears two main inefficiencies, namely the use of video recording via external tools which can increase the device processing power and impact

[10]https://www.w3.org/TR/resource-timing-2/

[11] The Time-Allow-Origin response header specifies origins that are allowed to see values of attributes retrieved via features of the Resource Timing API, which would otherwise be reported as zero due to cross-origin restrictions.

[12] https://www.w3.org/TR/paint-timing/



(a) Homepage *www.google.com* (b) Homepage *www.youtube.com*

Fig. 12: Visible portion of websites

Web metric	Research studies
PLT	[69] [70] [71] [72] [73] [27] [28] [11] [12] [74] [75] [54] [76] [52] [77] [78] [79] [80] [81] [82] [55] [20] [21] [35] [36] [83] [84] [85]
Resource Timing	[69] [70] [71] [27] [76] [83] [84] [85]
Paint Timing	[69] [27] [76] [81] [83] [84] [85] [75]
SpeedIndex	[69] [76] [81] [83] [85] [75]
RUM SpeedIndex	[69] [83] [84]
ATF	[27] [81] [83] [84] [86] [85]

Table 4. Web metrics' usage by the research community

loading times and with dynamic advertisements, defining the last chunk C_{ki} is not an easy task. These two inefficiencies can lead to wrong SpeedIndex values. We will later dive into the detail of these inefficiencies in section 3.1.

In order to make abstraction of the video recording phase, the RUM-SI (Real User Monitoring SpeedIndex) has been introduced in 2012 where DOM information exposed by web browsers is used to identify these different images chunks.

Above-The-Fold

The Above-The-Fold (ATF) [86] is meant to measure the time to fully load the visible portion of a web page without scrolling. The ATF calculation can make use of video recording to identify the chunks C_{ki} but can also be calculated via the RUM-SI (only the time needed to render the visible portion of the web page is exposed). Current implementations of the ATF make use of web metrics such as the FP and the PLT-W3C which can sometimes be biased due to advertisements and Progressive Web Applications (PWA) which offer a mobile environment when using a desktop version of any web browser.

Takeaways: With the increasing usage of the Web [87] [88] to access a wide range of services, measuring web pages loading times (entire or certain parts of it) is a means to qualify web browsing quality. To measure these loading times, research works have been using different web metrics exposed via different APIs, namely the PLT, Resource Timing, Paint Timing, SpeedIndex, RUM SpeedIndex and ATF as shown through Table 4. Among the different available web metrics, the de-facto used web metric to measure web browsing is the PLT-W3C. Based on Table 4, only 2 research studies take into account all available web metrics which is mainly due to the appearance of new web metrics over time. The PLT as

defined by the W3C and how different service companies, e.g *Google*, *Mozilla*, *Opera*, etc. implement it in their web browsers do not take into consideration content delivered after the *onLoad* event as shown in Fig. 9. Additional events may happen, being triggered by JavaScript, which available web metrics do not take into account. Over the last years, with the introduction of new web technologies, the objectiveness of the PLT is questioned [27] showing that the HAR (Http ARchive) networking logs offered by web browsers help in better detecting downloaded content. Calculating the PLT from the HAR, which we denote by PLT-HAR in this document, is more precise than the exposed PLT-W3C. Upon different research studies focused on web browsing patterns of end-users [89] [14] [90] [15] [91] [92] [93] or end-users' behaviors [94] [16] [95] [96] [97] during web browsing sessions, the visible portion of the web page at first glance without scrolling the web page should be measured [75].

1.5 Web browsing measurement tools

In order to measure web browsing, in particular the different actors implicated in the whole process, automated tools are needed due to the high number of websites offering different types of content.

Since the introduction of the Web in 1996, in particular the HTTP protocol, a wide set of tools have been developed as commercial means or for research studies. The Web offers a wide set of services and the different tools introduced pay interest to fingerprinting (assess configuration parameters employed by web services or types of content offered by web pages) [63] [98] [70] [99] [100] [101], web browsing quality [54] [75] [102] [103] [104] [105], web privacy [99] [106] [100] [107], web browsing systems [108] or web browsing behaviors [75] [80] [15]. The Table 5 provides an overview of these different tools.

Web browsing tools	**Tool**	**Plugin**		**Used Web browser**				**Finger-printing**	**Web system**	**Web quality**	**Web privacy**	**Web behavior**
		Chrome	Mozilla	Chrome	Mozilla	Browser Engine	IE					
AdFisher [107]	✓				✓						✓	
Chameleon Crawler [101]	✓			✓				✓				
CoLab [108]	✓								✓			
FourthParty [99]			✓					✓			✓	
FPDetective [98]	✓			✓				✓			✓	
Gaze [75]	✓			✓						✓		✓
InspectorGadget [63]			✓		✓			✓				
OpenWPM [70]	✓			✓	✓			✓				
PageSpeed Insights [102]	✓			✓	✓					✓		
SiteSpeed.io [104]	✓			✓	✓		✓			✓		
WebPageTest [103]	✓			✓			✓			✓		
WebXRay [106]	✓					✓					✓	
WProf [54]	✓			✓						✓		
XRay [100]			✓					✓			✓	
YSlow [105]		✓	✓							✓		

Table 5: Available web browsing measurement tools

When paying particular attention to tools meant to assess web browsing quality, only 2 distinct tools (*WebPageTest* and *SiteSpeed.io*) are used in order to assess the loading times and corresponding content downloaded since:

- *YSlow*, *WProf* and *PageSpeed Insights* provide information on contents which should be delivered with reduced priority, in order to enhance the overall web page loading time.
- *Gaze* provides information on portions of websites where end-users focus, and hence these portions of the corresponding web page should be primarily downloaded

The tools *WebPageTest* and *SiteSpeed.io* provide loading times of web pages by making use of all web metrics identified in Table 4, and make use of real web browsers to perform automated web browsing sessions. *WebPageTest* also allows performing measurements from different end-points, e.g Asia, Europe or America. Nevertheless a specific network operator offering a corresponding network bandwidth cannot be selected and the available web browsers versions to perform measurements are not regularly updated.

Takeaways: Different web browsing tools have been introduced in order to better qualify web browsing. When focusing on web browsing quality, a limited amount of tools exist which are upgraded on a regular basis, making use of real updated web browsers embarked with commonly used ad blockers. Furthermore, with the constant evolution of the Web, so far in the literature, there is no tool to the best of our knowledge being representative of an end-user's environment, i.e real web browsers, measurements performed over residential access networks or identify the different types of web servers delivering content.

1.6 Networking path of web traffic

When performing web browsing sessions, a wide range of objects are downloaded from different domains and hence different web servers. An end-user might visit a website, e.g *https://www.bbc.com*, from a specific geographic location, e.g France in Europe and although the website *bbc.com* puts forth English news, content will be downloaded from different web servers from all around the globe e.g London, Germany, France and from the United States.

When a web browser parses a web page, requests will be sent to different web servers and these network packets will be following different paths depending on the end-user's geographic location and Internet Service Provider (ISP). The responses (content) might also follow different network paths to reach an end-user's web browser. When network packets are sent to web servers (or received), following a route taken at a specific time of the day, its transfer rate might be decreased and thus increase the overall web page loading time.

Measuring networking path between routers

In order to measure the networking path taken by network packets, different tools have been introduced over the past years.

Traceroute

Traceroute [109] was introduced in 1993 to learn the path between two machines (e.g an end-user device and a web server) which allows performing diagnosis of network problems. Unfortunately through time, with the introduction of load balancers, *traceroute* measurements can be inaccurate and incomplete when the measured route goes through a load balancer. While per-flow balancers ascribe each packet to a flow defined by the header five-tuple (and each flow to an outgoing interface), per-packet load balancers assign packets regardless of the flow. Since *traceroute* varies some of the packet header fields that are used to define a flow, an incorrect path can be diagnosed.

Paris-Traceroute

Paris-Traceroute [110] was introduced in 2006 to counter load balancing routers by controlling the probe packet header fields and allow all probes to reach their destination in the presence or per-flow load balancing. But following the nature of per-packet flow balancers, this first version of Paris-Traceroute could not enumerate all possible paths. A stochastic probing algorithm (Multipath Detection Algorithm - MDA) was proposed in 2007 which gave birth to a new version of Paris-Traceroute [111], adapting the number of probes to send on a hop by hop basis, and thus enumerate all interfaces and links at each hop. In 2018 a new version of Paris-Traceroute [112] has been introduced with a Multilevel MDA-Lite to integrate router-level view of multipath routes.

Identifying Autonomous Systems along the network path

Autonomous systems (AS), were introduced in order to regulate organizations such as ISPs, educational institutions and government bodies. An AS has many different sub-networks with combined logic and common routing policies and every sub-network is assigned a globally unique 16 digit identification number by the Internet Assigned Numbers Authority (IANA). Network packets among these different autonomous systems are routed via the Border Gateway Protocol (BGP) [113].

In order to obtain an AS number from an IP address, different tools exist to obtain information from different Regional Internet Registry (e.g AfriNIC[13], ARIN[14], APNIC[15], LACNIC[16] and RIPE NCC[17]).

- The Internet service *Whois* will look up online data across multiple RIRs (from an IP address or domain name) and provide the name of the registrar, creation date but also technical contacts of the registrant (a record may also be *private*).
- The *RIPE Stat*[18] tool was introduced in 2010 and allows obtaining a wide set of information e.g AS number, AS holder, geographic location from the RIPE NCC online database.
- The *PyASN* tool provides on the other hand offline and historical lookups where a fresh database can be regularly retrieved from *Route Views*[19]. The University of Oregon Route Views Project was originally conceived for Internet operators to obtain real-time BGP information about the global routing system from the perspectives of several different backbones and locations from the Internet. The main advantage of the *PyASN* tool is to perform offline requests to a local database and hence reduce network traffic.

Takeaways: During web browsing sessions, several web servers deliver a wide range of content which are then rendered by web browsers. These web servers are located all around the globe and the corresponding networks packets might follow different paths and suffer from network degradations, e.g re-routing of traffic by load balancers, network state at different locations, etc. It is thus important to profile the upstream and downstream path taken by networks packets and corresponding RTT. When transfer rates are higher than past measurements, we can thus estimate in which AS we have network degradations. Several studies [114] [115] [116] [117] [118] [119] [120] have been conducted over the past years and have been able to identify where and when network degradations occur, e.g BGP routes between AS change and corresponding RTT increase, specific routers in different AS create congestion, etc.

[13] African Network Information Centre
[14] American Registry for Internet Numbers
[15] Asia-Pacific Network Information Centre
[16] Latin America and Caribbean Network Information Centre
[17] RIPE Network Coordination Centre
[18] https://stat.ripe.net/
[19] http://www.routeviews.org/routeviews/

1.7 Conclusion

The HTTP Internet protocol has been evolving relentlessly the past years in order to cope with the increased Quality of Service (QoS) demanded by end-users. QUIC which is undergoing a standardization process at the IETF and will be next-called HTTP/3 was mainly supported by two web browsers, namely Google-Chrome and Brave but as from October 2019, Firefox has started supporting it. In order to promote privacy, the different Internet protocols are coupled with new cryptographic protocols in order to secure all communications. The TLS 1.2 was mainly coupled to HTTP/2 and QUIC had its own cryptographic protocol, the QUIC-CRYPTO. With the introduction of TLS 1.3, both HTTP/2 and QUIC (HTTP/3) will be using it. Through the design of TLS 1.3 the corresponding RTTs are reduced during the security negotiations.

When browsing websites, content is downloaded from different web servers from all over the globe. In order to enhance the QoS, Content Deliver Networks have been introduced in the whole process, which are intermediate web servers located close to different end-users. CDNs have become a key factor involved in content delivery over the last decade. Web pages are all different among them but go through two main processes once needed content is downloaded: building the DOM and CSSOM. These two phases are compulsory for a web page to be rendered in a corresponding web browser. Web browsers being embarked with different processing engines, these two phases are done following different policies which can impact the web pages' loading process.

In order to measure the web pages loading process through time, a wide set of web metrics have been put forth by the W3C or industry. These web metrics, for some being draft versions since several years due to constant web browsing technologies' evolution, help to provide benchmarking parameters to qualify web browsing QoE. Through time, advertisements are strongly embedded in web pages and the need to measure the time to load the visible portion of a web page without scrolling is more and more needed. Nevertheless the web metrics put forth to measure primarily the visible portion of web pages make use of other web metrics evolving at a slow pace and do not provide fine-grained loading times.

During web browsing sessions, several web servers deliver a wide range of content and the corresponding networks packets might follow different routes and suffer from network degradations. It is thus important to profile the upstream and downstream path taken by networks packets and corresponding RTT. When transfer rates are higher than past measurements, we can thus estimate in which AS we have network degradations to better justify web browsing quality degradation.

The Web eco-system is complex where a large number of actors are implicated and to better understand web browsing quality, all these actors' implication should be assessed.

Part II

Contributions

Chapter 2

A New Web Browsing Measuring Tool: Web View

Contents

In order to better understand the Web browsing eco-system, a wide set of automated tools exist. As discussed in section 1.5, these tools are either meant to assess the browser fingerprinting which fully or partially identifies end-users' activity, Internet privacy, end-users' behavior or quality. Among all these tools, a limited amount focuses on Web browsing quality. These web browsing tools have proven to be useful to qualify web browsing quality but the measurements are not performed upon an end-users' representative environment. Furthermore new Web technologies have been introduced and their offered functionalities do not take into account these technologies. We have thus designed, developed and deployed a new automated tool, Web View, with probes connected to residential access network and use a wide range of real updated web browsers. The measurements performed by our probes are automatically represented on a public visualization website[20].

2.1 Available web browsing quality tools and efficiency

When assessing web browsing quality, measurements must be performed into the most end-user representative manner. The tool *PhantomJS* makes use of different web browsers' engines where web page loading times can be measured. Although the tool is easy to install on desktop devices and uses minimal computational power, the real graphical version of web browsers is not used and several real-life functionalities are disabled. Furthermore, this tool allows retrieving only the PLT-W3C (Page Load Time defined by the

[20] https://webview.orange.com

World Wide Web Consortium) and as we will discuss in section 3.1.1, the way the PLT-W3C is implemented in web browsers can lead to inaccurate loading times. New Internet protocols such as HTTP/2 or QUIC have been introduced since 2015 and *PhantomJS* performs measurements by requesting strictly the HTTP/1.1 Internet protocol. Web servers are all designed to respond in HTTP/1.1 and when performing measurements with *PhantomJS*, all requests and responses are made into HTTP/1.1 which is not representative of the Web browsing eco-system. Last but not the least, the tool *PhantomJS* has stopped being maintained since May 2018.

WebPageTest offers a more realistic end-user environment when performing measurements. The tool is accessed from a public website[21] where the measurement parameters can be set, i.e we can choose a web browser type along with the website to be measured and the geographic location of the probe. The advantage compared to the *PhantomJS* is the use of real web browsers but the corresponding web browsers' versions are not exposed although being updated every 20 days on average. The offered web metrics values are W3C metrics, such as the PLT-W3C or the browser-based ATF (Above-The-Fold). If the *Google-Chrome* web browser is selected, the First Paint value can also be collected. These web metrics do provide indications on a web page loading time but can provide incorrect loading times (see section 3.1). The First Paint Timing API can only be triggered when using the *Chrome* web browser since it is not implemented[22] into the *Firefox* web browser. *WebPageTest* bears another advantage where custom JavaScript can be loaded into the web browser when performing measurements. When performing web browsing measurements from *WebPageTest*, there is no control on the network access, i.e we cannot choose among ADSL, Wi-Fi or Fiber and since the devices running the measurements are Virtual Machines (VMs), the computing environment is not end-user representative.

SiteSpeed.io has been introduced in the last quarter of 2018 to offer web page loading times calculated from HAR (Http ARchive) information or from W3C metrics. The tool is downloaded and installed on either desktop devices or virtual machines, where different web browsers (types and limited versions) can be used to perform web browsing measurements. Apart from the limitations of offered web browsers' versions, *SiteSpeed.io* is the most end-user representative tool available, if used on desktop machines and connected to representative end-user network access. This tool offers some basic loading times, e.g PLT-W3C, PLT-HAR, browser-based ATF. The HAR raw file can also be collected and re-processed to retrieve additional information such as the Internet protocol distribution or MIME (Multipurpose Internet Mail Extension) types of downloaded objects. This has to be performed through other tools.

When performing web browsing measurements, tools must be the most end-user representative through the use of real web browsers, residential network access and must correlate the obtained loading times obtained from HAR files to the different factors contributing to the Web eco-system, i.e Internet protocols, types and location of web servers delivering content or upstream and downstream path of network packets. The obtained information from the measurements must also be graphically represented in real time to follow up the perceived web browsing quality over time. To meet all these needs, we have designed and deployed in 2017 a new web browsing measurement tool, Web View.

Takeaways: *PhantomJS* makes use of web browsers' engines and not the regular graphical web browser used by end-users. This tool performs automated web browsing by requesting strictly the HTTP/1.1 protocol, which prevents objects to be downloaded in HTTP/2 or QUIC. Furthermore *PhantomJS* is not maintained anymore. *WebPageTest* offers a limited selection of web browsers and corresponding versions and offers W3C web metrics. Measurements can be performed on probes located at different geographic locations but these

[21] https://www.webpagetest.org/

[22] accessible by triggering the development mode of the web browser which Web View does

probes are not connected to residential access networks. The offered metrics when performing web browsing measurements are limited. *SiteSpeed.io* has been introduced in the last quarter of 2018 and makes use of web browsers' networking logs to calculate different web metrics. For the measurements to be the most end-user representative, the tool needs to be installed on laptops and desktops connected to residential access networks. The HAR file representing the web browsing measurement needs to be further processed to extract protocol distribution and types or location of web servers.

2.2 Our proposed tool: Web View

Web View is a web browsing measuring tool deployed in desktop devices and laptops meant to be the most end-user representative by making use of different web browsers being regularly updated and connected to residential network access networks. Our tool allows measuring a unique website, a list of websites or the top N Alexa[23] websites. Web View makes use of web browsers' networking logs to calculate different loading times but also retrieves W3C metrics loading times to compare offered timings. Web View offers fine-grained information on the Internet protocol distribution, location and types of web servers delivering contents, downloaded objects MIME-type and estimated network path taken by packets to reach web servers. All measurements are visually represented in real-time on a public visualization website, https://webview.orange.com.

2.2.1 Web View architecture

Web View is composed of six independent modules represented through Fig. 13. While some modules are meant to configure measurement parameters, others drive real web browsers, perform calculations to offer additional information from HAR files and represent the collected measurements visually.

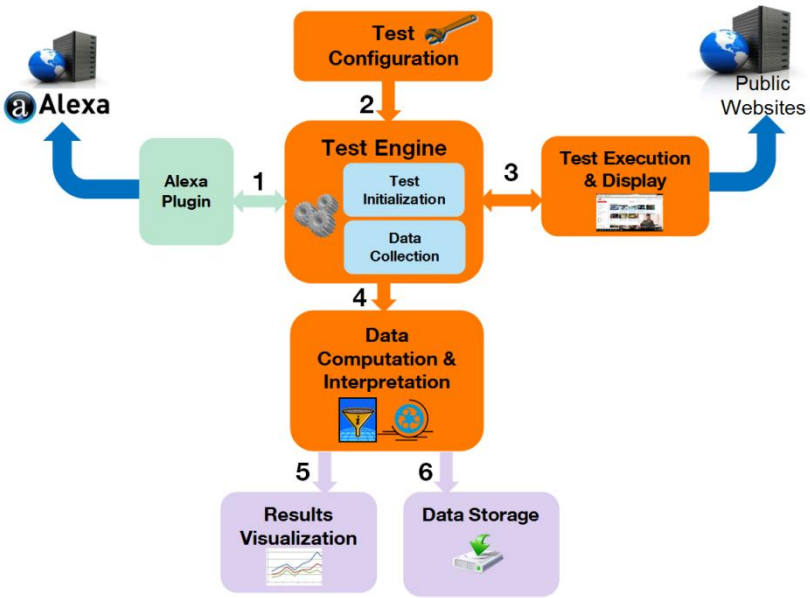


Fig. 13: Web View architecture

Behind the curtain: A bit apart from our tool, but useful as input for the configuration, a Python script in the *Alexa Plug-in* module collects on a daily basis the Top Ranked 1Million Alexa websites. This process is performed by a single probe and the list dispatched to all measurement probes. The collected data is automatically analyzed and the websites

[23] https://www.alexa.com/siteinfo

referenced with a unique identifier in a separate database, to further indicate the websites' rankings evolution through time.

Test Configuration module: This module is written in Python and verifies if input configuration parameters are known to Web View, i.e the requested functionality of the tool, Internet protocol and if the chosen web browser and version is installed.

Test Engine module: This module interacts with *Selenium*[24] which is the main automator. Upon input parameters, *Selenium* will drive a web browser thanks to its corresponding web driver, e.g *Selenium* will drive the `Google-Chrome v.78` web browser through the web driver `chromeDriver78`. The *Test Initialization* sub-module is meant to add needed options to the web browser, e.g auto-open the *devtools* console or `--private` option added if the measurement is meant to be performed in incognito mode.

Test Execution & Display module: This module launches the web browser with needed options and browses the designated homepage. A HAR file representing the web browser's networking logs is retrieved through a plug-in we have developed and will be discussed in section 3.2.1. All outputs from the console, e.g visible portion loading progression, W3C metrics, etc., are collected and handed to the *Data Collection* sub-module.

Data Computation & Interpretation module: From the collected HAR file, is calculated additional information, e.g downloaded objects' MIME-type along with the Internet protocol distribution, identification of web servers' type along with their estimated geographic location.

Results Visualization module: This module automatically parses obtained measurements which are then displayed on a public visualization website.

Data Storage module: This module is meant to send all measurements to a distant machine to be stored over time.

The different modules are all independent from each other which eases the addition of new functionalities. The different modules are written into the *Python* programming language.

2.2.2 Web View infrastructure

Web View [69] [121] is a measurement platform and its infrastructure is depicted in Fig. 14. Web View is composed of 3 main components: the probes, a mutualized database and a public visualization website showing the obtained measurement results. Although we perform measurements on the Top 10,000 Alexa websites, a limited number of homepages are represented on our visualization website.

Each probe is a user-oriented measurement tool whose main objective is to perform automated web browsing sessions, to measure representative information of web pages in order to better qualify and understand web browsing, both in terms of performance and delivery. The probes emulate end-users' web browsing within a real end-user environment: real web browsers, residential access network, etc. Actually, 17 probes are currently deployed at 5 different geographic locations: 10 in Lannion (France), 4 in Paris (France), 1 in Vannes (France), 1 in Curepipe (Mauritius) and 1 in Tokyo (Japan), represented in Table 6. Each Web View probe measures the Top 10,000 Alexa websites 24/7/365 by using different web

[24] https://www.seleniumhq.org/

browsers, namely *Google-Chrome* (versions 63, 68, 71, 73, 75, 76, 77, 78, 79) and *Mozilla-Firefox* (versions 63, 64, 66, 68, 69, 70, 71, 72).

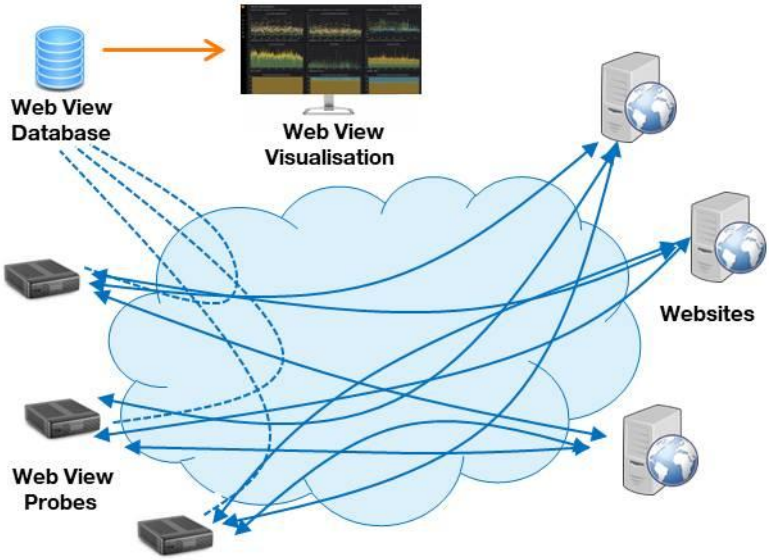


Fig. 14: Web View infrastructure

Type	Location	CPU	RAM	Network operator	Downlink		Uplink	
Desktop & Laptop	Lannion (France)	i5-2.5Ghz	8Go	Orange	ADSL Wi-Fi FIBER	10Mbps 200Mbps 800Mbps	ADSL Wi-Fi FIBER	1Mbps 200Mbps 300Mbps
Desktop	Lannion (France)	Xeon-2.8Ghz	12Go	Orange	ADSL FIBER	10Mbps 800Mbps	ADSL FIBER	1Mbps 300Mbps
Desktop	Vannes (France)	i5-2.5Ghz	8Go	Free	FIBER	800Mbps	FIBER	300Mbps
Desktop	Paris (France)	i5-2.5Ghz	8Go	Orange	FIBER	1Gbps	FIBER	300Mbps
Desktop	Paris (France)	i5-2.5Ghz	8Go	Free	FIBER	1Gbps	FIBER	570Mbps
Desktop	Paris (France)	i5-2.5Ghz	8Go	Bouygues	FIBER	1Gbps	FIBER	510Mbps
Desktop	Paris (France)	i5-2.5Ghz	8Go	SFR	FIBER	1Gbps	FIBER	470Mbps
Desktop	Curepipe (Mauritius)	i5-2.5Ghz	8Go	Mauritius Telecom	FIBER	10Mbps	FIBER	2Mbps
Cloud EC2-M5	Tokyo (Japan)	Xeon-2.5Ghz	8Go	Amazon	10Gbps		4Gbps	

Table 6: Web View probes

The different measurements performed are pushed to a mutualized database and represented on a public visualization website representing the corresponding loading times and estimated geographic location of web servers delivering contents.

Takeaways: Web View is a web browsing measuring tool deployed in desktop devices and laptops meant to be the most end-user representative by making use of different web browsers being regularly updated and connected to residential network access networks. Our tool is composed of independent modules written in Python and JavaScript. Probes are actually deployed in France and Mauritius to study the Web browsing eco-system.

2.3 Web View measurement functionalities

Web View is meant to perform automated Web browsing sessions upon different conditions such as assessing web pages' loading times, identifying the different web servers delivering content and the networking path (upstream and downstream) taken by network packets.

Web View offers 4 main functionalities during web browsing sessions:

- **lightDomains**: perform web browsing measurements upon desired parameters and provide information on loading times, estimated geographic location of web servers, Internet protocol distribution and objects MIME-type.
- **detailedDomains**: complementary to the *lightDomains* functionality, provide additional information on the types of web servers delivering content, i.e regular web server, web cache or CDN (Content Delivery Networks).
- **networkPath**: complementary to the *detailedDomains* functionality, *paris-traceroutes* are performed by our tool from the end-user device to the remote web server's IP address where the latter is exposed in the collected HAR file. Through the RIPE Stat API of RIPE NCC, *paris-traceroutes* can also be made from the remote AS (Autonomous Syste) to the end-user's public IP address.
- **dohMode**: complementary to the *lightDomains* functionality, we assess the perceived loading times when making use of public DNS resolvers such as *Google* (8.8.8.8) or *Cloudflare* (1.1.1.1) when using the Firefox v.70+ web browser.

Web View configuration parameters

For each measurement test, we have to specify different configuration parameters: the web browser we want to use, the access network of the probe since one probe is connected to only one access network which can be Fiber, ADSL or home Wi-Fi of different network operators, the transport protocol we want to evaluate to get the contents, the window size of the web browser we want to emulate, the use of an ad blocker or not and the list of websites to measure. It can be a list of pre-defined website(s) or the Top N Alexa websites. Measurements are launched through the command:

```
/usr/bin/initiator [requestedProtocol] [webBrower] [mode] [netwIface]
[adblockOption] [graphicsMode] [browserWidth] [browserHeight]
[functionality] [ipMode] [dohMode] [websites]
```

where

`[requestedProtocol]`: HTTP/1.1, HTTP/2, HTTP2_CACHE, HTTP2_REPEAT, QUIC, QUIC_CACHE, QUIC_REPEAT
`[webBrower]`: chrome-63/68/71/73/75/76/77/78/79 or firefox-63/64/66/68/69/70/71/72
`[mode]`: classic (regular usage of web browsers) or private
`[netwIface]`: corresponding network interface (ADSL, Wi-Fi or FIBER)
`[adblockOption]`: with or without ad blocker (*adBlockPlus v.3-7-0*)
`[graphicsMode]`: visible (graphical representation of web browser) or non-visible
`[browserWidth]`: 1920 / 1440 / 768
`[browserHeight]`:1080 / 900 / 1024
`[functionality]`:*lightDomains* / *detailedDomains* / *networkPath*
`[ipMode]`:strict IPv4 or dual IPv4-IPv6
`[dohMode]`:*withoutDOH* (regular operator's DNS) / *GoogleDOH* / *CloudflareDOH*

Since content servers might not implement all transport protocols, we allow fallback to protocols used by content servers. Typically, when performing measurements and requesting HTTP/1.1, we deactivate the HTTP/2 and QUIC protocol; requesting HTTP/2

implies deactivating QUIC but fallback to HTTP/1.1 is allowed; when requesting QUIC, we allow fallback to HTTP/1.1 and HTTP/2 for non-QUIC web servers; when requesting the Repeat mode (HTTP2_REPEAT or QUIC_REPEAT), we favor 0-RTT UDP and 1-RTT TCP connections by firstly navigating to the website, closing the web browser, clearing the resources' local cache but keeping the DNS (Domain Name System) cache. We then navigate once more to the website where measurements are collected. For every measurement, the resources' cache is always emptied except for the *_CACHE* (HTTP2_CACHE or QUIC_CACHE) mode and a timeout of 18 seconds is set to limit the impact of possible downtimes of content servers. This value has been derived from all our measurements.

Web View measurement results

For every visited website, Web View probes offer 84 parameters[25]. For the computation and collection of these parameters, we rely on the HTTP Archive (HAR) file which is the web browser's exposed networking logs. The use of networking information is privileged in order to be more accurate when calculating loading times of dynamic and progressive content, i.e Progressive Web Applications. From the obtained HAR, further calculations are performed in order to assess the protocol distribution through which responses are delivered to end-users and the location of web servers. Amongst the measured and computed 84 parameters by the Web View probes, we can mention 4 different loading times, namely the First Paint (FP), the Page Load Time (PLT from HAR and W3C), the Time for Full Visual Rendering (TFVR) which is a web metric that we have defined and will be discussed in section 3.2 and the processing time, all obtained from HAR files. The probe also provides information about resources[26] composing the web page, i.e the number of resources, their origin, type, size and transfer rate. Since content can be delivered using a different transport protocol than the one we requested, we compute the distribution of received protocols for the given webpage.


```
{"name": "X-Cache",
 "value": "MISS, HIT"},
{"name": "X-App-Cache",
 "value": "HIT"},
{"name": "X-Served-By",
 "value": "cache-iad2132-IAD,
cache-cdg20761-CDG"},
"serverIPAddress": "151.101.120.175"
```


Fig. 15: Part of HAR file related to CDN delivery

When requesting the `detailedDomains` functionality, information about the content server is deeply processed in order to identify if a resource was provided by the origin web server or a CDN provider. For this, we use a part of the HAR file which indicates if the resource is retrieved from a cache (*HIT*) or not (*MISS*). For example, in Fig. 15, the first server replies with a *MISS* (cache-miss), meaning that the resource is not in its cache and the second server replies with a *HIT* (cache-hit) meaning that the needed resource is present in its cache. The corresponding content is thus delivered by the second CDN. From the exposed IP address, we perform a *WHOIS* to query the registered assignees, which results into the CDN *Fastly*. Relying on the MaxMind GeoIP2 database[27], we then identify the geographic location of the web server (town, country and continent). From the exposed value `cache-cdg20761-CDG`, we can also identify on the fly that this cache is in Paris (CDG is the international airport code of Paris). As an example, when web browsing the website *tumblr.com* from a Web View probe in Europe, requesting the QUIC protocol with *Chrome*

[25] https://webview.orange.com/public/img/monitParam.png
[26] https://webview.orange.com/public/img/domainDetails.png
[27] https://www.maxmind.com/

web browser v.75, the domain *assets.tumblr.com* will be delivering 65 different objects through the QUIC protocol by using the CDN *Verizon* located into the *United States*:


```
{"wwwName":"tumblr.com",   "requestedProtocol":"QUIC",   "machine":"lili25",
"adBlocker":"Yes",    "networkIface":"FIBER",    "browserUsed":"chrome_v75″,
"hostName":"assets.tumblr.com", "country":"United States", cdn":"Verizon" ,
"geohash":[-97.822,   37.751],   "nbResH1":0,   "nbResH2":0,   "nbResQ":65,
"nbResP":0, "nbRes":65}
```


When requesting the `networkPath` functionality, for every distinct web server IP address delivering content, a *paris-traceroute* is carried out. From the obtained information, the corresponding IP addresses of routers on the path are correlated to the PyASN and RIPE Stat database to identify the corresponding network path taken by packets between different AS. As an example, when web browsing the homepage of *yahoo.com*, objects are downloaded from the domain *s.yimg.com* and corresponding upstream path for a Web View probe connected to the Orange operator is

`AS3215 - AS5511 - AS1299 - AS10310 - AS203070`

with identified AS holders being

`Orange - OpenTransit - TeliaNet - Yahoo Oath Holdings - Yahoo France`

And corresponding Round-Trip-Times towards each AS from the probe being

`9.08 ms - 9.12 ms - 10.78 ms - 42.12 ms - 10.71 ms`

where we can identify if a corresponding increase in RTT can impact loading times. The downstream path can also be measured thanks to the RIPE NCC probes through their exposed API where a RIPE probe in AS203070 can be selected and *paris-traceroute*s performed until reaching our public IP address. Although having assessed if the uplink and downlink path of network packets are the same, choosing a probe in AS203070 from RIPE NCC is a difficult task since the probes might be connected to different network operators which might not be the same used by `Yahoo France`.

Takeaways: Web View is meant to perform automated web browsing measurements and being the most end-user representative. Our tool offers several functionalities in order to obtain fine-grained information on the Web browsing eco-system. Web View provides as output 84 monitoring parameters such as the time to load web pages (or parts of them), the Internet protocol distribution, types of downloaded objects, location and types of web servers delivering content, the uplink network path of network packets from the client to the remote web server and finally the use of public DNS infrastructures.

2.4 Web View visualization tool

The Web View visualization tool is a public website, based on *Grafana*, and allows a straightforward visual analysis of our collected measurements. *Grafana* is connected to an *Elasticsearch* database, which stores all measurement results performed and sent by our Web View probes. The website offers several tabs illustrating how our platform works and has 2 main pages: one showing different panels related to the analysis of the web page browsing (loading times, resources, protocol distribution, etc.) and the second one representing content servers (CDNs or origin servers) on a world map, with information about resources and protocols. For the web page offering websites' analysis, a menu allows a user to filter out several parameters: select a specific website, transport protocol, access networks, location of probes, web browser's window size, used web browser and use of an ad blocker or not. For the web page representing servers delivering contents, the menu offers an additional filter to select a specific CDN provider.

Web pages' analysis

The first web page (*https://webview.orange.com*) of our visualization tool depicts information to better understand the implications of the Web eco-system during web browsing sessions. While some panels show the time needed to load a web page as a whole or through its progression by comparing different browsers (*Chrome* and *Firefox*), others compare the obtained timings when requesting specific transport protocols (HTTP/1.1, HTTP/2, QUIC). To better follow up with the deployment of the newest transport protocols, namely HTTP/2 and more recently QUIC, additional panels show the distribution of Internet protocols (how many resources are retrieved from web servers through HTTP/1.1, HTTP/2 or QUIC) as per a requested transport protocol. Additional information is also provided regarding the geographic location of web servers continent-wise. Following a set of filters selected by a user, all the panels are automatically updated.

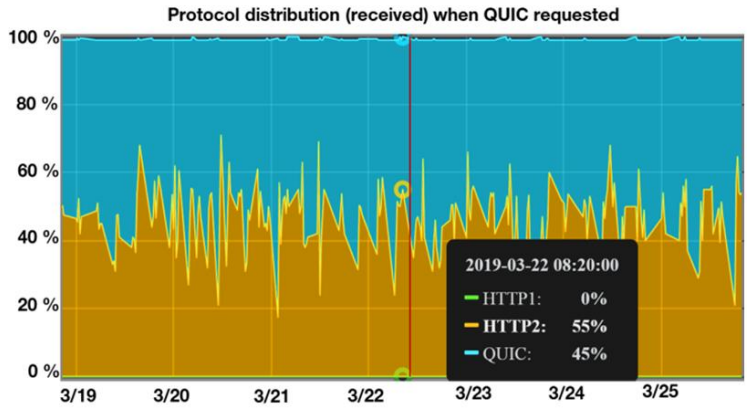


Fig. 16: Protocol distribution for the homepage *youtube.com*

For instance, Fig. 16 shows the protocol distribution when requesting QUIC and browsing the homepage of *youtube.com*. The web browser used is *Google Chrome* and we can see that even if *Google* promotes the QUIC protocol, on average 55% of contents are still distributed through HTTP/2 over TLS/TCP. When using an ad blocker, a mean number of 56 resources are downloaded from 8 different domains. These domains deliver all contents in a secured manner (HTTPS) and only 5 domains are QUIC-enabled.

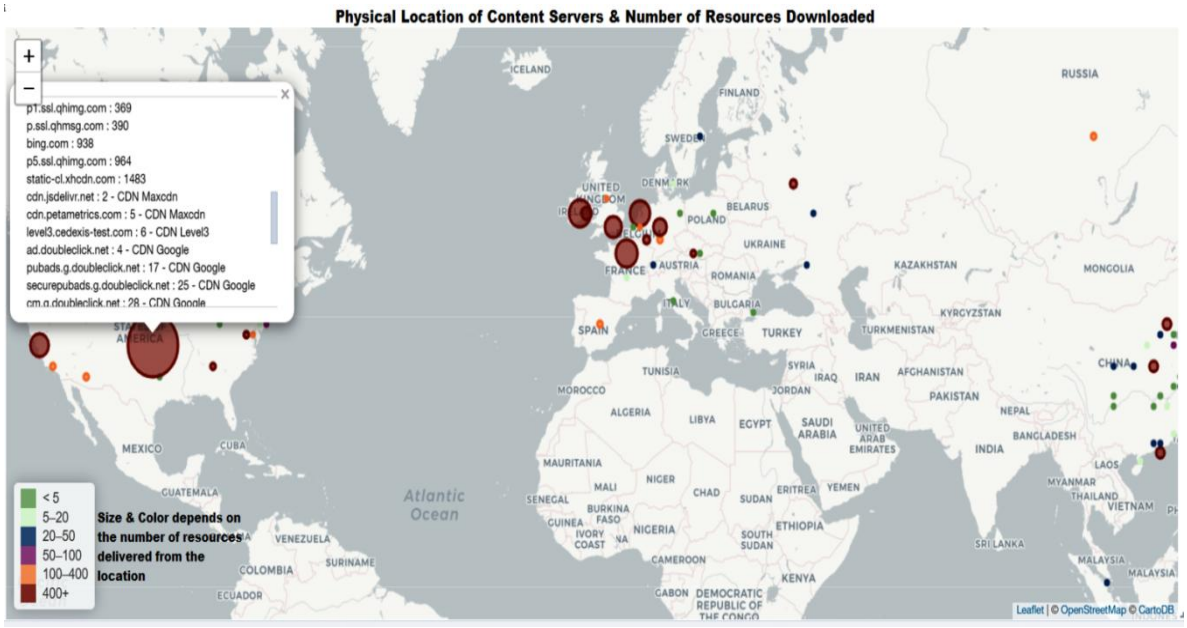


Fig. 17: Location of origin web servers and CDN delivering content

Content Servers Analysis

The CDN web page (*https://webview.orange.com/d/UyIIcrUmz/*) allows laying out figures on the delivery of contents, and mainly which web server provides a corresponding resource, the type of service delivered and their geographic location. An end-user located in Europe expects to firstly download objects from the same homepage domain and secondly from web servers located in Europe. Our visualization website allows detecting that content might be fetched from different web servers at different geographic locations mainly due to Content Delivery Networks edge servers. Domains delivering contents and having an authoritative DNS name server different from the one of the homepage are entitled *Non-Origin* domains and conversely *Same-Origin* domains.

For instance, we can see through Fig. 17 that when an end-user is located in France and performs web browsing on the Top 50 Alexa websites, contents are mainly retrieved from France and Europe. Nevertheless, a non-negligible amount of content is also downloaded from North America and Asia. The different circles represent the distribution of content following the geographic location of web servers. Any user can hover the different circles and obtain fine-grained details about the web servers and the amount of content served. The Fig. 17 popup window illustrates the web servers delivering contents from a datacenter located in the Kansas County, United States.

Additional panels represent the Internet protocol distribution (HTTP/1.1, HTTP/2 and QUIC) from servers located all around the globe when performing web browsing measurements. Content servers referenced as *"No CDN"* or *"No CACHE"* implies that they are regular origin web servers. From all our measurements performed on a set of websites, the list of CDN providers is automatically updated.

The Web View visualization website shows measurements performed since February 2018 where different web browsers have been used. Since web browsers are often updated on average every 20 days, the platform is also regularly updated. Measuring websites' loading times on large time-spans helps to identify the impact of actors of the Web ecosystem which can increase or decrease loading times. The Web View website being public, everyone can access it and analyze different behaviors.

Takeaways: Web View is a tool meant to perform automated web browsing measurements and also offers a public visualization website representing all performed measurements in real-time. Any end-user can visit the website and select on the fly different measurements parameters, e.g location of the probe, used web browser, residential network access type, etc. Upon this selection, the different loading times are put forth, as well as the content distribution such as location and type of web servers delivering content.

2.5 Conclusion

The Web browsing ecosystem is complex since it involves several actors and when visiting a website, content is delivered by web servers located all around the world. Several web servers are implicated in the delivery of content due to a large amount of services embedded in web pages. With the aim to better understand the Web ecosystem in order provide better QoS, several tools have been introduced over the last years by corporate companies or researchers.

When focusing on Web browsing quality, three main tools are actually used, namely *PhantomJS*, *WebPageTest* the last new comer *SiteSpeed.io*. These tools may not be always end-user representative as the tools are not deployed on end-user representative devices, do not embark the graphical version of web browsers or request the latest Internet protocols. In order to be the most end-user representative and provide information on the different factors contributing to web pages' graphical representation, we have introduced a new web browsing tool, Web View. The latter although being able to be deployed in VMs (Virtual Machines),

we have chosen for our study to deploy our tool into end-user representative devices, i.e laptops and desktops. Web View is composed of different independent modules written in Python and JavaScript which eases the continuous upgrade of the tool, as web browsers are usually upgraded every 20 days.

Compared to other existing tools, Web View can be deployed rapidly to assess Web browsing quality by offering 84 monitoring parameters. Web View embarks to the best of our knowledge in 2019, all available web metrics, such as the First Paint (FP), Above-The-Fold (ATF) and the Page Load Time (PLT). In addition to web pages' loading times, Web View also sheds light on the different actors implicated in content delivery, i.e location and types of web servers, Internet protocol distribution, types of downloaded objects, etc. All measurements are represented in real-time on a public visualization website (https://webview.orange.com) where end-users, service companies or researchers can better understand the Web browsing eco-system.

The latest web browsing tool introduced is *SiteSpeed.io* and like Web View, it makes use of real graphical web browsers during web browsing measurements and loading times are derived from web browsers' exposed networking logs. The Table 7 shows the different offered capabilities of the tool Web View versus the mostly used web browsing tools. When making use of other tools, specifying a wide range of web browsers is not possible and although web browsers' networking logs are retrieved and stored on the device, further calculations have to be performed to have a precise and objective view of the web browsing ecosystem. Furthermore, no web browsing tool estimates the uplink and downlink path of network packets in real-time.

Last but not the least, DoH (DNS over HTTPS) has been lately introduced to put forth privacy of end-users. When favorising the DoH, public DNS, e.g Google, Cloudflare, OpenDNS, Quad9, etc., will perform DNS resolution over HTTPS rather than the network operator to which a device is connected. It is important to compare the pros and cons of public DNS usage. As an example, for an end-user located in France, the network operator will resolve the different domain names to web servers located the closest to an end-user, i.e in Europe. When making use of public DNS, the different domain names are resolved to web servers located in North America which can increase the average download time of resources.

Over the last 2 years, thanks to measurements performed by Web View and represented in real-time on a public monitoring website, we have been able to detect root causes of degraded Quality of Experience, e.g From the 4th to 7th November 2017, Google web servers delivered content in HTTP/1.1 and HTTP/2, where QUIC protocol was deactivated on all their web servers. Web View visualization platform also allows to follow in real-time the evolution of the Web ecosystem, i.e adoption of Internet protocols by large service companies, use of new Web technologies or routing strategies adopted by service providers.

Metrics / Functionality	WebPageTest	SiteSpeed.io	Web View
End-user representative device		✓	✓
Residential network access			✓
Disk usage of local machine		✓	✓
RAM usage of local machine			✓
Measurement in strict IPv4			✓
Measurement in dual IPv4/IPv6	✓	✓	✓
DNS caching			✓
DNS over HTTPS (DoH)			✓
Resources caching (local machine)			✓
Web browser classic mode	✓	✓	✓
Web browser private mode	✓		✓
Web browsers versions	✓		✓
Web browser window size			✓
Requested protocol			✓
Ad block use	✓	✓	✓
Use of web browser's networking logs		✓	✓
Number of downloaded objects	✓	✓	✓
Types of downloaded objects		✓	✓
Location of web servers (continent)			✓
Location of web servers (town or country)			✓
Web servers' IP address		✓	✓
Origin or Non-Origin web servers			✓
Type of web server (origin or CDN)			✓
Visible portion of web page without scrolling	✓	✓	✓
RTT of origin web server			✓
Protocol distribution of downloaded objects			✓
Secured download of objects (HTTP/HTTPS)			✓
First Paint (FP)	✓	✓	✓
Protocol distribution before First Paint			✓
Types of downloaded objects before FP			✓
Above-The-Fold (ATF) through videos			✓
ATF browser-based	✓	✓	✓
Protocol distribution before ATF			✓
Types of downloaded objects before ATF			✓
RUM-SpeedIndex		✓	✓
Time for Full Visual Rendering (TFVR)			✓
Protocol distribution before TFVR			✓
Types of downloaded objects before TFVR			✓
PLT W3C	✓	✓	✓
PLT HAR		✓	✓
DNS time		✓	✓
Networking time		✓	✓
Total web page size		✓	✓
Visible web page size			✓
Estimated network uplink path between client and web servers			✓
Real-time visualization of measurements			✓

Table 7: Comparison of web browsing tools offered metrics

Chapter 3

A new Web metric: Time for Full Visual Rendering

Contents

In this chapter we firstly go through commonly used web metrics to measure web pages' loading times and assess their objectiveness as per newly introduced Web technologies. We have studied the objectiveness of these web metrics as they are embarked in our tool, Web View. Following different studies on web browsing behaviors or web browsing patterns, end-users generally wait for the visible portion of web pages to be fully loaded before scrolling them down or navigating through websites. Although new measurement techniques have been introduced to measure this visible portion loading time, they bear several inefficiencies with regard to dynamic web pages embarked with new Web technologies. Secondly, to circumvent those inefficiencies, we have designed and developed the Time for Full Visual Rendering (TFVR) which is a new web metric providing fine-grained loading times being independent of web pages' structure and embedded Web technologies. Compared to commonly used web metrics, the TFVR provides more accurate loading times by taking into account all objects rendered into the visible portion of the web browser with a mean extra computational time of 0.156 seconds.

3.1 Actual web metrics qualifying web browsing quality

Web pages go through different phases, as discussed in section 1.3, until being fully rendered where certain parts can be visible only by scrolling a web page. The de-facto used web metric over the last years to qualify web browsing quality has been the Page Load Time standardized by the W3C (PLT-W3C), which measures the time to load an entire web page.

With the introduction of smart phones and tablets, together with end-users making use of desktop screens of different sizes, the visible portion of a web page at first glance without scrolling differs from one end-user to another. To be able to measure this visible portion loading time, web metrics such as the Above-The-Fold (ATF) and Time To Interactive (TTI) have been introduced.

The PLT-W3C, as discussed in section 1.4, is naturally exposed by all web browsers. The ATF web metric can be calculated in two different ways: the use of video recording and web browser's exposed W3C[28] information. The TTI introduced in 2018 is exposed by the Paint Timing API, discussed in section 1.4, and only available in Chromium-based web browsers, e.g *Google-Chrome* or *Brave* (so far in November 2019).

3.1.1 Inefficiencies of the PLT-W3C

The PLT is the time needed to load an entire web page. The exposed loading time takes into account the network time to request and download every needed object from a remote web server and also the needed time by the web browser to process and render these downloaded objects. The PLT can be calculated in two different ways: by extracting information directly from the web browser implemented W3C information, which is commonly known as the PLT-W3C or calculated from the networking logs exposed by the web browser, which we denote, the PLT-HAR.

In order to calculate the PLT-W3C, the Resource Timing API exposed in section 1.4 is used. Fig. 18 shows the output from the Resource Timing API when web browsing the homepage of *wikipedia.org*, where a total number of 5 objects (highlighted in yellow `length: 5`) are downloaded. To calculate the PLT-W3C, every downloaded object's duration time (highlighted in green for the 4th downloaded object) is summed up to their corresponding processing time by the web browser. The PLT-W3C value through this example is 315ms.

The different needed objects to render a web page are downloaded from several web servers which may have their *time-allow-origin*[29] unexposed. When the Resource Timing API wants to obtain the different timings regarding the request and download of an object, the returned values can be zero; which is the case for the 4th downloaded object in Fig. 18 where the duration is 0ms (highlighted in green). Although the processing time for this object is exposed by the web browser, the time to request and download this object is not taken into account. This leads to an incorrect PLT-W3C value, which is the first inefficiency. In our example for *wikipedia.*org, the duration is not exposed for only 1 object and this duration's time contribution to the PLT-W3C is negligible. But from measurements performed over the Top 1 Million Alexa websites, for some homepages more than 200 different objects can be downloaded and up to 52% of objects' duration time can be 0 ms, and hence offering a PLT-W3C value being 48% short.

A second inefficiency of the PLT-W3C is with regards to the introduction of Progressive Web Applications (PWAs) which aim to offer a mobile visual environment through a desktop web browser. PWAs make use of several *service workers*[30] which are reactive upon an end-user action when browsing the corresponding web page e.g re-size images and web page layout automatically. The needed objects to render the web page are already downloaded and the PLT-W3C is exposed. But a simple movement of the mouse in the web browser window can lead these *workers* to re-build the Document Object Model (DOM) and Cascading Style Sheet Object Model (CSSOM) tree. Re-building the DOM and CSSOM modifies the display of the web page and sometimes additional objects are downloaded. Since the PLT-W3C is already exposed, these additional actions are not taken

[28]World Wide Web Consortium

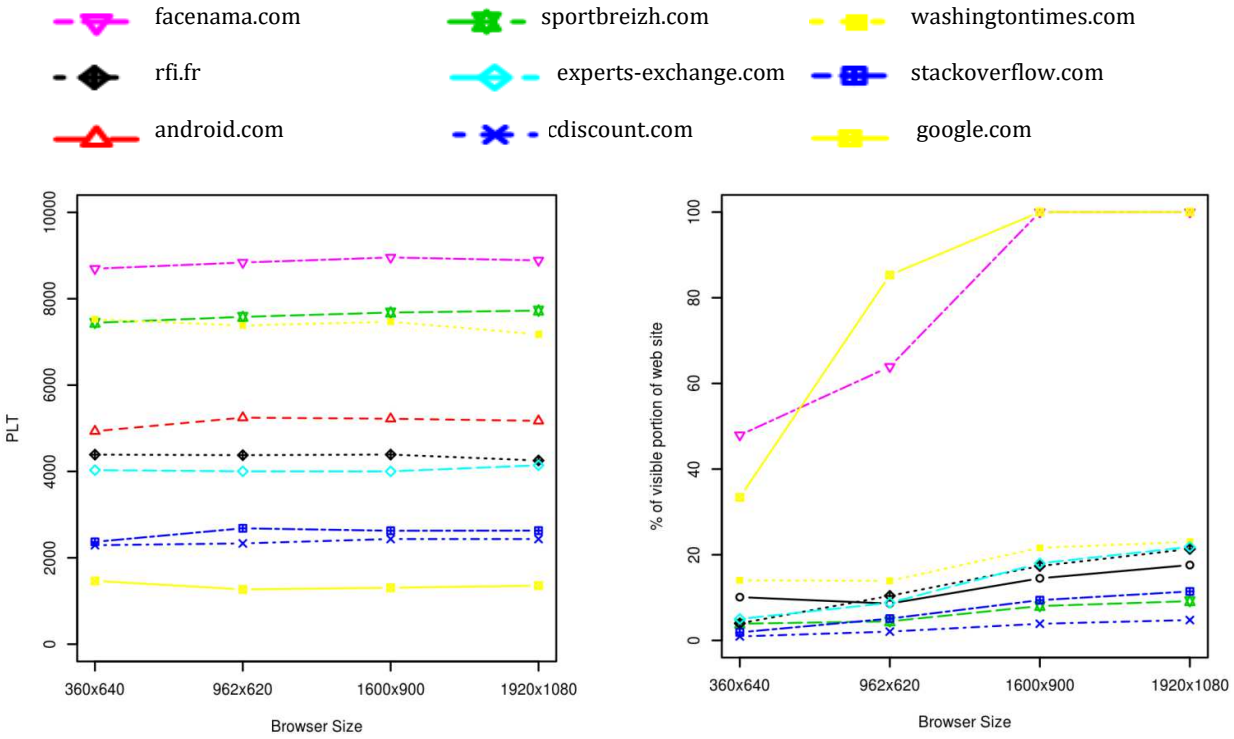
[29]The Time-Allow-Origin response header specifies origins that are allowed to see values of attributes retrieved via features of the Resource Timing API, which would otherwise be reported as zero due to cross-origin restrictions.

[30]Scripts that run in the background in the end-user's web browser

```
0: PerformanceResourceTiming {initiatorType: "img", nextHopProtocol: "h2", workerStart: 0,
redirectStart: 0, redirectEnd: 0, …}
1: PerformanceResourceTiming {initiatorType: "script", nextHopProtocol: "h2", workerStart: 0,
redirectStart: 0, redirectEnd: 0, …}
2: PerformanceResourceTiming {initiatorType: "script", nextHopProtocol: "h2", workerStart: 0,
redirectStart: 0, redirectEnd: 0, …}
3: PerformanceResourceTiming
        connectEnd: 0
        connectStart: 0
        decodedBodySize: 0
        domainLookupEnd: 0
        domainLookupStart: 0
        duration: 0
        encodedBodySize: 0
        entryType: "resource"
        fetchStart: 0
        initiatorType: "css"
        name: "https://www.wikipedia.org/portal/wikipedia.org/assets/img/sprite-81a290a5.svg"
        nextHopProtocol: ""
        redirectEnd: 0
        redirectStart: 0
        requestStart: 0
        responseEnd: 0
        responseStart: 0
        secureConnectionStart: 0
        serverTiming: [PerformanceServerTiming]
        startTime: 0
        transferSize: 0
        workerStart: 0
__proto__: PerformanceResourceTiming
4: PerformanceResourceTiming {initiatorType: "css", nextHopProtocol: "h2", workerStart: 0,
redirectStart: 0, redirectEnd: 0, …}
length: 5
__proto__: Array(0)
```

Fig. 18: Loading times exposed by the Resource Timing API

into account, yielding an unprecise page load time value. The PLT-W3C as implemented into web browsers does not take into consideration the effects of PWAs.



(a) PLT-W3C for different window sizes (b) Visible portion upon web browser window sizes

Fig. 19: PLT-W3C and visible portion upon different viewports

A third inefficiency of the PLT-W3C, as per end-users' web browsing behaviors, is that it exposes the time to load the entire web page. End-users make use of different devices' window sizes and primarily concentrate on the visible portion at first glance [75] before

scrolling down or navigating into the website. Taking into account the PLT-W3C to qualify end-users' Quality of Experience (QoE) is inappropriate since depending on the homepage visited, the visible portion at first glance may range from 1% to 100%. Fig. 19 shows the loading time and visible portion at first glance of 9 different homepages when using a web browser with different window sizes. As shown in Fig. 19(a), the PLT stays the same, except some minor changes in timings due to the network state, although the web browser window sizes increase. Through Fig. 19(b) following a web browser window size increase, the visible portion at first glance without scrolling also increases. Although the visible portion of web pages increases as per an increased web browser window, the PLT stays merely the same and it is hard to qualify the web browsing experience of end-users.

Takeaways: The PLT-W3C as implemented into web browsers does not offer accurate timings since it uses the Resource Timing API which cannot retrieve the corresponding request and download time of objects if a distant web server does not allow it. With the introduction of PWAs, scripts run permanently in the end-user's web browser to change the display or download additional objects upon end-users action. The download of these additional objects is not captured by the PLT-W3C. End-users generally concentrate on the visible portion of websites at first glance and devices' screens come in different sizes. The PLT measures the time to load the entire web page, visible or not and is not adapted to measure finely end-users' perceived QoE.

3.1.2 Inefficiencies of the ATF

The Above-The-Fold (ATF) metric is meant to measure the time to load the visible portion of a web page at first glance without scrolling. The ATF web metric can be calculated in two different ways: the use of video recording and web browser's exposed W3C information.

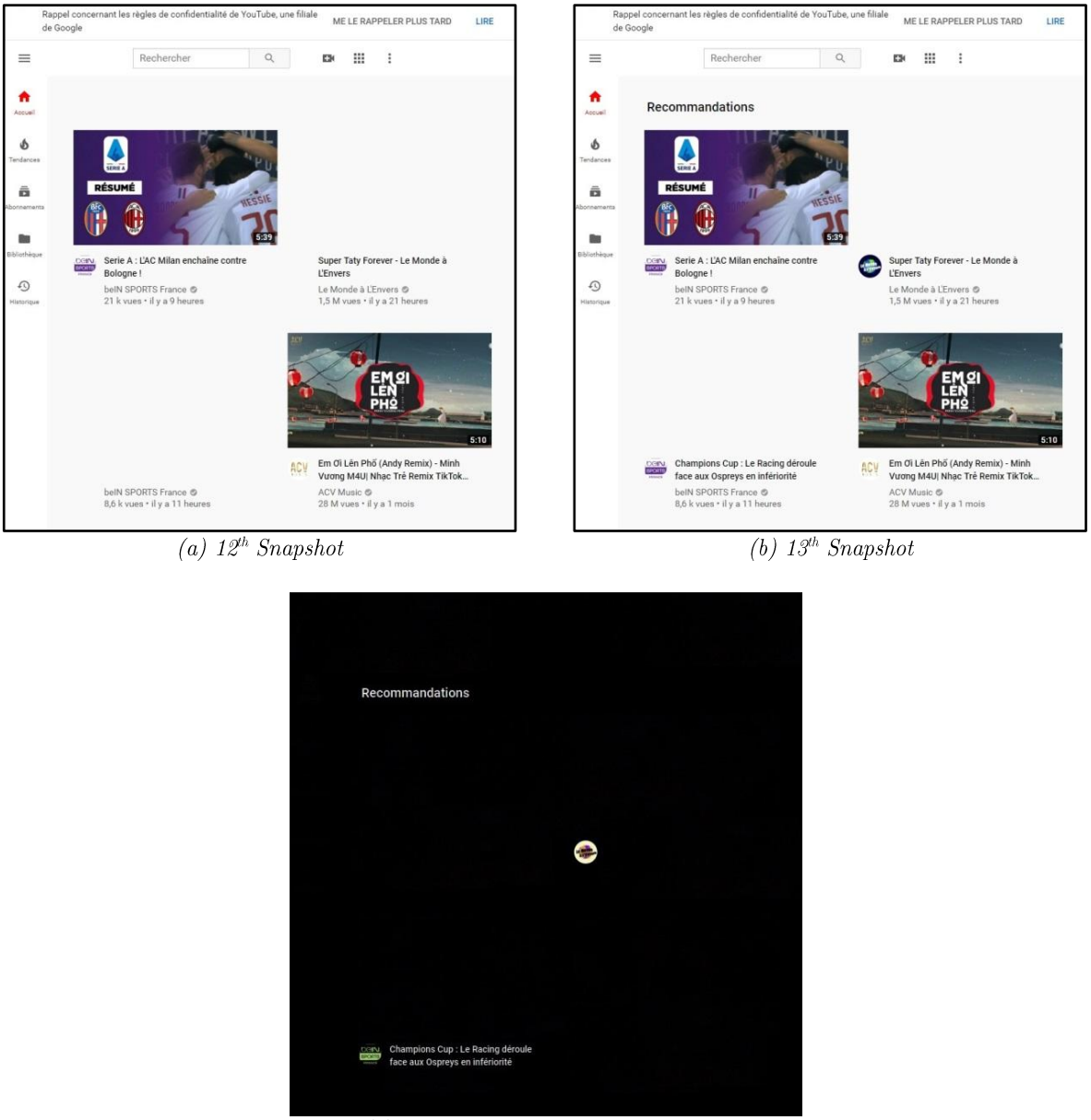
Calculating the ATF through video recordings

When calculating the ATF through video recordings 3 main steps are involved:

1. Perform a video recording of the visible portion of web page loading progression until fully rendered,
2. From the video recording, extract k-images snapshots,
3. Compare the different images among them to define at what moment pixels between two images do not change anymore, which indicates the moment the visible portion of the web page fully rendered.

During the video recording phase, the web browser is firstly opened and the recording started by selecting manually the region inside the end-user's screen to be recorded. The website *url* is then added into the web browser address bar and the web page requested. When the visible portion of the web page is visually estimated to be fully rendered, the video recording is stopped. At this very stage, we can identify 3 inefficiencies which are selecting manually the region to be recorded, estimating visually when the visible portion is loaded to stop the video recording and lastly the use of tools being external to the web browser which can impact web browsing measurements. Human interaction in the video recording process is compulsory.

From the recording is then extracted k-images snapshots represented by $Sn_1, \ldots, Sn_k$. As an example, if the recorded video is 30 frames per second, a recording ratio of 150 will create an image snapshot every 5 seconds. The value of k cannot be pre-defined on the fly since it depends on the frame rate of the video and its total duration. From the obtained images snapshots, the pixels are then compared between them, i.e between Sn_1 and Sn_2 until



(a) 12th Snapshot *(b) 13th Snapshot*

(c) Difference between Sn_{12} and Sn_{13}

Fig. 20: Identifying missing pixels between snapshots for *youtube.com*

Sn_{k-1} and Sn_k. The *python* library `ImageMagick` offers a method `compare` to find the amount of different pixels between two images:

```
compare -metric AE snapshot12.jpg snapshot13.jpg null: 2>&1
```

where zero difference in pixels' amount means that two snapshots are identical. If there is no pixel change between Sn_1 and Sn_2 but a pixel change between Sn_1 and Sn_3, the image snapshot Sn_2 is discarded. This process is performed until reaching the image snapshot which does not change anymore. Fig. 20 illustrates how missing pixels can be identified for the visible portion of the homepage of *youtube.com* when using an ad blocker. While Fig. 20(a) shows the 12th snapshot Sn_{12} extracted from the video recording, Fig. 20(b) shows the 13th snapshot Sn_{13}. These two snapshots are compared and the underlying difference is illustrated by Fig. 20(c) where 3412 pixels are missing between these two snapshots.

Calculating the ATF from browser-exposed web metrics

To lift the burden of making video recordings and pixels comparison, a browser-based ATF loading time can be obtained from the Real User Monitoring SpeedIndex (RUM-

SpeedIndex), discussed in section 1.4. This browser-based ATF technique makes use of the web browser's DOM information to identify when the rendering of objects have finished.

This technique follows the assumption that the $ATF \leq PLT$ where the ATF should be equal to the PLT if the visible portion of a web page is 100%. This ATF browser-based web metric makes use of the Resource Timing API and every downloaded object is assessed if it is rendered into the visible portion of the web page through DOM information. If this is the case, for every object the networking time is retrieved from the Resource Timing API and processing time from the web browser's DOM information. The processing and networking time for all objects rendered into the visible portion is then summed up to provide the ATF time. The way the browser-based ATF is designed faces issues in its calculation:

1. The time to request and download objects is based on the Resource Timing API and as discussed in section 3.1.1, this duration value is not always exposed.
2. Accessing the DOM information is costly and an *event* happens every time a node of the DOM or CSSOM tree is re-built. As an example, to render the web page of *youtube.com*, 73 objects are downloaded among which 42 objects are rendered into the visible portion of the web browser window size 1920x1080. The DOM and CSSOM trees are modified 3168 times to render the entire web page. The way the browser-based ATF algorithm is designed makes that these 3168 events will be assessed 73 times to identify which objects are rendered into the visible portion and corresponding time.
3. To identify if an object is rendered into the visible portion of the web page, JavaScript code has to be injected into the command line of the web browser and only images are assessed if rendered into the visible portion.

Takeaways: The process of calculating the ATF from videos is time-consuming where human interaction in the whole process is compulsory. High definition videos increase the accuracy of the pixels comparison but reduce the end-user's device processing and storage capacity. Furthermore determining the needed k snapshots from the video is not an easy process. The pixels comparison task is also time-consuming as it depends on the number of snapshots. With the proliferation of advertisements in the visible portion of the web page, another question arises from this measurement technique since advertisements change regularly. It is thus hard to define when the video recording should be stopped and if pixels from advertisings should be taken into account. The browser-based ATF technique can expose incorrect timings since it uses the Resource Timing API information where networking information is not always exposed. The RUM-SI algorithm only takes into account images rendered into the visible portion of the web page.

3.1.3 Inefficiencies of the TTI

The Time To Interactive (TTI) is the moment when the visible portion of a web page is estimated to be fully rendered by the web browser and when an end-user can start interacting with the web page. The TTI as exposed in section 1.4 is not a standardized web metric and only available in the *Google-Chrome* web browser. The TTI calculation is performed as:

1. The started time is at the First Contentful Paint (FCP) moment.
2. As from the FCP moment, the web browser looks for a moment when the network is stable during 5 seconds.
3. When this *no-network* transaction is identified, the algorithm assesses backwards the different DOM and CSSOM events to identify the latest longest task performed. The TTI is then exposed.

The first inefficiency of the TTI is that it can be calculated only when the entire web page is fully rendered to identify among all events the latest longest task. The second inefficiency of the TTI is that it cannot be detected until no requests are made or responses

received by the web browser during 5 seconds. Web pages are nowadays most of the time dynamic where a simple movement of the mouse can trigger tens of objects to be downloaded, the network is thus rarely idle more than 5 seconds. As an example, for the homepage of *chinatimes.com*, the PLT-HAR is 5.07 seconds when using *Google-Chrome* version 68 for a European end-user. The exposed TTI is 2.89 seconds for a web browser size 1920x1080 but calculated 17.89 seconds prior to navigation-start; i.e the network is stable during 5 seconds, 12.89 seconds prior to navigation-start. Based on our measurements performed for the Top 10,000 Alexa websites, an average time of 9.71 seconds is needed to calculate an average TTI of 1.98 seconds.

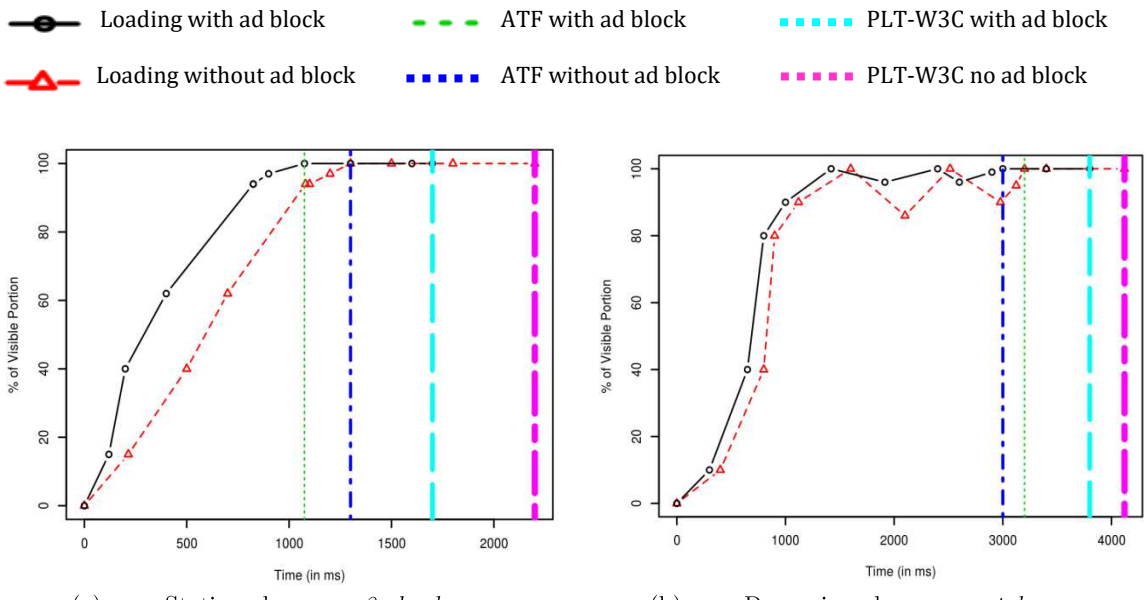
Takeaways: The TTI is only implemented in the *Google-Chrome* web browser and can be calculated if and only if the network state is idle during 5 seconds. Web pages are mostly dynamic nowadays and network states are rarely idle, which leads the TTI not to be easily exposed.

3.1.4 Accuracy of actual web metrics

Putting apart the techniques to calculate the ATF time, we now go through different scenarios where new Web technologies strongly impact the exposed ATF timings. When measuring the ATF for web pages not being 100% visible at first glance, i.e the web page needs to be scrolled to reach the end of the web page, the ATF should logically follow the rule $ATF < PLT$. Fig. 21 depicts the loading times of two different homepages, *w3schools.com* being a static web page and *youtube.com* being a dynamic web page. As shown through Fig. 21(a), for a static web page, the corresponding loading times with and without the use of an ad blocker follows the rule:

$$ATF_{with\_adblock} < ATF_{without\_adblock} < PLT_{with\_adblock} < PLT_{without\_adblock}$$

The use of an ad blocker helps in reducing the ATF by 190 ms and 1 more object is downloaded and rendered when not using an ad blocker. The PLT-W3C is increased by 960ms when not using an ad blocker. On the other hand, when measuring the dynamic homepage of *youtube.com* as shown in Fig. 21(b), the visible portion of the web page fluctuates when not using an ad blocker since advertisements change regularly, and thus the pixels change more often. The use of video recording identifies that the $ATF_{without\_adblock}$ is smaller than the $ATF_{with\_adblock}$ since the advertising panel occupies a bigger part of the visible portion and thus less images are rendered.



(a) Static web page: *w3schools.com* (b) Dynamic web page: *youtube.com*

Fig. 21: Loading times of static versus dynamic web pages

Without the use of an ad blocker, 39 objects are rendered into the visible portion but with an ad blocker, the advertisement panels disappears and the dynamic web page renders 43 objects into the visible portion. The networking and processing of these 4 additional objects increase the ATF by 189 ms. The PLT-W3C value on the other hand when using an ad blocker is 215 ms lower than when an ad blocker is not used since the download and processing of the advertising panel does not happen.

Another scenario is represented through Fig. 22 where the web page of *forgeofempires.com* is composed of a large number of Flash[31] objects. The PLT-W3C is 1350 ms less than the observed ATF and thus $ATF > PLT$. A JavaScript is fired 450 ms after web navigation starts which triggers the download of 2 additional objects, namely an image and an asynchronous JavaScript. The image is processed and rendered into the web browser but the asynchronous JavaScript processing is delayed. Fig. 23 shows the visual progression to fully load this web page. At that very moment, the PLT-W3C is 700ms, i.e Fig. 23(a), meaning that the web browser detects that the web page is entirely loaded. The asynchronous JavaScript is processed 100 ms after the exposed PLT-W3C to download 297 additional objects where Fig. 23(b) shows the loading progression bar at 35%, Fig. 23(c) with a progression bar at 80% and Fig. 23(d) when the visible portion is entirely loaded. In this example, due to 1 asynchronous JavaScript, the PLT-W3C offered by web browsers is biased. The ATF helps in identifying the corresponding visible loading times, but for the web page of *forgeofempires.com*, a video of 45 seconds had to be made to identify the ATF of 1815.7 ms.

On the other hand, when using the browser-based ATF technique, the exposed timing was 496 ms since it relies on the PLT-W3C time-frame to identify downloaded objects. Following this real-life scenario, the PLT-W3C and browser-based ATF is biased since the download and rendering of nearly 300 objects is not captured. Although the default ATF technique using video recordings works fine, the ATF timing was exposed after more than 4 minutes of calculations (45sec for the video recording, 1 minute to create images snapshots and 3 minutes to compare pixels).

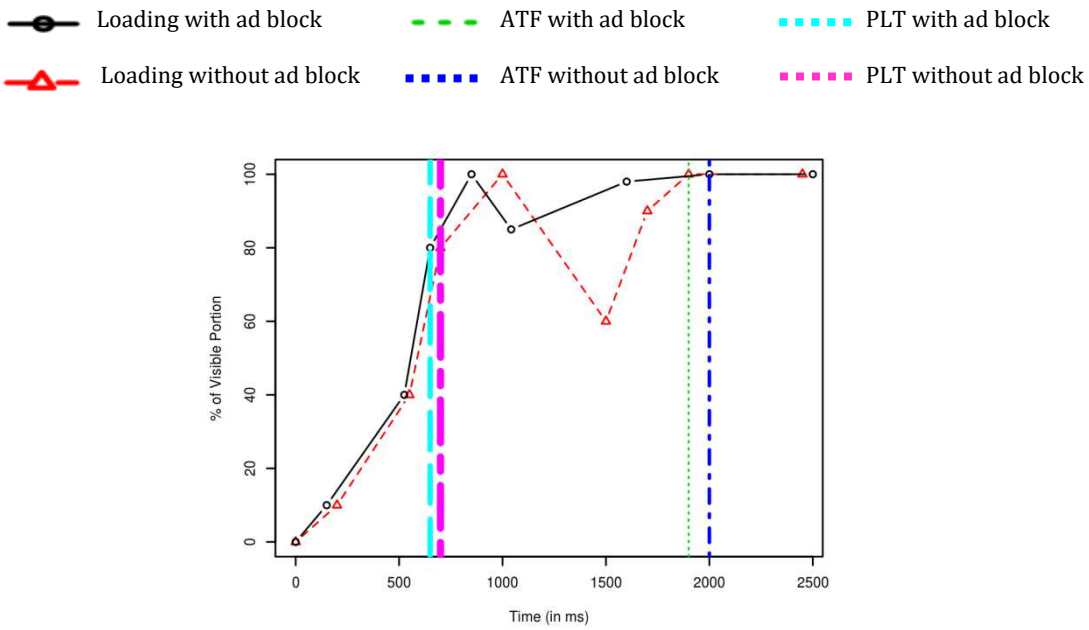


Fig. 22: Side effects of asynchronous JavaScript on loading times

[31] Flash objects will not be supported anymore by 2020 by the Google-Chrome web browser but is still used in web pages

(a) PLT-W3C exposed (b) 35% progression (c) 80% progression (d) Web page loaded

Fig. 23: Loading progression of *forgeofempires.com* web page

Takeaways: Actual web metrics are not efficient to offer fine-grained loading times since they are differently implemented in different web browsers by large service companies and have not evolved to cope with new Web technologies. The use of external tools to identify corresponding web pages' loading times can be impacted since more computing power will be used by the device during the measurement. A new web metric is needed to measure finely the loading time of the visible portion of web pages, without using external tools to capture loading progression or actual web metrics like the Resource Timing API.

3.2 Our proposed web metric : The TFVR

The Time for Full Visual Rendering [83] is a browser-based measurement technique which is browser-type and browser-version independent. Our web metric makes use of web browsers' networking logs to offer fine-grained loading times regarding the visible portion of a web page at first glance without scrolling. The TFVR also offers detailed information on the different objects downloaded and rendered into this time frame, i.e step by step processing phases for every object and identify through which Internet protocol objects are downloaded as well as the geographic location of the corresponding web servers.

3.2.1 The TFVR design and mechanics

The TFVR is entirely written in JavaScript and interacts with the web browser engine through the *devtools*[32] console. The *devtools* is accessed through the *F12* keyboard button for any recent web browser where an end-user can interact with the engine of the web browser, as shown in Fig. 24. For the TFVR calculation, no external tools are needed and calculations are performed in real-time. Being written in JavaScript, the TFVR web metric can also be added as a plug-in in different web browsers.

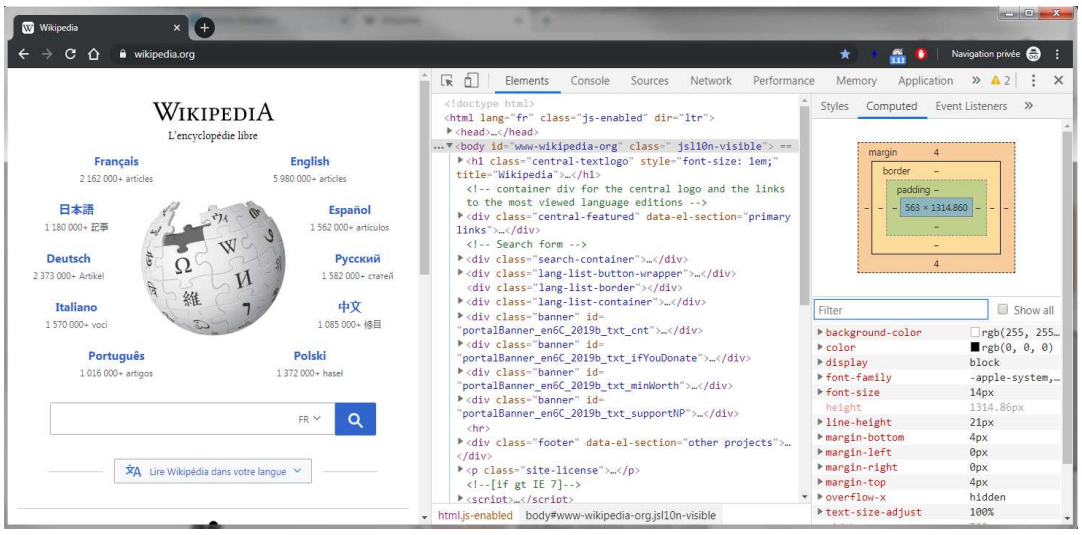


Fig. 24: *Devtools* console

[32] Is a frontend developer toolbox built-in web browsers since 2015

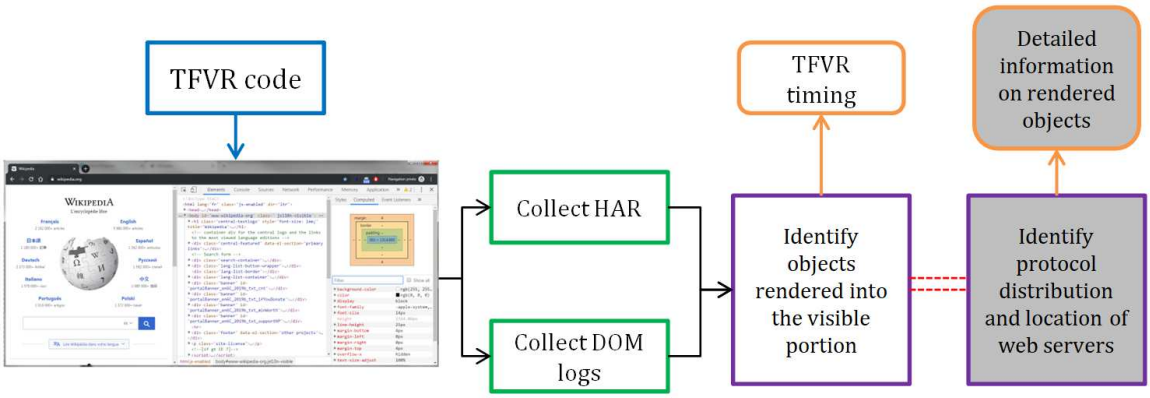


Fig. 25: TFVR design and mechanics

The overall design and mechanics of the TFVR is shown in Fig. 25. In order to obtain the TFVR timing and detailed information on objects rendered into the visible portion of the web page, the TFVR JavaScript code is injected into the web browser's *devtools* console and the calculations go through 3 main steps:

1. The web browser's networking logs are collected into the form of a JSON (JavaScript Object Notation). To be able to perform additional calculations, these networking logs are stored on the end-user device into a compressed HAR (Http ARchive) file. From these networking logs is extracted the name of downloaded objects, e.g *https://www.website.com/object.txt* and the corresponding durations since navigation-start. Using the web browser's network logs palliates to the inefficiency of the Resource Timing API; durations for downloading objects are never null values.
2. The web browser's DOM logs is collected, which expose all events occurred into the web browser's engine since navigation-start.
3. From the identified objects being downloaded $\{objectName,\ durationSinceNavigationStart\}$ in step 1, a comparison is made with the DOM logs to identify which object is rendered into the visible portion of the web page without scrolling. The TFVR timing is then exposed.

Collecting web browsers' networking logs

All modern web browsers, since 2015, offer their networking logs through HAR files via the *devtools* console. To obtain these networking logs, built-in functionalities in respective web browsers are triggered via a plug-in we have developed which is independent of the used web browser. When using a *Chrome* web browser, the corresponding HAR is collected through the `chrome.devtools.network.getHAR()` command and through `extensions.netmonitor.har` option for the *Firefox* web browser. These commands trigger built-in functionalities of the web browsers and our plug-in measures the moment when the DOM and CSSOM are entirely unloaded, i.e the corresponding DOM tree is fully built and not triggered anymore, which means that the web page is entirely loaded.

Web browsers continue downloading different objects, even if a web page is entirely loaded in order to reduce loading times when performing further web navigation across the same website. In order to capture all objects downloaded and rendered by the web browser, we poll every 30 ms the DOM and CSSOM information offered by the web browser during a maximum time of 2000 ms to identify if the DOM is rebuilt. We use this mechanism in order to detect finely the execution of asynchronous JavaScript which are sometimes not captured through traditional web metrics as discussed in sections 3.1.1 and 3.1.2. Retrieving and going

through the entire logs of the DOM can be costly, this is why we only poll if the number of *events* in the array has increased or not via `window.document.getElementsByTagName('*').length`. To identify the maximum polling time of 2000 ms, we have measured the Top 1 Million Alexa websites and a statistical analysis has shown that the DOM tree can be rebuilt in this 2000 ms lapse of time. The average number of polls has shown to be 12, i.e the DOM tree can be rebuilt on average 360 ms after the PLT-W3C value is exposed. In order to be precise in our offered timings, if after 15 consecutive polls (450 ms) in the 2000 ms time frame, the DOM is never rebuilt, we collect the networking logs. This helps in identifying if an asynchronous JavaScript is fired after the *onLoad* event, e.g an asynchronous JavaScript is fired 100 ms after the *onLoad* event for the website *forgeofempires.com* and this event is captured by our plug-in and thus the additional download of 297 objects.

Identifying objects downloaded and rendered into the visible portion

From the obtained HAR, we identify all objects downloaded since navigation-start by the web browser. As an example, Table 8 shows the different objects downloaded and the moment they are rendered in the web browser for the entire web page of *wikipedia.org*. While the PLT-W3C is exposed as being 315 ms for this web page, additional objects are downloaded and rendered 602 ms after navigation-start. Compared to Fig. 18 in section 3.1.1, 2 additional objects are downloaded, so that the entire web page is correctly displayed. These 2 additional objects are not captured by the PLT-W3C web metric and the exposed loading time is incorrect. Furthermore through the analysis of web browser networking logs, we retrieve detailed download timings for the object `sprite-81a290a5.svg`, which the Resource Timing API exposed as being 0 ms (highlighted in green) through Fig. 18.

	Downloaded objects	**Time after navigation-start (ms)**
1.	https://www.wikipedia.org/portal/wikipedia.org/assets/img/sprite-81a290a5.svg	177.53
2.	https://www.wikipedia.org/portal/wikipedia.org/assets/js/gt-ie9-a2995951ca.js	169.95
3.	https://www.wikipedia.org/	112.15
4.	https://www.wikipedia.org/portal/wikipedia.org/assets/js/index-c1cb7f1287.js	168.97
5.	https://www.wikipedia.org/portal/wikipedia.org/assets/img/Wikipedia-logo-v2.png	558.51
6.	https://www.wikipedia.org/portal/wikipedia.org/assets/img/Wikinews-logo_sister.png	601.19

Table 8: Downloaded objects for *wikipedia.org* from HAR

The object `gt-ie9-a2995951ca.js` is an asynchronous JavaScript whose request and download is captured by the PLT-W3C metric, but not its processing which downloads additional contents. Thanks to our plug-in mechanism to retrieve the HAR, at the 13th poll, the download and rendering of 2 additional objects needed by this asynchronous JavaScript is detected. The side effects of asynchronous JavaScript has been studied in [54] and through this example, we can identify that actual web metrics do not cope with new Web technologies since the exposed PLT-W3C is 315 ms but the real PLT, which we denote by PLT-HAR, to load the entire web page is 652 ms.

Once we have obtained the precise list of downloaded objects for the entire web page, we assess if they are rendered in the visible portion of the web browser window. To obtain the visible portion of the web page, we identify the visible surface area without scrolling in the web browser. All downloaded and processed objects are placed at specific positions when rendered; positions are defined by (x, y) coordinates through an `offsetTop` and `offsetWidth` value exposed by the DOM logs. If the object is rendered into the visible surface (`innerHeight` x `innerWidth`) area as shown in Fig. 26, the object is considered to be visible. Depending on the web browser's visible window size, an object might also appear in-between the visible and non-visible surface area but since a downloaded object is fully rendered by a web browser, it is considered as being visible. As exposed in [83] texts and scripts do not have an (x, y) coordinate[33] when processed and rendered into the visible portion of the web page. We thus retrieve their duration time since navigation-start and assess if they are processed before another image found into the visible surface area. When all objects to be rendered into the visible portion of the web page are identified, the highest time value represents the TFVR, i.e the moment when the last object is rendered. The TFVR calculation process represents a mean extra computational time of 0.156 seconds for the Top 10,000 Alexa websites where on average the processing power of the end-user device increases by 0.014%.

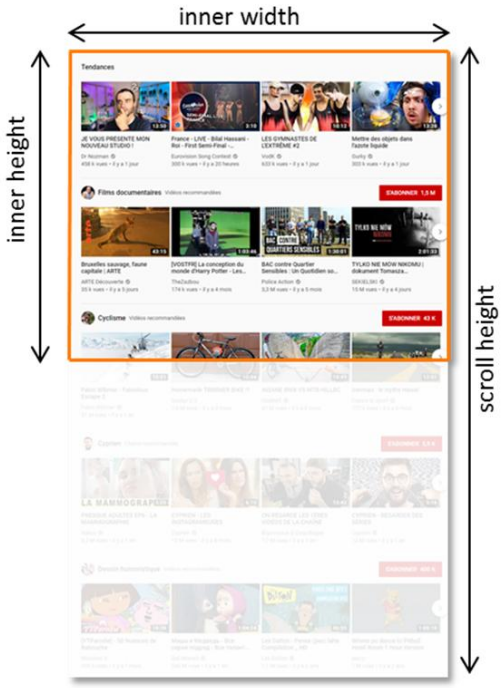


Fig. 26: Visible surface area of web pages

Takeaways: The Time for Full Visual Rendering is a browser-based measurement technique which is browser-type and browser-version independent. The TFVR is calculated in real-time with a mean extra computational time of 0.156 seconds by making use of the web browser's offered networking logs. The TFVR takes into account new Web technologies and captures events triggered by JavaScript or Progressive Web Applications.

3.2.2 Additional functionalities of the TFVR

The TFVR is meant to calculate the time to fully load the visible portion of a web page but also offers means to identify through time the rendering phases of objects as well as the Internet protocol used for delivery, the MIME-type (Multipurpose Internet Mail Extensions) of downloaded content and estimated geographic location of web servers.

Objects' rendering phases through time

To render the entire webpage *wikipedia.org*, a total number of 936 DOM *events* happen to process the downloaded 6 objects. Let's consider the object `sprite-81a290a5.svg` represented through Table 8 which is rendered 177.53 ms after navigation-start. Table 9 illustrates the 9 different steps through which the corresponding DOM and CSSOM tree is rebuilt in order to render that object.

[33]As from Chrome v.68 (July 2018) and Firefox v.61 (June 2018), these two web browsers expose the `offsetTop` and `offsetWidth` for all objects; we have updated our web metric algorithm.

The object `sprite-81a290a5.svg` is downloaded and rendered in 45.41 ms (Step 1 to 9). A sprite is a collection of images packed together, being small in size and its processing starts the first time at step 4 and lasts 14.93 ms (to reach step 5). As illustrated in Table 8, a JavaScript `index-c1cb7f1287.js` is downloaded and processed 168.97 ms after navigation-start. This JavaScript processes the object `sprite-81a290a5.svg` at 169.23 ms (step 6). The object `sprite-81a290a5.svg` is cut into different images in 5.89 ms (to reach step 7) and finally rendered (all images extracted from this sprite) 177.53 ms after navigation-start.

Steps	Event Id	Object name	Time after navigation-start (ms)	Event
1	96	sprite-81a290a5.svg	132.12	Requesting object
2	99	sprite-81a290a5.svg	141.09	Download started
3	122	sprite-81a290a5.svg	143.20	Download ended
4	184	sprite-81a290a5.svg	144.01	Processing started
5	315	sprite-81a290a5.svg	158.94	Processing ended
6	384	sprite-81a290a5.svg	169.23	Re-processed
7	392	sprite-81a290a5.svg	175.12	Processing ended
8	451	sprite-81a290a5.svg	176.04	Rendering started
9	484	sprite-81a290a5.svg	177.53	Rendering ended

Table 9: Rendering phases of each object

This exposed information helps in troubleshooting why the object number 5 (`Wikipedia-logo-v2.png`) illustrated in Table 8 is rendered 558.51 ms after navigation-start. The delay in triggering the download and rendering of the object `Wikipedia-logo-v2.png` is due to the object number 2 (`gt-ie9-a2995951ca.js`) which is an asynchronous JavaScript.

Object name	Object Type	Object Size (KB)	Duration (ms)	Delivery Protocol	Web server	Location
index.html	text/html	58.5	112.15	HTTP/2	Web server	United States
index-c1cb7f1287.js	JavaScript	8.2	168.97	HTTP/1.1	Cache	Netherlands
gt-ie9-a2995951ca.js	JavaScript	3.2	169.95	HTTP/2	Cache	Netherlands
sprite-81a290a5.svg	image/svg	18.2	177.53	HTTP/2	Cache	Netherlands

Table 10: Downloaded objects for *wikipedia.org* with window size 768x1024

Offering detailed information on objects rendered into the visible portion

An additional step which is optional, illustrated through Fig. 25 in section 3.2.1 with a grey background allows to obtain further information on content rendered into the visible portion, i.e their MIME-type, the Internet protocol through which they have been downloaded (obtained from the HAR through the `httpVersion` HAR header) or the estimated geographic location of the remote web server (correlation between HAR header `serverIPAddress` and Maxmind GeoIP2 database[34]).

From Table 8, only four downloaded and processed objects are needed to fully render the visible portion of the homepage *wikipedia.org* when having a web browser window size of 768x1024, where 81% of the entire web page is visible at first glance to the end-user. The TFVR is 178.12 ms. In order to have a more precise overview of the different objects downloaded and rendered in 178.12 ms, the TFVR also exposes their MIME-type, Internet

[34] http://www.maxmind.com

protocol through which they have been requested and downloaded and the location of web servers delivering content. Table 10 shows the downloaded objects for *wikipedia.org* during the TFVR time lapse for a European end-user using a Firefox web browser with window size 768x1024 and requesting HTTP/2.

The type of web server delivering content, as discussed in section 1.2, can be an origin web server, cache or CDN. To identify if a particular content is downloaded from a cache or CDN, we analyze its `X-Cache` header in the HAR file. If this header is not present in the HAR file for this object, this means that the content is not served by a CDN or cache. Following the presence of this header, its value can be:

- `MISS`: The content is *cachable* but was not present into the cache edge web server and retrieved from the origin web server.
- `HIT`: The content was retrieved from a cache edge web server.

If the value of the `X-Cache` header is `HIT`, we retrieve from the HAR file the web server's IP address which is correlated to RIPE NCC[35] database to obtain the corresponding Autonomous System (AS). As an example when performing measurements for the website *ebay.com*, 2 JavaScripts are downloaded from the domain *assets.adobedtm.com* where the `X-Cache` value is `HIT`. The corresponding web server exposed IP address is `2.2.77.19` and when correlated to the RIPE NCC database, the AS holder is *AKAMAI*. When comparing the IP address to *Maxmind GeoIP2 database*, the edge web server is estimated to be in Ireland. When having a `HIT`, if the AS Holder belongs to the major CDN companies[36], we estimate the remote web server to be a CDN and conversely a cache, e.g *WIKIMEDIA*.

Takeaways: In addition to the time to load the visible portion of a web page, the TFVR also exposes the step by step processing phases for every downloaded object and identify through which Internet protocol they are downloaded as well as the geographic location of the corresponding web servers.

3.2.3 Accuracy of the TFVR

We assess in this section the accuracy of the TFVR versus the ATF where the latter can be calculated either through video recording or from the web browser's exposed Resource Timing information.

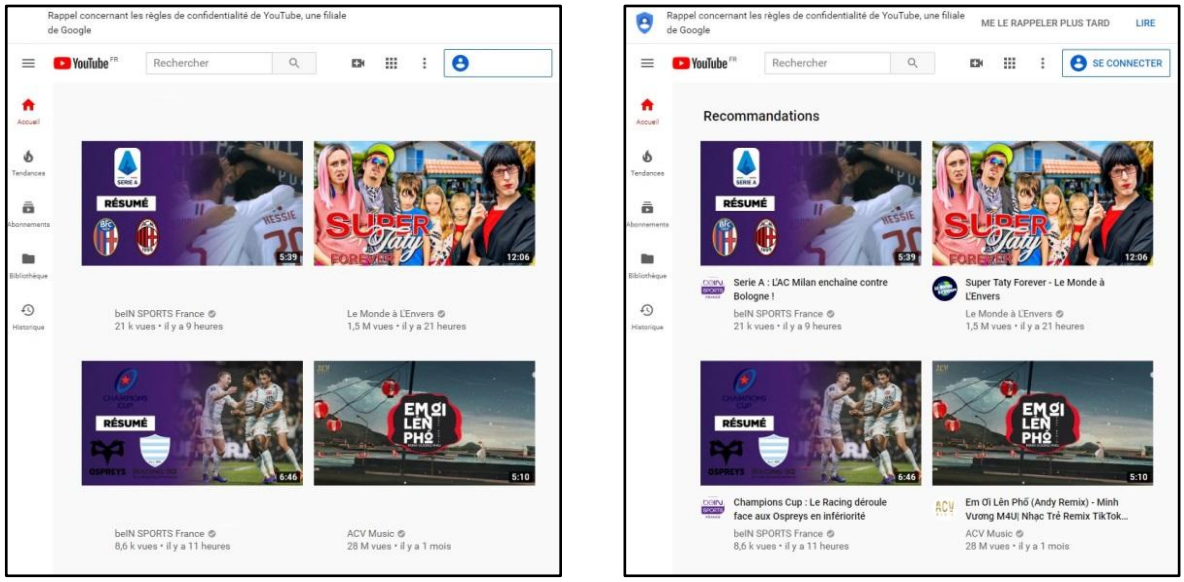
Rendering output

When making use of the ATF to calculate the time to load the visible portion, some objects are not taken into account for two reasons. Firstly when making use of videos to calculate the time to render the visible portion, dynamic content can change very often and the pixel comparison phase as well as the identified moment to stop the video recording can be a tedious task. Secondly, when calculating the browser-based ATF (through the RUM-SpeedIndex), only downloaded objects into the PLT-W3C lapse of time are taken into account. As identified in section 3.1.1, the PLT-W3C does not always take into account all needed objects to render a web page.

Fig. 27 shows the fully rendered visible surface area of the web page *youtube.com*, at the browser-based ATF and TFVR moment. When performing the ATF browser-based calculation, the ATF is 1278 ms and Fig. 27(a) shows the graphical representation of the web page at that moment. Compared to Fig. 27(b) several texts, images or headers are missing. When comparing the pixels of Fig. 27(a) with the final visual rendering of the *youtube.com* homepage at the PLT-HAR moment 21% of pixels are missing. On the other hand the comparison for the TFVR indicates 0.7% of pixels missing.

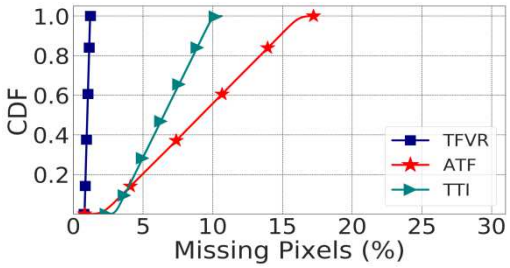
[35] https://stat.ripe.net/
[36] https://www.cdn-advisor.com/companies/

On a wider scope, Fig. 28 depicts the amount of missing pixels following different web browser window sizes, with (or without) ad block use for the Top 1,000 Alexa websites when comparing with the visible portion at the PLT-HAR moment. For a window size 768x1024, when using an ad blocker, less than 60% of the measured homepages have less than 1% pixels missing for the TFVR, 7.3% for the TTI and 10.6% for the browser-based ATF. Without an ad blocker under same measurement conditions, the amount of missing pixels increases by 3.7% for the TTI and 8.1% for the browser-based ATF. This shows that the TFVR calculation is more accurate compared to the browser-based ATF or TTI with lowest number of missing pixels with or without ad block use.

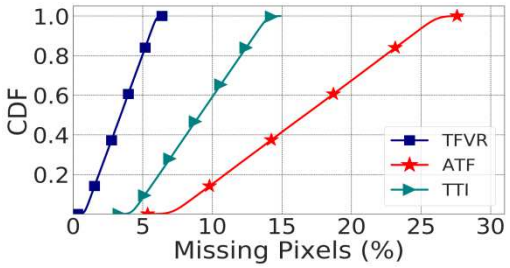


(a) ATF calculation (b) TFVR calculation

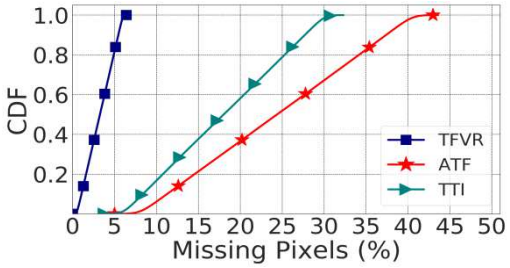
Fig. 27: Visual representation of *youtube.com*



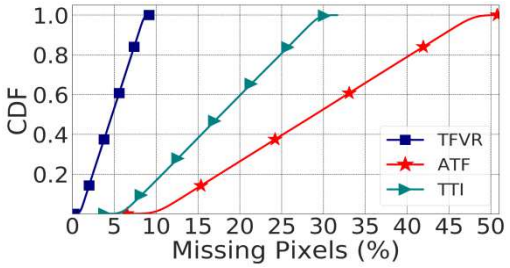
(a) With ad block and window size 768x1024



(b) Without ad block and window size 768x1024



(c) With ad block and window size 1920x1080

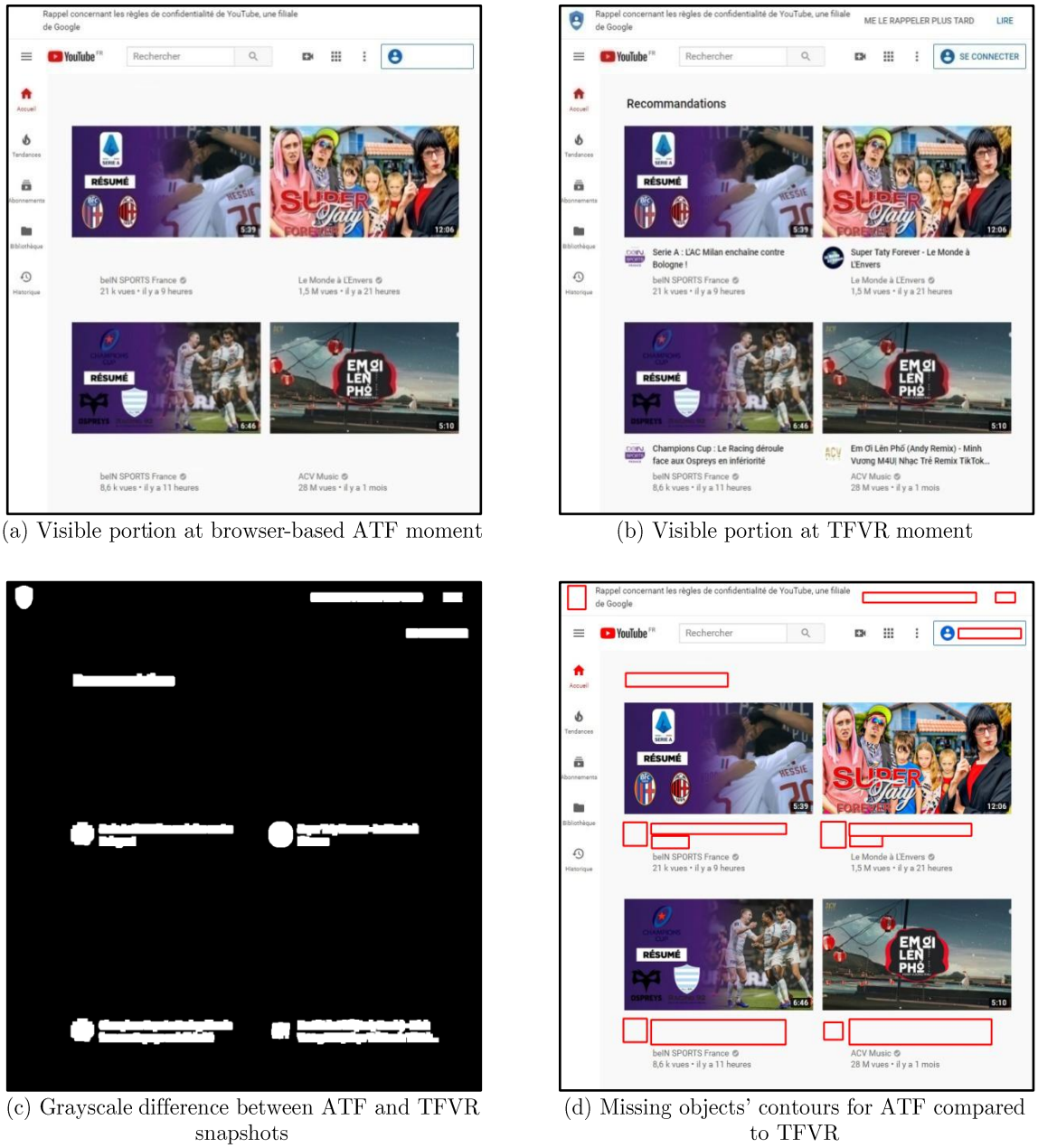


(d) Without ad block and window size 1920x1080

Fig. 28: Missing pixels compared to visible portion at PLT-HAR

Since the amount of missing pixels is more important for the TTI and browser-based ATF compared to the TFVR, we now focus on the amount of missing objects. We compare the screenshot of the visible portion of the web page for the ATF and TTI versus the TFVR. The identification of missing objects is done by:

1. We make use of the Structural Similarity Index (SSIM) [122] which helps to identify at which (x, y) coordinate location the image differences occur,
2. From the obtained (x, y) coordinates, we retrieve from the DOM information which objects should be rendered at this position,
3. The number of missing objects is then summed up.



(a) Visible portion at browser-based ATF moment

(b) Visible portion at TFVR moment

(c) Grayscale difference between ATF and TFVR snapshots

(d) Missing objects' contours for ATF compared to TFVR

Fig. 29: Identification of missing objects contours

Fig. 29 shows the rendering output of the homepage of *youtube.com* for a European end-user connected to an ADSL network access and using a Firefox web browser with window size 1440x900. The Fig. 29(a) depicts the visible portion at the browser-based ATF moment and Fig. 29(b) the visible portion at the TFVR moment. These two images are converted to grayscale and the SSIM is calculated. The exposed SSIM is 0.944 and 263,670

pixels are missing between both images. The threshold value which minimizes the weighted within-class variance through the Otsu Binarization algorithm [123] between these two grayscale images is then calculated. From the threshold value, the contours of missing objects are then drawn, depicted by Fig. 29(c) and Fig. 29(d). From the (x, y) coordinates of these contours, a comparison is performed with the DOM information to identify which downloaded objects should have been rendered in the corresponding region with a red contour are missing.

Fig. 30 depicts the overall distribution of missing objects for the Top 1,000 Alexa websites when having a web browser window 1920x1080. Compared to the TFVR, for less than 80% of measurements, when using an ad blocker, less than 18 objects are missing for the TTI and less than 25 objects for the browser-based ATF.

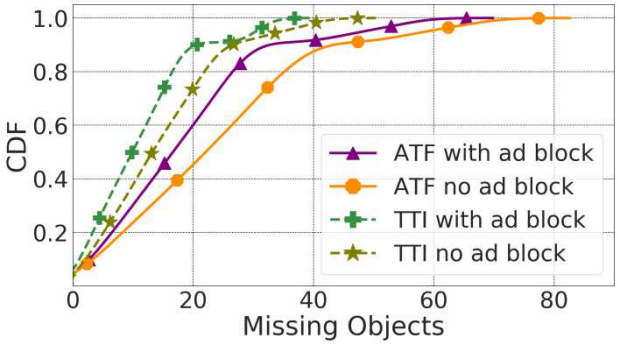
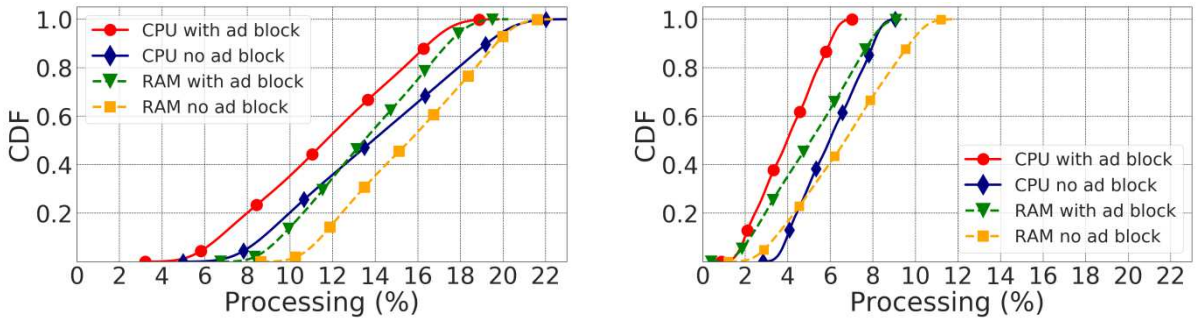


Fig. 30: Missing objects not taken into account compared to the TFVR



(a) ATF through video recording

(b) ATF browser-based

Fig. 31: Computing power increase compared to the TFVR calculation

Computing power

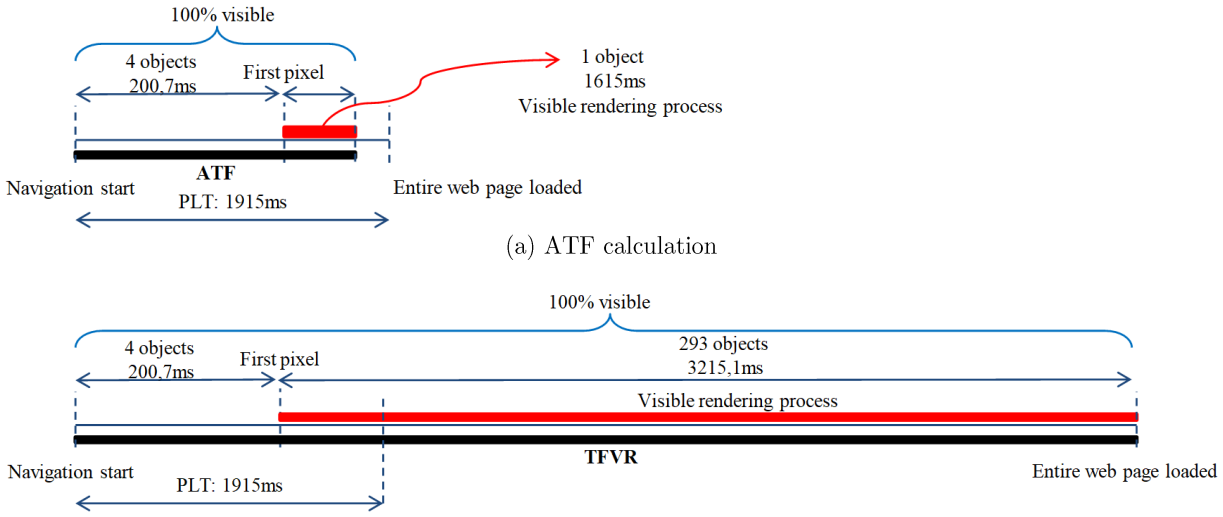
The calculation of the browser-based ATF where DOM information is assessed several times or through video recording needs more computing power, compared to the TFVR. Fig. 31 shows the increase in CPU (Central Processing Unit) and RAM (Random Access Memory) usage compared to the TFVR calculation. The use of video recording to calculate the ATF needs more computing power since a video recording is firstly made, followed by a decomposition into several chunks of images and finally a comparison of pixels between images. Both techniques meant to calculate the ATF lead to an increase in computing power and hence can impact web browsing measurements.

When measuring the time to render the visible portion of the web page, the ATF can be calculated by making use of external tools, meant to record the loading progression. We have noticed that an end-user device processor usage increases by 8% on average and memory usage by 9%. Furthermore to be able to provide precise loading times, video recordings are on average done during 40 seconds, although the visible portion is loaded into less than one second which is mainly due to advertisements in a web page.

Loading times

When comparing the obtained loading times of the visible portion, the ATF value is on average 26.9% higher compared to the TFVR where the video recording and pixels analysis process increase the usage of the end-user's device processing capacity, and hence impact web browsing measurements.

When comparing the browser-based ATF measurement technique making use of the Resource Timing API versus the TFVR, ATF exposed timings can be up to 90% less than the TFVR timings, which is mainly due to timings for different objects not offered by the Resource Timing API. Furthermore, with the usage of asynchronous JavaScript as discussed in section 3.1.2, additional downloaded objects to be rendered in the visible portion are not captured, which is illustrated in Fig. 32(a) when browsing the web page *forgeofempires.com*. Thanks to the TFVR as shown in Fig. 32(b), the entire loading process is captured where the visible surface area is loaded into 3415.8 ms instead of the exposed ATF loading time being 1815.7 ms.



(a) ATF calculation

(b) TFVR calculation

Fig. 32: Loading time of the visible portion of *forgeofempires.com*

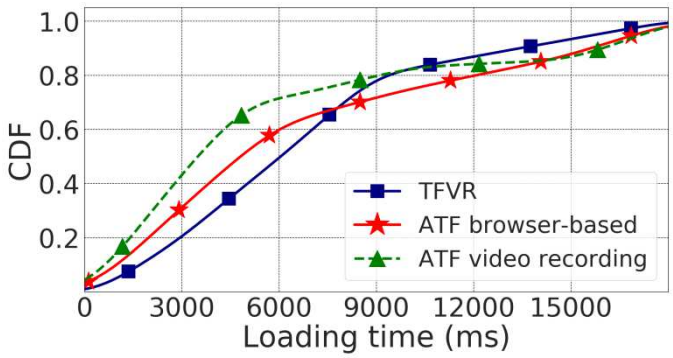


Fig. 33: Loading time for the TFVR and ATF with window size 1920x1080

On a wider scope, for the Top 1,000 Alexa websites, when compared to the TFVR, ATF loading times can be smaller or greater as illustrated in Fig. 33. The value for ATF through video recording is smaller for less than 80% of the measurements and for the browser-based ATF less than 70% of the measurements. This is mainly due to bad pixels' comparison for the ATF through video recording and missing objects not taken into account

for the browser-based ATF. The ATF values may also be greater than the TFVR due to extra computing power needed for the ATF calculation through videos.

When the visible portion of a web page is 100%, the Page Load Time should be equal to the time to load the visible portion. The Fig. 34 shows the different loading times as defined into the literature. The TFVR equals to the PLT-HAR but the browser-based ATF values are on average higher for more than 25% of the measurements, which is most of the time due to JavaScript triggering the re-construction of the DOM and CSSOM tree and increasing the overall processing time. On the other hand, the ATF calculation through videos is smaller for less than 54% of the measurements and conversely more due to the video capture quality and pixels analysis phase which needs more computing power.

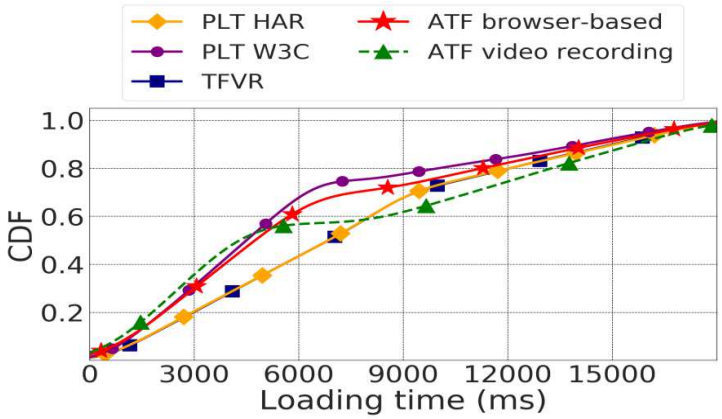


Fig. 34: Loading time when main web page is in Europe and 100% visible

Takeaways: When comparing different metrics offering the time to load the visible portion of a web page versus the final rendering output of the visible portion of the web page, less than 1% of pixels are missing for the TFVR, 7.3% for the TTI and 10.6% for the browser-based ATF. When calculating the ATF through video recording, we have noticed that an end-user device processor usage increases by 8% on average and memory usage by 9%. When the visible portion of a web page is 100%, the TFVR equals to the PLT-HAR but the browser-based ATF values can be higher due to JavaScript processing not captured. The ATF calculation through videos might also be smaller if the video recording is of bad quality and thus pixels being wrongly compared among them.

3.3 Conclusion

The PLT-W3C as implemented into web browsers does not offer accurate timings since it uses the Resource Timing API which cannot retrieve the corresponding request and download time of objects if a distant web server does not allow it. With the introduction of PWAs, scripts run permanently in the end-user's web browser to change the display or download additional objects upon end-users' action. The download of these additional objects is not captured by the PLT-W3C. End-users generally concentrate on the visible portion of websites at first glance and devices' screens come in different sizes. The PLT measures the time to load the entire web page, visible or not and is not adapted to measure finely end-users' perceived QoE.

The process of calculating the ATF from videos is time-consuming where human interaction in the whole process is compulsory. High definition videos increase the accuracy of the pixels comparison but reduce the end-user's device processing and storage capacity. Furthermore determining the needed snapshots from the video is non-trivial. The pixels comparison task is also time-consuming as it depends on the number of snapshots. With the proliferation of advertisements in the visible portion of the web page, another question arises

from this measurement technique since advertisements change regularly. It is thus hard to define when the video recording should be stopped and if pixels from advertisings should be taken into account. The browser-based ATF technique can expose incorrect timings since it uses the Resource Timing API information where networking times are not always exposed. The RUM-SpeedIndex algorithm only takes into account images rendered into the visible portion of the web page. The TTI is only implemented in the *Google-Chrome* web browser and can be calculated if and only if the network state is idle during 5 seconds. Web pages are mostly dynamic nowadays and network states are rarely idle, which prevents TTI to be exposed.

The Time for Full Visual Rendering is a browser-based measurement technique which is browser-type and browser-version independent. The TFVR is calculated in real-time with a mean extra computational time of 0.156 seconds by making use of the web browser's offered networking logs. The TFVR takes into account new Web technologies and captures events triggered by JavaScript or Progressive Web Applications. In addition to the time to load the visible portion of a web page, the TFVR also exposes the step by step processing phases for every downloaded object and identifies through which Internet protocol they are downloaded as well as the geographic location of the corresponding web servers.

When comparing different metrics offering the time to load the visible portion of a web page versus the final rendering output of the visible portion of the web page, less than 1% of pixels are missing for the TFVR, 7.3% for the TTI and 10.6% for the browser-based ATF. When calculating the ATF through video recording, we have noticed that an end-user device processor usage increases by 8% on average and memory usage by 9%. When the visible portion of a web page is 100%, the TFVR equals to the PLT-HAR but the browser-based ATF values can be higher due to JavaScript processing not captured. The ATF calculation through videos might also be smaller if the video recording if of bad quality and thus pixels being wrongly compared among them.

The TFVR has proven to provide fine-grained loading times taking into account new web technologies such as Progressive Web Applications and side effects of JavaScript. The TFVR is calculated in real-time and no external tool which can consume end-users' computing power is needed.

Chapter 4

Characterizing and Quantifying Web Browsing

Contents

In this chapter we pay particular attention to the Web browsing eco-system. As discussed in previous chapters, we have developed a new tool meant to measure web browsing quality being embarked with new web metrics and measurements have been performed over 2.5 years. Thanks to these measurements and their statistical analysis, we firstly characterize the Web browsing eco-system to better understand which and how different actors are implicated into the delivery process. Secondly, we expose how different factors can contribute in increasing or decreasing the perceived Quality of Experience.

4.1 Web browsing delivery

When browsing websites, several factors are implicated such as the web browser used, different network access bandwidths or ad blockers to visit web pages which can be static or dynamic, being composed of content in different flavors. These contents are delivered by different types of web servers such as regular web servers or Content Delivery Networks (CDNs) through different Internet protocols.

We present in this section the analysis of 244 Million distinct measurements performed over the Top 10,000 Alexa websites. Although our measurements sum up to more

than 18 Trillion, we pay particular attention to measurements performed via the web browsers Chrome v.63 and v.68 and Firefox v.56 and v.62 from Web View probes in Europe. These web browsers have been released in 2017 and 2018 and have been used during more than 2 years to perform measurements. Apart from new functionalities added to recent *Google-Chrome* or *Mozilla-Firefox* web browsers, they make use of the same processing engine implemented for example in Chrome v.68 and Firefox v.67 respectively.

From these 244 Million distinct measurements representing 9597 distinct websites, 403 distinct websites were discarded due to the small amount of web browsing results (these websites were unavailable, blocked as per our geographic location or not responding in less than 18 seconds). Although our probes are located in Europe (EU) and Africa (AF), we focus on the measurements performed by EU Web View probes.

Our dataset then includes measurements for websites having their main web page estimated to be in North America (NA) for 52.23%, in Europe (EU) for 28.44%, in Asia (AS) for 16.22%, and in South America (SA) for 1.10%. To have a clearer view of the Web browsing delivery, we will assess:

1. The amount and types of downloaded objects,
2. The types of web servers delivering content,
3. The Internet protocol through which content is delivered to web browsers,

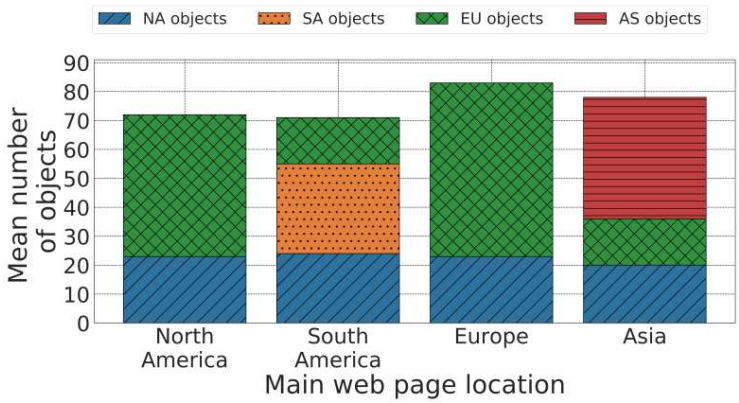
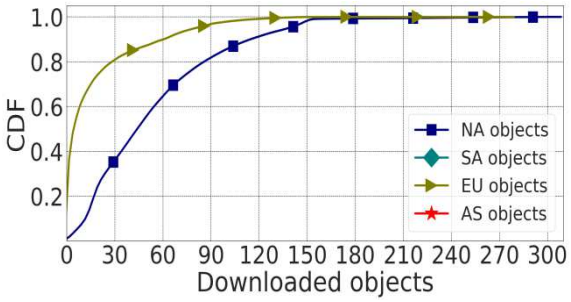


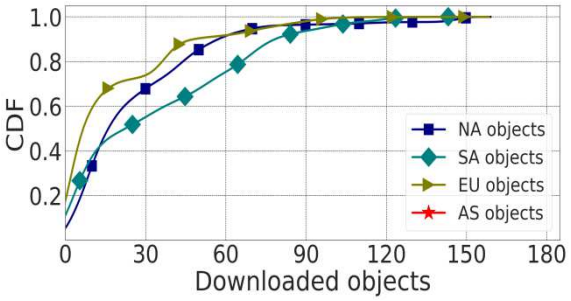
Fig. 35: Downloaded content from different continents

4.1.1 Amount and Types of downloaded content

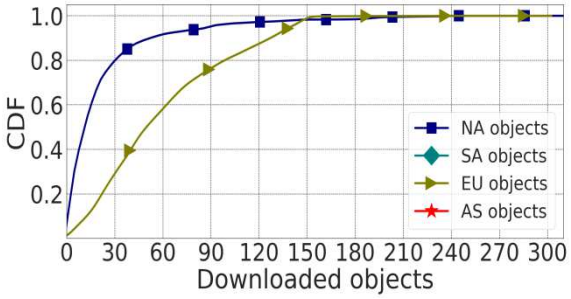
Web pages are more complex nowadays and we wanted to evaluate their evolution, compared to what it was few years ago [11] [12] [64]. Fig. 35 depicts the average distribution of objects when browsing the Top 10,000 Alexa websites and shows that following a main web page location, content is delivered from different continents to end-users. The mean distributions help in profiling the overall location of the content servers for a European end-user. Fig. 36 provides the complete distribution of downloaded objects for the 4 continents where the main home page is estimated. We can observe that for a European end-user, many objects are downloaded from Europe, which is due to Content Delivery Networks. For main web pages located in Asia, we can detect that for a European end-user, a bigger amount of objects are downloaded from Asia, which could impact the quality, because of the network delay. Last but not the least, we can also identify that following the estimated location of a main web page, different amounts of objects will be downloaded, i.e a maximum of 300 for main web pages in NA or EU and 185 for SA or AS. Since 2014, the amount of downloaded resources composing a web page has increased by 17% on average for the Alexa websites ranked 1-2000 and by 31% on average for websites ranked 5000-10000.



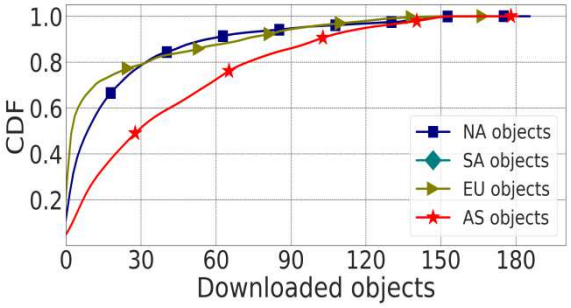
(a) Main web page in NA



(b) Main web page in SA



(c) Main web page in EU



(d) Main web page in AS

Fig. 36: Aggregated distribution of objects from different continents

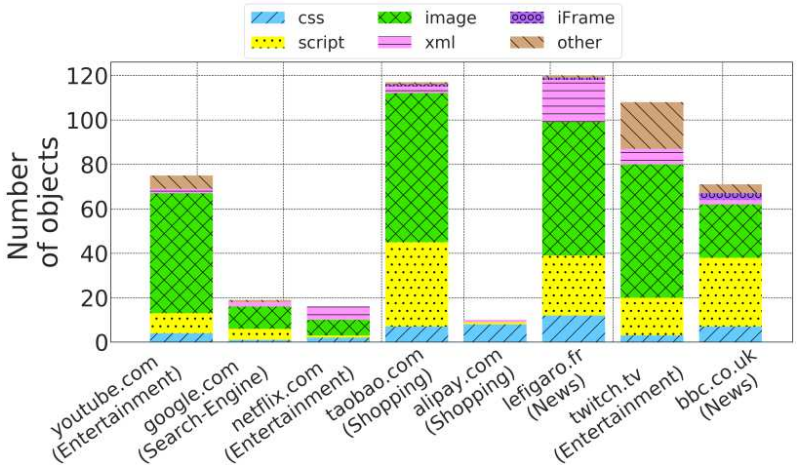
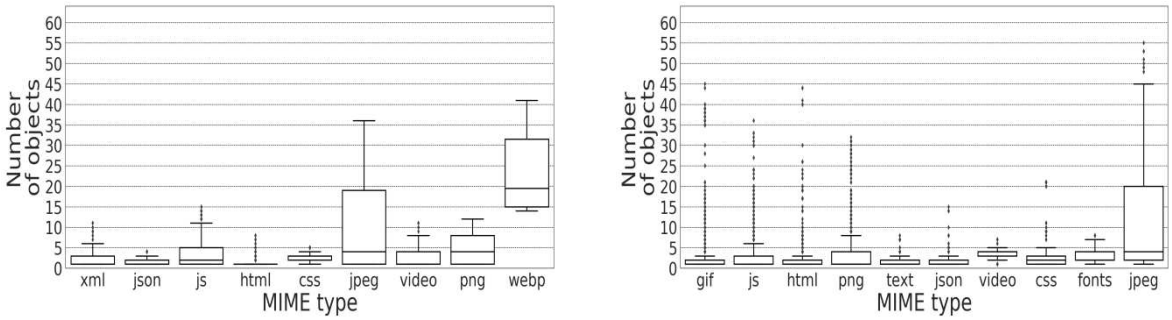


Fig. 37: MIME-type of downloaded objects

Following the question about what are these objects, Fig. 37 represents the breakdown of the downloaded objects by content MIME types of 8 random websites belonging to different categories as referenced by Alexa[37]. While images occupy most of the time the highest distribution type of downloaded objects, on average web pages (except *Search-Engines* category) are composed of 4 *css*, 5 scripts, 16 images and 2 *xml*. We identified that the average number of scripts and images has increased by 53% over the last 15 years from past studies [51] and by 7% from recent studies [11]. Furthermore when paying particular attention to the different types of images, when using a *Google-Chrome* web browser and visiting a *Google* website, on average 80% of images are now in WebP format

[37]https://www.alexa.com/topsites/category

(As from the last quarter of 2019, the Firefox v.70 web browser can process images in WebP format). Compared to studies conducted in the timeframe 2011 – 2014, *Flash* usage has been reduced by 61% as Adobe will remove support for Flash in 2020. We also noticed an increasing PWA (Progressive Web Apps) usage of 6% between July 2018 and January 2019. On a wider scope, based on the Alexa websites category listings available, 3921 websites were assessed. The *News* websites download on average 19.81% more resources, *Kids and Teens* websites have a significant greater fraction of Flash objects and *Shopping* websites make greater usage of JavaScript.



(a) Distribution for *Entertainment* category (b) Distribution for *Shopping* category

Fig. 38: MIME-types of objects downloaded per category

When assessing the overall distribution of resources aggregated by MIME type, the Fig. 38 represents two different Alexa-referenced categories where a large variety of objects are incorporated in the corresponding web pages. We selected these 2 categories, since they represent a major part of the popular websites. The Fig. 38(a) shows that *Entertainment* websites make greater usage of images in *WebP*, *PNG* or *JPG* format, followed by JavaScript and videos which can be further decomposed in different formats like *WebM*, *Mp4* or *Mp2t*. The *Shopping* category depicted by Fig. 38(b) also makes use of a high number of images in different formats which greatly change all along the day thanks to JavaScript.

4.1.2 Types of web servers implicated

We also paid attention to which extent, when an end-user browses to a specific website from Europe, the types of web servers from which the needed objects are downloaded. For this, we analyzed if the contents are downloaded from the same authoritative DNS name server of the main web page. Domains delivering contents and having the same authoritative DNS name server as the main web page are entitled *Same-Origin* domains and conversely *Non-Origin* domains.

From our measurements, when a main web page domain is in North America or Europe, irrespective of the preferred Internet protocol, contents are served on average by 2 Same-Origin domains and 7 *Non-Origin* domains and when the main web page is in Asia or South America, contents are delivered on average by 3 Same-Origin domains and 13 *Non-Origin* domains. Those *Non-Origin* domains represent specific services involved in the web page composition e.g *Google* services, advertisements or analytics, but also CDN nodes providing contents on behalf of the origin web servers. The average number of domains is not huge, but we can have some websites where several domains are involved in the delivery.

As an example of the complexity of web browsing delivery, Fig. 39 points out the web servers delivering content when browsing the website *lemonde.fr*, where the main web page is located in Europe and belongs to the *News* category. Although this website is a French online newspaper, upon embedded advertisements or web services, content will be downloaded from Europe, North America, Asia and Africa. The main web page, as well as the 13 *Same-Origin*

domains, is hosted in Europe by *Fastly* serving 61.36% of the overall number of objects, while *Akamai* and *Amazon* serve 11.36% of the objects. The other content servers regroup 9 different *Non-Origin* domain web servers. Our website Web View[38] proposes a graphical representation of this kind of analysis.

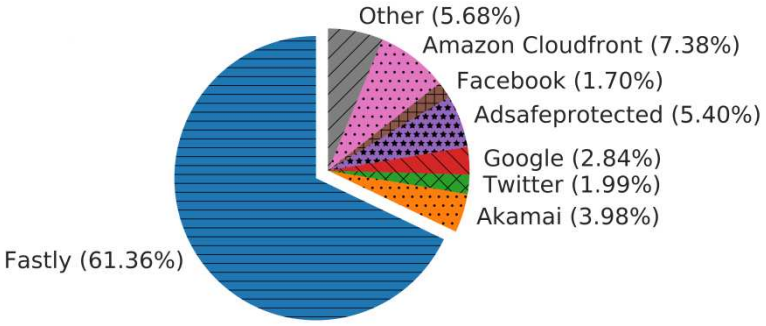


Fig. 39: Web servers serving content for the homepage *lemonde.fr*

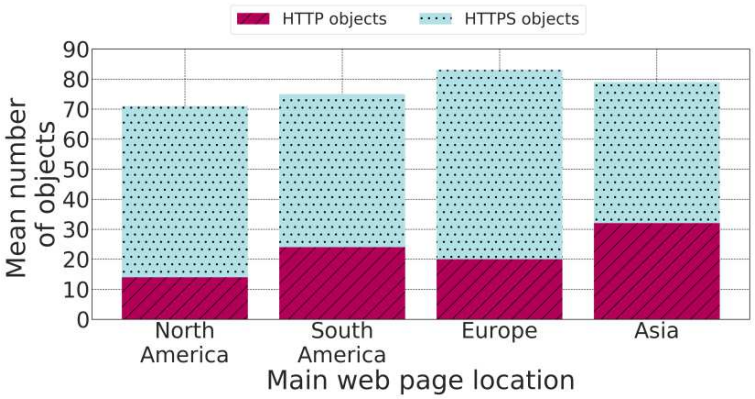


Fig. 40: Objects downloaded in HTTP or HTTPS

Compared to the previous studies [3] [124] [4] encryption has been largely adopted to preserve privacy. Indeed, web browsers nowadays favor it by adding by default *https://* when a user requests a web page. Furthermore, since *Google* marked non-HTTPS websites as insecure in its Chrome browser in July 2018, HTTPS adoption has increased. However, from the Top 10,000 Alexa websites, only 36.02% of the websites deliver their contents in full HTTPS (100% HTTPS) and still 0.28% of the websites deliver their contents in full HTTP (100% HTTP, no resource composing the web page is delivered by an HTTPS server). In between, we have websites composed of resources received with both HTTPS and HTTP. As per Fig. 40, on average, when the main web page is located in North America and Europe, a greater fraction of content is delivered into a secured way, compared to websites' main web page located in South America and Asia.

4.1.3 Internet protocols adoption

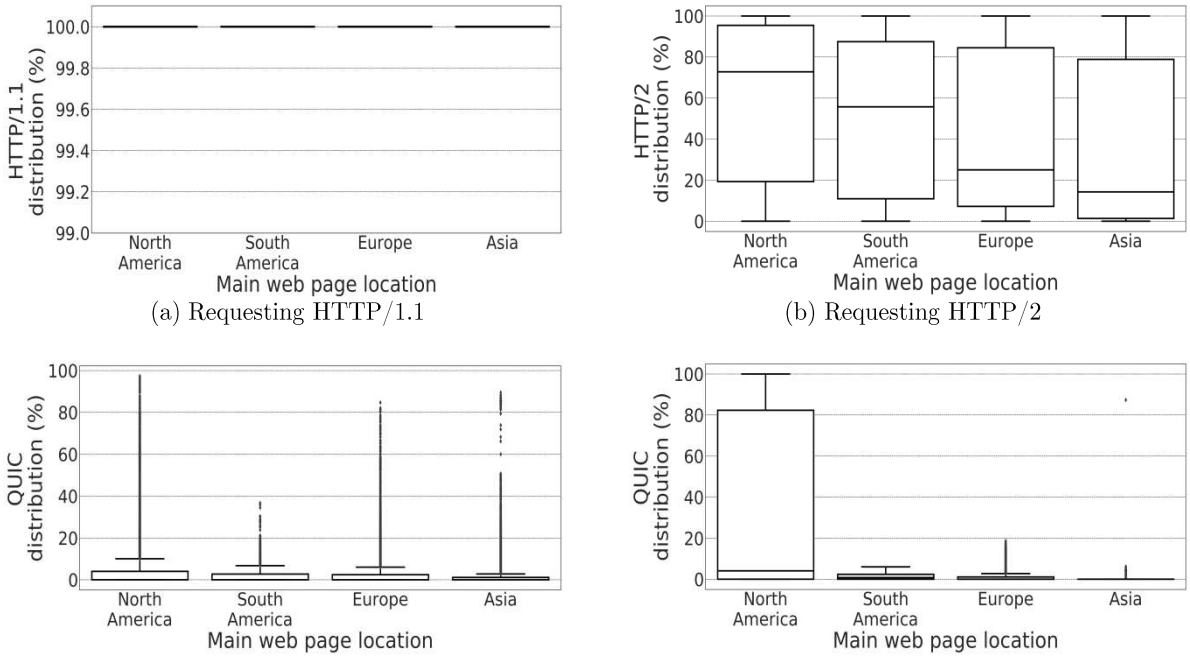
Since the previous published papers [20] [21] [38] [40] [19] [33] analyzing Web browsing, new Internet Protocols namely HTTP/2 and QUIC have been promoted. We thus evaluated

[38] https://webview.orange.com/d/UyIIcrUmz/

the adoption of these protocols by web servers by performing measurements explicitly requesting these protocols and analyzed if the remote web servers reply using them or if they fall back to another one. Fig. 41 and Fig. 42 show the results of those measurements, in terms of protocol distribution.

First and very logically, we note that when requesting web pages in HTTP/1.1, all the servers reply in HTTP/1.1 as shown in Fig. 41(a). When performing measurements and requesting HTTP/2, objects are downloaded in strict HTTP/2 (100% H2) only for 11.82% of the Top 10,000 Alexa websites. The other 88.18% of websites reply with a mixed distribution of HTTP/1.1, HTTP/2 and Server Push. We can then see that although standardized in 2015, HTTP/2 is not yet widely deployed and that it is not equivalent worldwide, more used for web servers in NA or SA, whereas in Europe and Asia, HTTP/1.1 is still prevalent as shown in Fig. 42(b). When analyzing the adoption of HTTP/2 by web servers serving content for the Top 10,000 Alexa websites, only 4% more objects are delivered in HTTP/2 through time (63% in March 2019 versus 59% in March 2018).

When requesting QUIC and using a *Chrome* web browser, we see that QUIC is not deployed by many servers, but mainly *Google* ones. On average 97% of the different QUIC-enabled web servers are *Google* web servers replying in QUIC. We detected that none of these websites reply in full QUIC. QUIC responses are highest when the main web page is in North America (in particular from *Google* servers). QUIC is natively used jointly with HTTP/2 for the first request, but offers a QUIC 0-RTT connection when connecting to an already known website. We then evaluate if the distribution is different with the QUIC Repeat mode. In this configuration only 7.21% of the websites fully reply in QUIC.



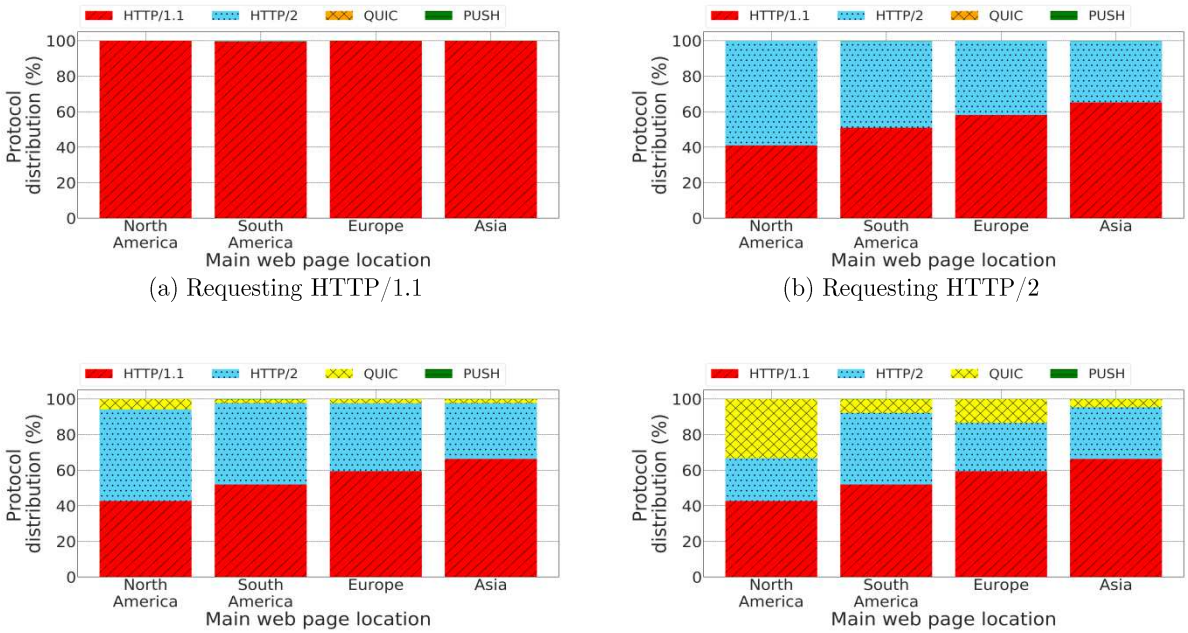
(a) Requesting HTTP/1.1

(b) Requesting HTTP/2

(c) Requesting QUIC

(d) Requesting QUIC Repeat

Fig. 41: Received protocol distribution upon request



(a) Requesting HTTP/1.1

(b) Requesting HTTP/2

(c) Requesting QUIC

(d) Requesting QUIC Repeat

Fig. 42: Average received protocol distribution

Takeaways: Since 2014, the amount of downloaded resources has increased on average by 17% for Alexa websites ranked 1-2000 and by 31% for websites ranked 5000-10000. While images occupy most of the time the highest distribution type of downloaded objects, on average web pages are composed of 4 *css*, 5 scripts, 16 images and 2 *xml*. We identified that the average number of scripts and images has increased by 53% over the last 15 years from past studies [51] and by 7% from recent studies [11]. When the main web page is located in North America and Europe, a greater fraction of content is delivered into a secured way, compared to websites' main web page located in South America and Asia. HTTP/1.1 is still widely used by web servers and in particular when downloading objects for websites whose main web page is in Asia. HTTP/2, although standardized in 2015, is deployed at a low pace and even if an end-user makes use of the latest updated web browser, content is downloaded in both HTTP/1.1 and HTTP/2. When comparing measurements in March 2018 versus March 2019 for the Top 10,000 Alexa websites, HTTP/2 protocol distribution has increased by only 4%. No website replies in 100% QUIC which is mainly deployed on *Google* web servers. Since May 2019 where the QUIC standardization process has been started at the IETF, major CDNs have started implementing QUIC in their web servers.

4.2 Factors impacting web browsing quality

An indicator of end-users' perceived quality when performing web browsing is the time needed to load a web page (entirely or certain parts of it). This section points out the different factors impacting these loading times. We firstly assess the impact of a remote web page structure on perceived quality, followed by elements implicated into the content delivery as well as their impact on loading times and finally end-users' environment such as the network access or time of the day when performing web browsing

The 25^{th} percentile and 75^{th} percentile is denoted by Q_1 and Q_3, the inter-quartile range $(Q_3 - Q_1)$ is denoted by IQ, and to identify the extreme values in the tails of the distribution, the upper inner fence is calculated as $(Q_3 + (1.5 * IQ)$ and upper outer fence as $(Q_3 + (3 * IQ)$. Mild outliers are timings beyond inner fences and extreme outliers beyond outer fences.

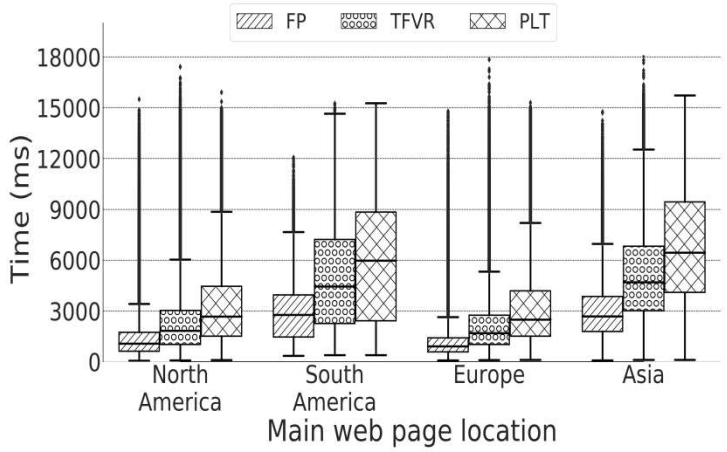


Fig. 43: Overall loading times following the main web page location

4.2.1 Web pages' structures and categories

We firstly pay particular attention to the structure of the web pages, i.e upon an end-user geographic location how the main web page location can impact quality, following the category to which the website belongs how the profiled content can increase loading times and how the visible portion of a web page impacts its corresponding loading time.

Impact of main web page location

The Fig. 43 depicts the overall web pages' loading times, grouped by estimated continents location of the home page. The overall centered (median) time follows the rule $FP \leq TFVR \leq PLT$. Being in Europe when performing those measurements, we noticed that the overall loading times for a European end-user are smaller for the websites located in North America and Europe than for those located in South America and Asia. This is mainly thanks to content delivered closest to end-users visiting web pages in Europe and North America. Loading times are higher due to high number of domains (mainly *Non-origin*), high amount of downloaded objects and objects served over HTTP/1.1. The extreme outliers are for objects downloaded from South America and Asia. When the main web page is located in South America and Asia, the observed times are higher in general but the number of outliers reduced since more resources are downloaded from the same continent of the main web page domain and the upper limit closer to the defined timeout of 18 seconds for each measurement.

Impact of websites' referenced categories

Websites are all different among them, being composed of a wide number and types of objects. Following their content, they can be referenced as belonging to different types of categories, e.g Alexa, QuantCast[39] or Web Filter[40]. In our study, we make use of Alexa referenced category listing where websites are classified upon 17 different categories. The Fig. 44 shows the overall observed PLT for different categories. Websites belonging to the *Computers*, *Reference* and *Shopping* category have loading times being close among them (compared to websites belonging to the *Recreation*, *Business* and *News* category) mainly due to objects' MIME types composing the webpage.

We can observe that the CDF is similar for many categories except for 2 of them (*Adult* and *Education*) where we have a considerable gap in loading times. While these two categories download on average the same types of objects, mainly images and videos, the main difference which results in a higher observed PLT is that objects for *Adult* categories are mainly downloaded in HTTP/2 protocol (HTTP/1.1 for *Education* category) and that

[39] https://help.quantcast.com/hc/en-us/articles/115014006128-Interest-Category-Definitions
[40] https://fortiguard.com/webfilter/categories

images and videos for the *Education* category are bigger in size. We can then see that the category can play a role in the web browsing quality, but it is limited and less than the sizes of downloaded objects and the protocol through which they are downloaded which contribute most of the time to a higher PLT value.

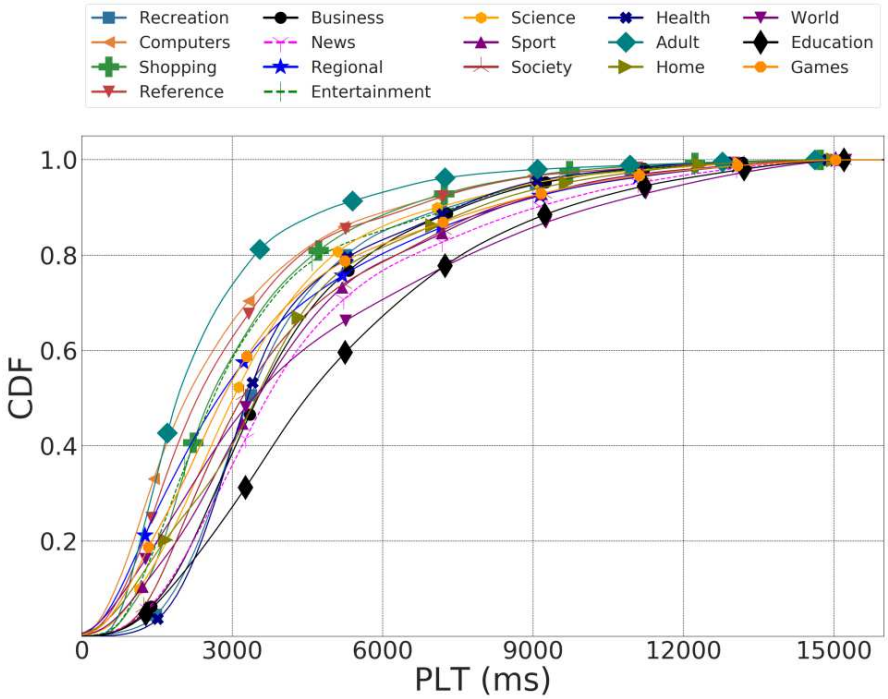
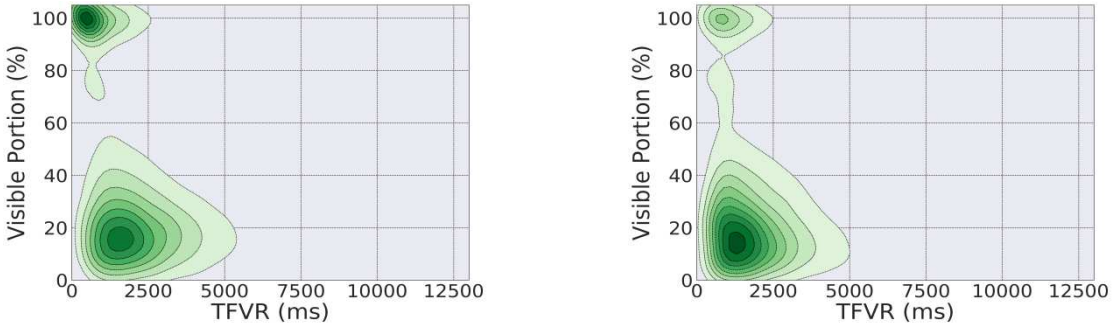


Fig. 44: PLT for different websites' categories



(a) Main web page in NA

(b) Main web page in EU

(c) Main web page in SA

(d) Main web page in AS

Fig. 45: Impact of the visible portion on the TFVR

Impact of the visible portion on loading time

Our measurements have been performed following different web browsers' window sizes and the Fig. 45 (Heat map: Dark green represents higher concentration of values and light green lower concentration of values together with the corresponding TFVR) shows that depending on the main web page continent location and corresponding web browser window size, an end-user might have a larger overview of the web page without scrolling. The default logical assumption would be that the time to load λ % of a web page would be proportional. Fig. 45(d) depicts websites whose main web page is in Asia, where we can see that these web pages have a visible portion massively ranging from 13% to 17% with loading times between 2700 ms and 5000 ms (apart from a limited number of web pages having a visible portion of 100% and TFVR between 1815 ms and 2104 ms). The loading time of the visible portion increases according to the visible surface area if and only if the visible portion of the web page is less or equal to 99%.

Takeaways: From the Top 10,000 Alexa websites, a greater portion of main web pages are located in North America and Europe where the use of Continent Delivery Networks is higher. This contributes to lower perceived loading times compared to web pages in South America or Asia. Websites are all different among them but when regrouped as per different categories, they share most of the time same types of content. Quality differs between different categories mainly due to the types of content and Internet protocol to deliver them. Websites referenced as *Search-Engines* are most of the time 100% visible at first glance with small loading time values. For the other categories, the scroll length of web pages all differ and the loading time increases accordingly to the visible surface area if and only if the visible portion is strictly less than 100%.

4.2.2 Content delivery factors

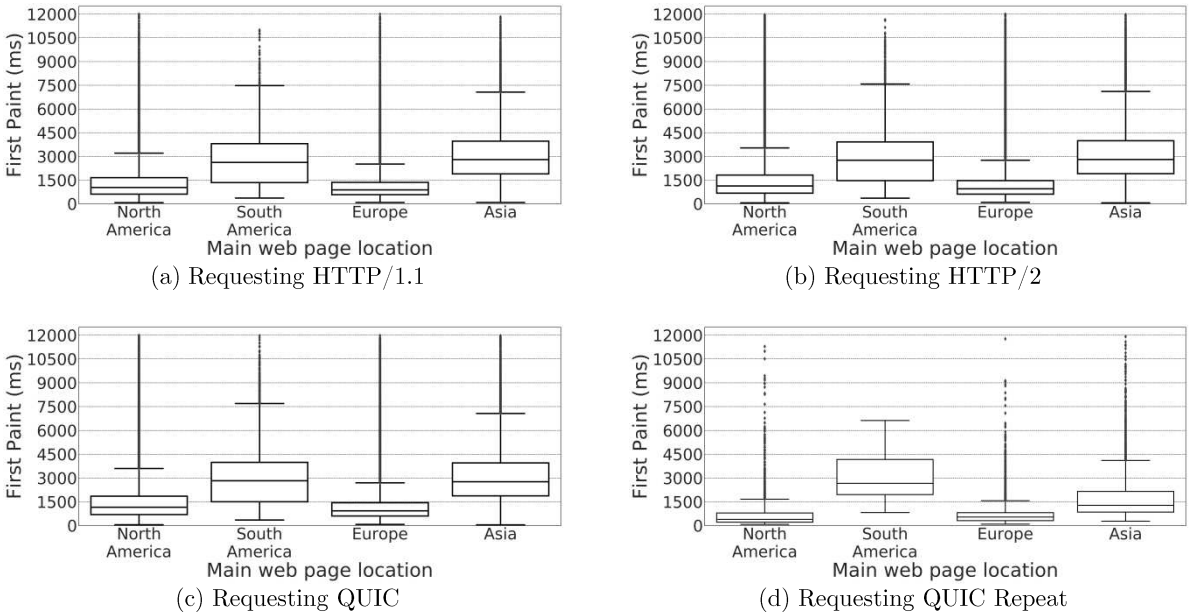
Secondly we go in depth on the factors which contribute to content delivery and how they can impact loading times, i.e the Internet protocol distribution received when requesting a specific one, the number of domains as well as CDNs serving content at different times of the day.

Impact of the requested protocol

We present in this sub-section the impact of Internet Protocols on end-users' perceived quality following different loading times such as the FP, TFVR or PLT.

The First Paint (FP)

When the requested protocol is HTTP/1.1, as shown in Fig. 46(a) the median FP value is close for websites having their main web page in North America and Europe. For main web pages in South America or Asia, the median value is greater since a higher fraction of content is not downloaded from Europe. This fact induces network delay and impacts the FP value. When requesting HTTP/2, depicted in Fig. 46(b), the observed FP time is close to FP time when requesting HTTP/1.1. The reason is that on average 2 objects are downloaded in the FP lapse of time and all the benefits of the HTTP/2 protocol (multiplexing, header compression, server push function) cannot happen. When requesting QUIC, the observed FP time is very close to HTTP/2 measurements. The reason is that the first request is always sent in HTTP/2 by the web browser when a remote web server is triggered for the first time and since the number of objects downloaded is mean, 99.76% of requested QUIC protocol measurements receive HTTP/2 replies. When requesting QUIC Repeat, as depicted in Fig. 46(d), objects are downloaded in 0-RTT from UDP-enabled web servers and in 1-RTT from TCP-enabled web servers which contributes to a decrease in the FP value.



(a) Requesting HTTP/1.1

(b) Requesting HTTP/2

(c) Requesting QUIC

(d) Requesting QUIC Repeat

Fig. 46: Impact of requested Internet protocol on the First Paint

The Time for Full Visual Rendering (TFVR)

The TFVR represents the loading time of the visible portion of the web page and is proportionally linked to the end-user's browser window size. Fig. 47 depicts the different visible portion of websites following different viewports. In general, main web pages located in North America offer a higher visible surface area to end-users since a large portion of these websites mainly represent *Search-Engines*. When requesting HTTP/1.1, for an end-user having a browser size of 1440 x 900, and main web page in EU, less than 50% of the measurements have a TFVR less than 1646 ms and for Asia less than 4800 ms. The time to load the visible portion of web pages in Asia is greater mainly due to a higher number of downloaded objects needed to render the visible portion, contents served by a greater number of domains and due to the geographic position of the probe being in Europe which induces network delay, on average the TFVR is higher. When requesting HTTP/2, in Fig. 48(b), we can see that TFVR values are greater by 7.37% than HTTP/1.1 measurements. This is mainly due to Server Push where 1.21% more objects are downloaded. When requesting QUIC, irrespective of the end-user's browser window size or main web page location, the observed timings are close to HTTP/2 measurements since replies from web servers are mainly done over HTTP/2. When requesting QUIC Repeat, the overall TFVR values are decreased mainly due to more replies delivered avec the QUIC protocol.

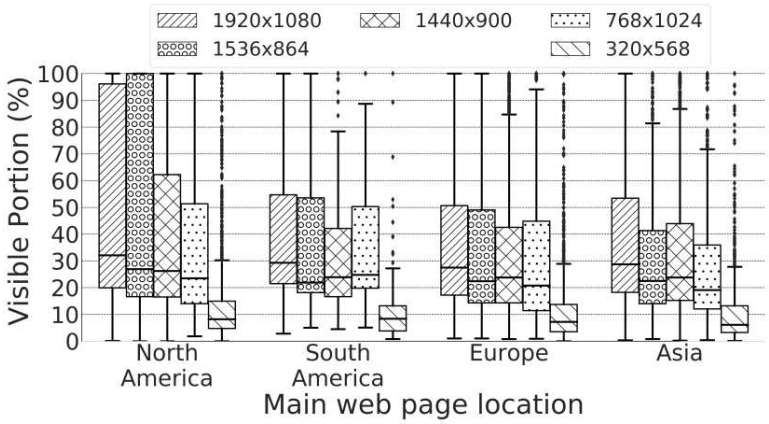
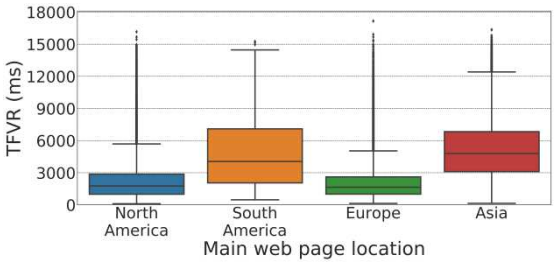
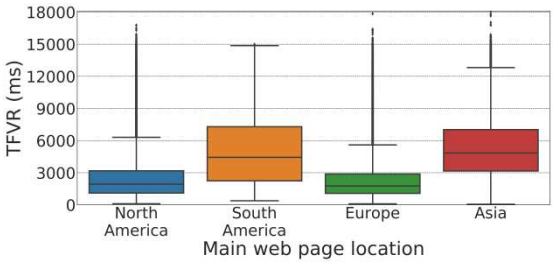


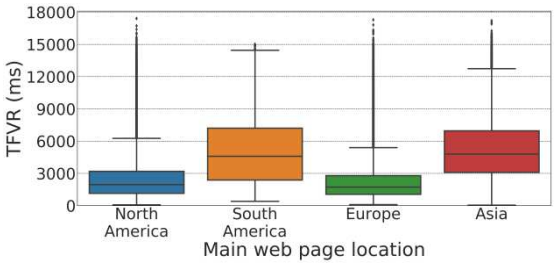
Fig. 47: Visible portion of websites upon different viewports



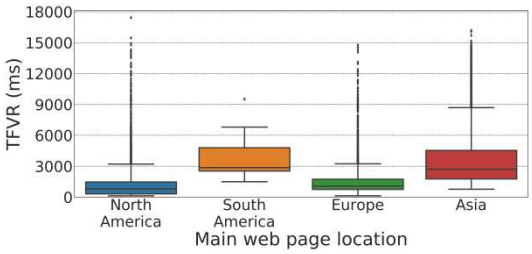
(a) Requesting HTTP/1.1



(b) Requesting HTTP/2

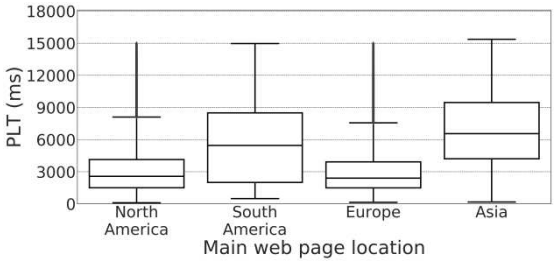


(c) Requesting QUIC

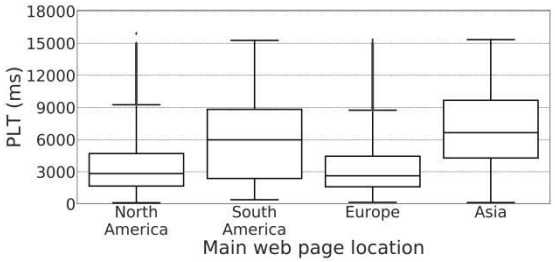


(d) Requesting QUIC Repeat

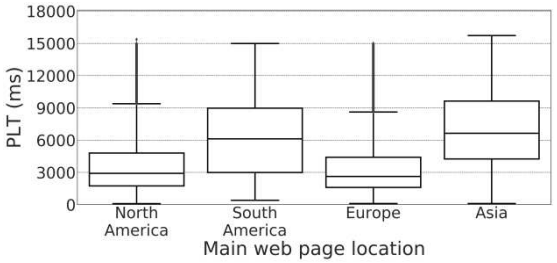
Fig. 48: Impact of requested Internet protocol on the TFVR



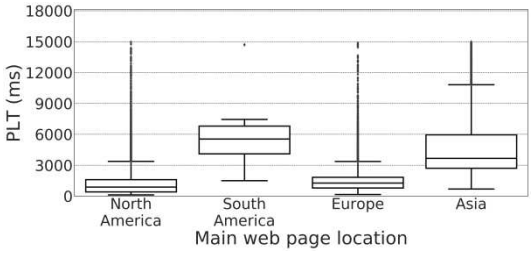
(a) Requesting HTTP/1.1



(b) Requesting HTTP/2



(c) Requesting QUIC



(d) Requesting QUIC Repeat

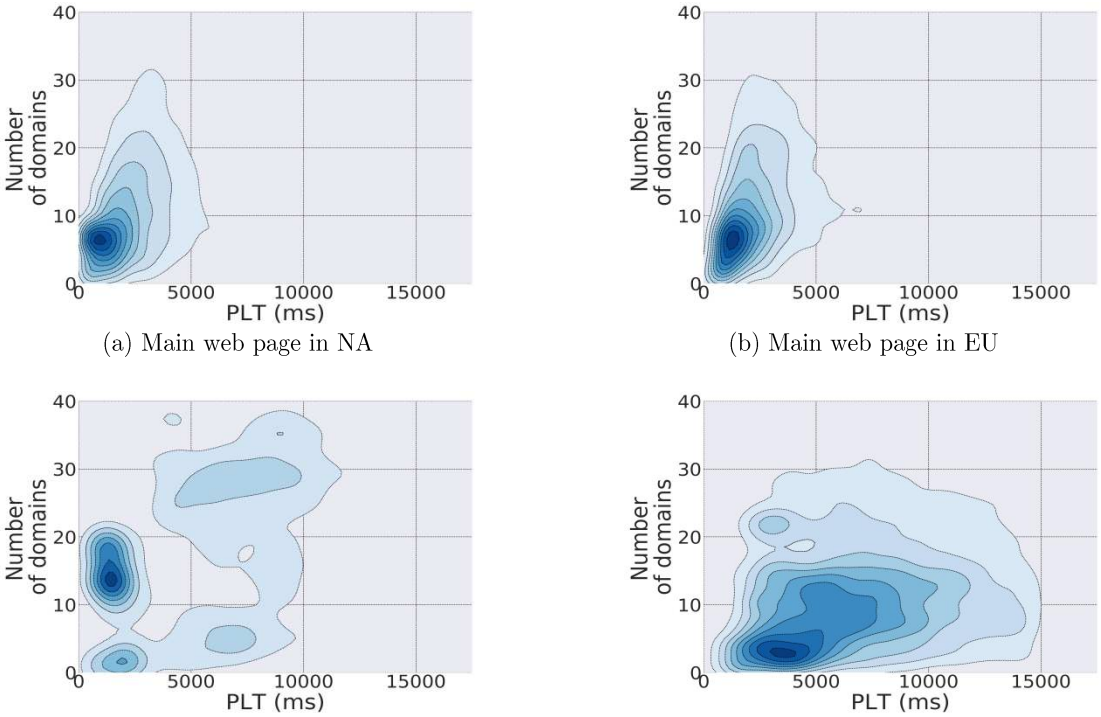
Fig. 49: Impact of requested Internet protocol on the PLT

The Page Load Time (PLT-HAR)

The Page Load Time is the needed time to load an entire web page (visible and non-visible parts). Through Fig. 49(a), we can see that when requesting HTTP/1.1 and the main web page is in NA, less than 50% of the measurements have a PLT-HAR less than 2575 ms. The parameters leading to a PLT-HAR less than 1505ms (25^{th}percentile) are small number of domains delivering contents and an average number of 21 resources downloaded. When the main web page is in Asia, we can observe higher loading times since on average a higher number of domains are involved into the content delivery process. Although HTTP/1.1 favors pipelining, only six parallel TCP connections can be performed, thus increasing proportionally blocking and waiting time.

When requesting HTTP/2, as shown in Fig. 49(b), the HTTP/1.1 Internet protocol is omnipresent mainly due to services embedded in the web page which yields a loading time

nearly identical to HTTP/1.1 measurements. When requesting QUIC and the main web page is not in North America, the end-user's perceived quality as compared to HTTP/2 stays the same due to the small amounts of QUIC-enabled content servers. When requesting QUIC Repeat, every observed measurement is performed in 0-RTT UDP or 1-RTT TCP and a decrease up to 66.54% in loading times happen compared to HTTP/2 measurements. Main web pages located in North America are served on average at a rate of 38% of QUIC, and thus being the most QUIC-friendly. For web pages located in Asia, a decrease of 37.82% of the PLT is observed but only thanks to the 1-RTT TCP.



(a) Main web page in NA

(b) Main web page in EU

(c) Main web page in SA

(d) Main web page in AS

Fig. 50: Number of domains serving content to a European end-user

Impact of the number of domains serving content

When performing web browsing, content is downloaded from various domains (as discussed in section 4.1.2, from either *Same-Origin* or *Non-Origin* domains). The number of domains contacted increases proportionally to the DNS lookup times and thus the overall loading times for the TFVR or PLT. When a web page is located in North America or Europe, on average, resources are downloaded from 9 domains, with an overall DNS time of 316 ms and when the main web page is in South America, resources are downloaded on average from 16 domains, with an overall DNS time of 1715 ms. In general websites being served by large number of domains belong to the *News* and *Shopping* category.

Fig. 50 illustrates the overall tendency regarding the number of domains through which resources are downloaded when requesting HTTP/2 and the corresponding loading time. This is represented as a heat map where dark blue represents higher concentration of values and light blue lower concentration of values together with the corresponding Page Load Time. When a web page is located in North America or Europe, the PLT is good and time increases with the number of domains from which the resources are downloaded. When the main web page is in Asia, the PLT values increase mainly due to the number of domains serving content mainly over HTTP/1.1. Fig. 50 allows to better profile the number of domains contacted as per the main web page location which can increase the PLT.

Home page location	Mean downloaded content (%)	Top 5 CDN providers	Mean distribution (%)			
			Content	HTTP/1.1	HTTP/2	QUIC
North America	29.14	Google	24.2	3.6	70.2	26.2
		Amazon	19.9	30.9	69.1	-
		Akamai	17.8	48.5	51.5	-
		Fastly	9.8	24.1	75.9	-
		Verizon	9.4	20.1	79.2	0.7
Europe	24.52	Akamai	27.9	45.4	54.6	-
		Google	26.3	3.5	69.7	26.8
		Verizon	13.01	15.17	84.81	0.02
		Amazon	11.3	32.07	67.93	-
		Fastly	7.4	21.09	78.91	-
South America	25.72	Amazon	41	1.2	98.8	-
		Google	24.4	0.9	65.8	33.3
		Verizon	12.8	20.1	79.9	-
		Akamai	9.6	84.4	15.6	-
		Fastly	7.05	14.14	85.86	-
Asia	9.89	Amazon	25.8	45.9	54.1	-
		Google	20.8	3.2	32.1	34.7
		Akamai	19.2	59.7	40.3	-
		Verizon	7.8	32.8	63.95	3.25
		China Cdn	6.06	7.44	92.6	-

Table 11: Content delivered when requesting QUIC

Impact of Content Delivery Networks on loading times

When performing web browsing, content is downloaded to web browsers from different domains. These domains are hosted by web servers which can be of different types, i.e origin web servers, caches or CDNs as discussed in section 1.2.

Since different web servers from around the globe might deliver needed content and as discussed in section 4.1.3, they might not implement the latest Internet protocol. We analyzed [125] through which Internet protocol content is delivered by CDN providers and noticed that only 32 CDNs (out of 84 well-known CDNs) have adopted HTTP/2 and only 20 CDNs reply in QUIC. For those replying in QUIC, *Google* is the main actor (60.7%), *Akamai* delivers 5.3%, whereas the others provide less than 1% (*Verizon*, *Fastly*, *Level3*, etc). Since contents can (or not) be available in the CDN node, we performed additional tests with a *Repeat* mode, to compare with the *First* mode, which is the normal behavior. This *Repeat* mode means that we first get the home page of the website, clear all local objects' caches and request the same home page again. Since those resources have just been requested, they might have been cached in a CDN node and thus we expect a larger number of resources fetched from a CDN when requesting the home page the second time.

When requesting QUIC (with possible fallback to HTTP/2 and HTTP/1.1) for the *First* mode, the CDN providers deliver an average of 22.3% of the contents to end-users, illustrated in Table 11. When a main web page is in North America, South America and Europe, the CDN delivery is at an average of 26.5% but less than 10% for websites having their homepage located in Asia. Furthermore when a main web page is in Asia and the end-user in Europe, apart from *Google* or *Verizon*, HTTP/1.1 is privileged by the different web servers. When performing measurements in the *Repeat* mode as illustrated through Table 12, the overall average CDN distribution increases from 22.3% to 31.2%, irrespective of the main

web page location. With the *Repeat* mode, the average cache-hit rate is 98.6% which indicates that most of *cachable* objects are retrieved from CDN edge servers.

Home page location	**Mean downloaded content (%)**	**Top 5 CDN providers**	**Mean distribution (%)**			
			Content	**HTTP/1.1**	**HTTP/2**	**QUIC**
North America	42.28	Google	26.9	0.4	8.7	90.9
		Amazon	22.1	21.4	78.6	-
		Akamai	19.7	41.9	58.1	-
		Verizon	11.1	8.6	11.5	79.9
		Fastly	9.9	26.8	73.2	-
Europe	31.49	Google	28.1	0.3	4.2	95.5
		Akamai	29.8	32.1	67.9	-
		Verizon	14.02	4.3	12.5	83.2
		Amazon	12.1	21.4	78.6	-
		Fastly	7.6	20.09	79.91	-
South America	38.21	Amazon	45.9	0.2	96.7	3.1
		Google	24.4	0.4	7.1	92.5
		Akamai	10.75	61.3	38.7	-
		Verizon	14.3	7.2	8.9	83.9
		Cloudflare	7.9	-	100	-
Asia	12.71	Amazon	26.6	44.2	55.8	-
		Google	21.4	0.36	7.29	92.35
		Akamai	19.8	49.2	50.8	-
		Verizon	8.03	11.9	60.2	27.9
		China Cdn	6.24	100	-	-

Table 12: Content delivered when requesting QUIC Repeat

As discussed previously, we have seen that the Internet protocol through which objects are downloaded can decrease web pages' loading times. In order to analyze how CDNs can enhance web browsing quality, we measured the PLT for the *First* and *Repeat* mode, both with HTTP/2 and QUIC, at different times of the day. The *Peak Period* involves measurements performed between 16H and 21H CET and the *Off-Peak Period* involves measurements performed between 02H and 06H CET. When performing measurements in the *Off-Peak* period, between every measurement we intentionally clear the network's operator router cache and wait 11 minutes. From the HAR files, we have identified that the median time for objects being stored in a cache is 660 Kms (indicated by a Time-To-Live value). Fig. 51 illustrates the observed PLT of the Top 1,000 Alexa websites.

During *Peak Periods*, requesting HTTP/2 Repeat versus HTTP/2 increases CDN usage and decreases the PLT on average by 31.2%. Requesting QUIC Repeat versus QUIC increases CDN usage from 22.3% to 30.4% and decreases the PLT on average by 31.4%. The *Repeat* mode irrespective of the protocol yields merely the same decrease in loading times. The difference of CDN usage between the QUIC Repeat mode versus the HTTP/2 Repeat mode is 1.5% and decreases the average PLT by only 181.2 ms since only three CDNs involved in the Top 1,000 Alexa websites are QUIC-enabled.

For the *Off-Peak Periods*, when requesting HTTP/2 and QUIC, more resources need to be fetched from the origin servers since during this period, CDN caches are less populated with content. This results into a mean CDN usage of 15.3% when requesting HTTP/2 and 15.6% when requesting QUIC. The difference of CDN usage between the QUIC protocol and HTTP/2 is only 0.28% which is mainly due to the latency to reach websites located in Asia and South America. With the Repeat mode irrespective of the time of the day, the benefits of

the CDN delivery is clearly highlighted in the web browsing quality, leading to a non-negligible loading time reduction for all websites. If CDN providers deploy the new protocols in their CDN nodes, the end-users' QoE can even be more improved.

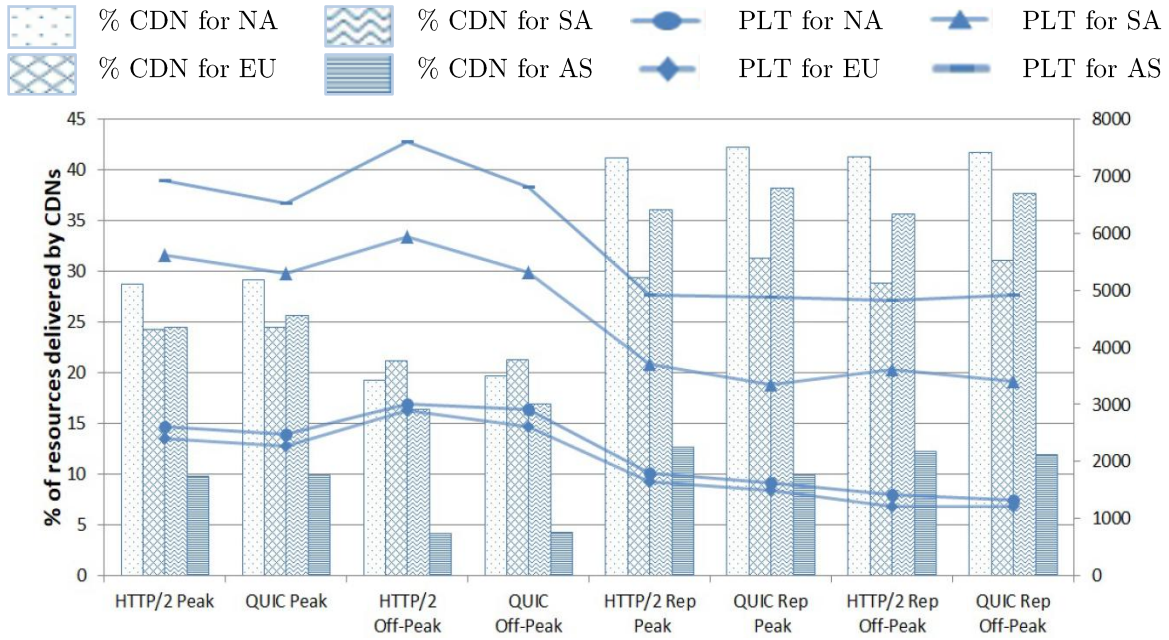


Fig. 51: CDN usage and impact on PLT

Takeaways: While HTTP/1.1, HTTP/2 and QUIC do not bring a fuliginous asset for the FP value, QUIC Repeat on the other hand brings a notable enhancement on the observed FP. When comparing QUIC Repeat versus HTTP/2, the TFVR loading time is reduced by 54.2% for a main web page in North America. The TFVR loading times increase when the end-user's browser visible surface area increases. HTTP/1.1 impacts the TFVR, HTTP/2 and QUIC bring along close loading times and QUIC Repeat enhances the most the TFVR thanks to 0-RTT UDP and 1-RTT TCP. The Page Load Time is only enhanced for the Repeat mode since web servers do not fully implement new Internet protocols; HTTP/1.1 is omnipresent. The greater is the number of domains serving content for a web page, the higher will the cumulative DNS time be and hence will increase the average TFVR and PLT. The FP is not impacted by the number of domains serving content since the rendering of the first pixel is most of the time thanks to content downloaded from the origin web server. CDNs are now widely used and some big CDN providers have a very large footprint on the world. For web browsing, our measurement campaign highlights its benefits, since in the *Repeat* mode, the Page Load Time can be reduced by an average of 43.1% when requesting HTTP/2 and 38.5% when requesting QUIC, leading to a better perceived quality by end-users. In short, about more than a quarter of the global contents are served by CDNs but in Asia, the use of well-known CDNs is yet limited (only 10%).

4.2.3 End-users' environment

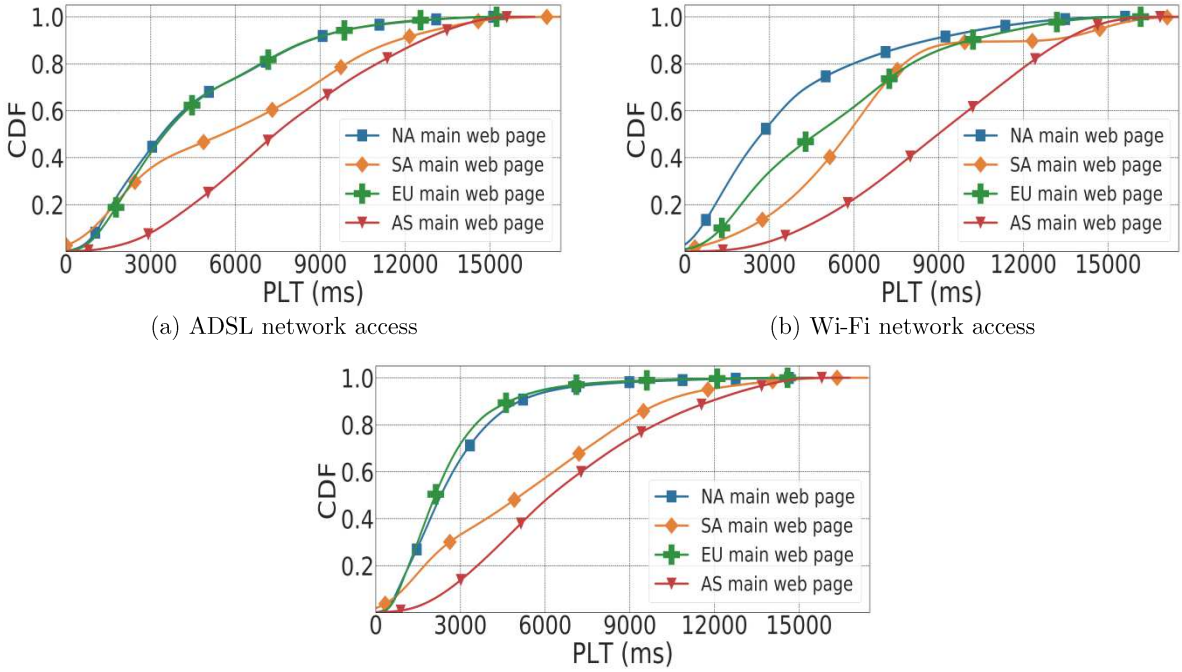
In this sub section, we pay attention to the end-users' environment which can impact loading times. We consider that the computing device, i.e the desktop or laptop has at least an Intel processor i5 in terms of processing power and a minimum of 8Go of RAM, with updated web browsers. We focus on the network access technology of end-users as well as the time of the day during web browsing to assess their impact on perceived QoE.

Impact of the network access

End-users may be served by different network service providers, along with different bandwidths and data communication technologies (e.g ADSL, Wi-Fi, Fiber). Our measurements reflect observed web browsing quality where

$$Bandwidth_{ADSL} < Bandwidth_{Wi-Fi} < Bandwidth_{Fiber}$$

Fig. 52 depicts the observed PLT loading times when requesting HTTP/1.1 following different network access. For a main web page located in North America, moving from an ADSL to Fiber network access brings a reduction of the loading time by 42.03%. These measurements take into account Web View probes located in Europe and when a main web page is in Asia, an ADSL network access helps in reducing the PLT compared to a Wi-Fi network access. Although a Fiber network access offers greater bandwidth, main web pages located in Asia bear the particularity of downloading content mainly from Asia which increases network delay for a European end-user and the increased bandwidth does not contribute to an important decrease in loading times.



(a) ADSL network access

(b) Wi-Fi network access

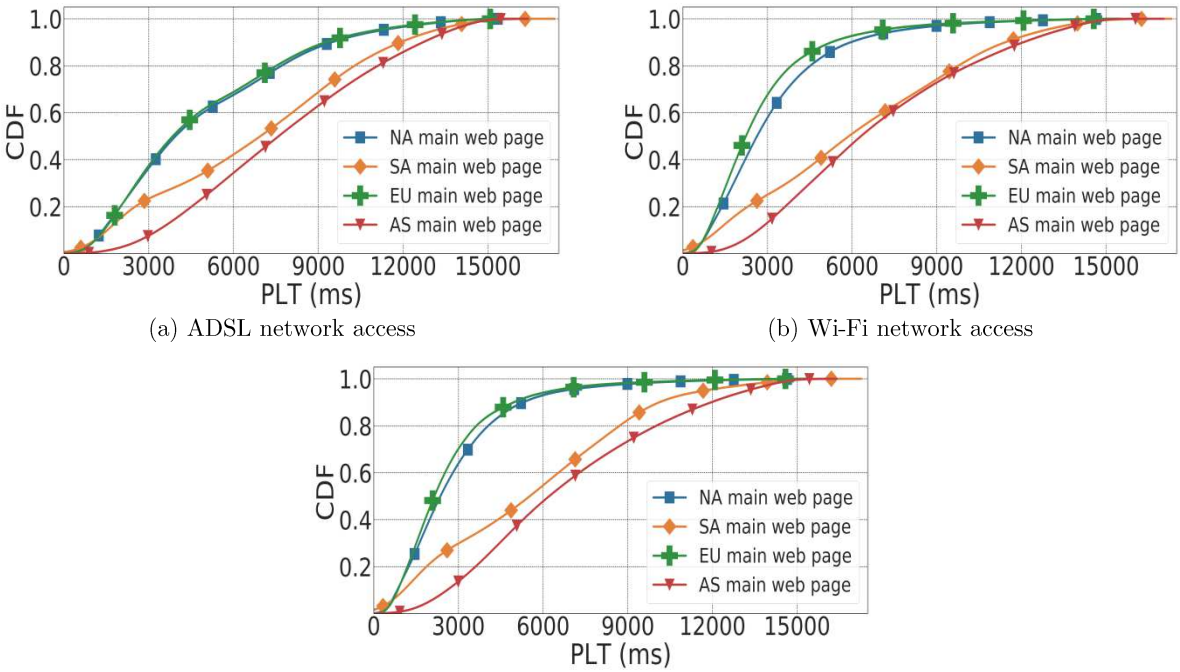
(c) Fiber network access

Fig. 52: PLT when requesting HTTP/1.1

When requesting the QUIC protocol, as shown in Fig. 53, the end-user's QoE increases proportionally to bandwidth increase for main web pages located in Europe and North America. The decrease in the average loading times for these web pages is mainly due to increased bandwidth when moving from ADSL to Fiber but also thanks to web servers in Europe and North America implementing at a higher rate the QUIC or HTTP/2 protocol. On average depending on the network access (ADSL versus Fiber), an end-user's perceived PLT is decreased by 30.25%. Increased end-user QoE is tightly linked to the corresponding network access and requested protocol.

When focusing on the PLT for main web pages located in Asia, requesting QUIC through a Fiber network access (Fig. 53(c)) versus requesting HTTP/1.1 through a Fiber

network access (Fig. 52(c)) does not bring any enhancement. As discussed in section 4.1.3, main web pages in Asia deliver contents mainly in HTTP/1.1 and when requesting QUIC, fallback is firstly made to HTTP/2 and lastly to HTTP/1.1. Globally, when requesting HTTP/2 or QUIC, irrespective of the main web page location, web pages' loading times are decreased on average by 19.73% from ADSL to Wi-Fi, 16.02% from Wi-Fi to Fiber and 30.25% from ADSL to Fiber.



(a) ADSL network access

(b) Wi-Fi network access

(c) Fiber network access

Fig. 53: PLT when requesting QUIC

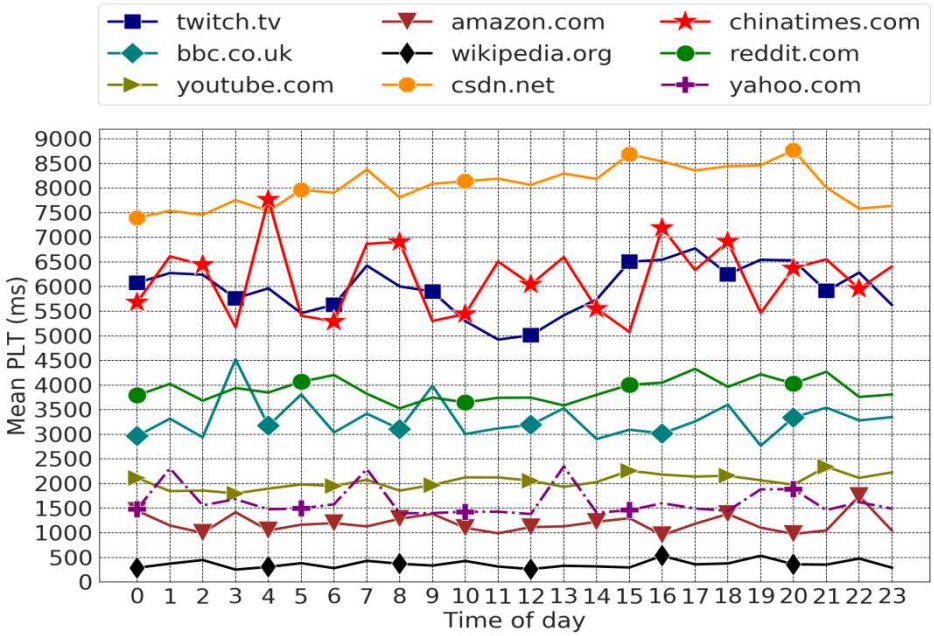


Fig. 54: Mean PLT of different websites at different times of the day

Parameters	FP	TFVR	PLT
Main web page location	✓	✓	✓
Requested protocol		✓	✓
Web server location		✓	✓
Round Trip Time	✓	✓	✓
Website category	✓	✓	✓
Number of objects	✓	✓	✓
Main HTML page size	✓	✓	✓
Size of objects		✓	✓
Types of objects		✓	✓
Visible portion			✓
Number of domains		✓	✓
Use of ad blockers		✓	✓
Time of day	✓	✓	✓
Network access	✓	✓	✓

Table 13: Parameters impacting web browsing quality

Impact of time of the day

Our measurements have been performed round the clock all day long, since following different times of the day a website might be visited by a larger number of end-users, and thus increasing the time needed for the servers to deliver different objects. Furthermore, the overall network state to reach the remote servers might be overloaded, e.g peak hours versus off-peak hours. We have thus assessed the impact of the time of day on the perceived page load time, depicted by the Fig. 54. While some websites' loading times stay relatively the same all day long, e.g *wikipedia.org*, other websites depending on their category (thus type and size of content) may have their inner structure changing several times per day (e.g *News*: *chinatimes.com* or *Entertainment*: *twitch.tv*). Our measurements have been performed in Europe, and we can notice that the website *chinatimes.com* average PLT increases drastically at 04H CET (12H in Asia) and 08H CET (16H in Asia). We can thus suppose that following these times of the day in Asia that the website has a higher visit rate and a European end-user experiences the side effects. In overall, we can notice that between 12H CET - 14 CET or 20H CET -22H CET, the average loading times perceived by end-users also increase which is most of the time due to the network state.

Takeaways: Globally, when requesting HTTP/2 or QUIC, irrespective of the main web page location, web pages' loading times are decreased on average by 19.73% from ADSL to Wi-Fi, 16.02% from Wi-Fi to Fiber and 30.25% from ADSL to Fiber. Web pages loading times are impacted following different times of the day mainly due to their visit rate (web servers impacted) and network state (overloaded during peak hours). The corresponding loading times might also fluctuate depending upon the website category since the number of downloaded objects (and MIME type and size) might increase.

4.3 Conclusion

The Web browsing eco-system is complex where several factors are implicated. We have been able to identify, as per an end-user located in Europe, the different factors which can impact QoE at different phases of web pages' loading progression.

In terms of web browsing delivery, we have identified that since 2014, the amount of downloaded resources has increased on average by 17% for Alexa websites ranked 1-2000 and by 31% for websites ranked 5000-10000. We identified that the average number of scripts and

images has increased by 53% over the last 15 years from past studies [51] and by 7% from recent studies [11]. HTTP/1.1 is still widely used by web servers and in particular when downloading objects for websites whose main web page is in Asia. HTTP/2, although standardized in 2015, is deployed at a low pace and even if an end-user makes use of the latest updated web browser, content is downloaded in both HTTP/1.1 and HTTP/2. When comparing measurements in March 2018 versus March 2019 for the Top 10,000 Alexa websites, HTTP/2 protocol distribution has increased by only 4%. No website replies in 100% QUIC which is mainly deployed on *Google* web servers. Since May 2019 where the QUIC standardization process has been started at the IETF, major CDNs have started implementing QUIC in their web servers. When websites are regrouped as per different categories, they share most of the time same types of content. Quality differs between different categories mainly due to the types of content and Internet protocol to deliver them.

When focusing on factors which can decrease or increase web pages' loading times, HTTP/1.1, HTTP/2 and QUIC do not bring a fuliginous asset for the FP value, QUIC Repeat on the other hand brings a notable enhancement on the observed FP. HTTP/1.1 impacts the TFVR, HTTP/2 and QUIC bring along close loading times and QUIC Repeat enhances the most the TFVR (reduction up to 54.2% for a main web page in North America). The Page Load Time is only enhanced for the Repeat mode since web servers do not fully implement new Internet protocols; HTTP/1.1 is omnipresent. The greater is the number of domains serving content for a web page, the higher will the cumulative DNS time be and hence will increase the TFVR and PLT value. The FP is not impacted by the number of domains serving content since the rendering of the first pixel is most of the time thanks to content downloaded from the origin web server. CDNs are now widely used and some big CDN providers have a very large footprint on the world. For web browsing, our measurement campaign highlights its benefits, since in the *Repeat* mode, the Page Load Time can be reduced by an average of 43.1% when requesting HTTP/2 and 38.5% when requesting QUIC, leading to a better perceived quality by end-users. In short, about more than a quarter of the global contents are served by CDNs but in Asia, the use of well-known CDNs is limited to an average of 10%. Following the end-user's environment, in particular the used network access, web pages' loading times are decreased on average by 19.73% from ADSL to Wi-Fi, 16.02% from Wi-Fi to Fiber and 30.25% from ADSL to Fiber. Web pages loading times are impacted following different times of the day mainly due to their visit rate and network state.

The Table 13 shows the different parameters which play the most important role in either increasing or decreasing perceived quality as per the First paint (FP), TFVR (Time for Full Visual Rendering) and PLT (Page Load Time). Since only measurements performed by Web View probes in Europe have been considered, the main web page location can contribute positively or negatively to loading times. When requesting the main *html* of a web page located in Asia, the corresponding time will be higher (Round-Trip-Time and network delay) and is generally downloaded in HTTP/1.1. Following the network access (ADSL versus Fiber) and time of the day (peak or off-peak periods), the network delay can fluctuate. The number, size and type of objects and the main HTML page size have a strong link with the website's Alexa-referenced category listing (e.g *Search-Engine* category websites are composed of small number of objects of small size but *News* category websites are longer in `scrollHeight`, thus composed of many objects and served by a large number of domains, etc.). Finally, other factors like the time of day or the visible portion can have an impact on the quality but at a lower degree.

Chapter 5

Predicting Web Browsing Quality

Contents

In this Chapter, we focus on Web browsing quality prediction for the visible portion and entire web page. With more than 1.7 Billion websites in 2019, it is time-consuming and not feasible to measure the different loading times following a large set of configurations. Our aim is to make use of Machine Learning techniques to learn the rules qualifying and quantifying the web browsing quality of the Top 10,000 Alexa websites. From these identified rules-based models, we predict the perceived QoE of end-users on never assessed websites. When applying these rules-based models to predict the time to load the visible portion of web pages, the prediction correctness on never measured websites is 90.4%. When focusing on the web browsing quality prediction for the entire web page, the classical clustering techniques used together with decision Trees yield up to 42.1% of error in prediction rates. This is mainly due to a large number of websites whose loading times fluctuate a lot. We have thus applied a Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) with Gaussian Mixture Model (GMM) emissions distributions to detect these fluctuations and hence enrich our rules-based models which has helped in increasing the correctness in prediction rates up to 25% for the entire web page.

5.1 Quality prediction of web pages' visible portion

In this section, we focus on the quality prediction of the visible portion of web pages. Web pages are all different and corresponding loading times of the visible portion may differ, e.g *Search-Engines* versus *News* category. This is mainly due to some categories where the visible portion is higher, e.g for a web browser viewport 1920x1080, a web page belonging to the *Search-Engine* category will have a visible portion ranging between 80% and 100% while a web page belonging to the *News* category can have the visible portion at first glance ranging between 7% and 50%. Several studies have been conducted with real end-users assessing the perceived QoE through Mean Opinion Score (MOS) benchmarking. We firstly

make use of these satisfaction degrees defined by the state of the art to identify rules-based models through decision trees. From these identified rules, we predict the web browsing quality of the visible portion. The error in prediction is high since these satisfaction degrees have been defined from a limited number of web pages. Thanks to our large dataset, we defined new satisfaction degrees and our corresponding rules-based model has proven to predict correctly the web browsing quality of the visible portion of web pages.

5.1.1 Decision trees based on satisfaction degrees

To be able to get hold of the parameters qualifying and quantifying web browsing quality, we have made use of Decision Trees. Decision Trees build classification or regression (one or more independent variable which may determine an outcome) models into the form of a tree structure. Data is broken into smaller and smaller subsets while at the same time an associated tree is incrementally developed. The final result is a tree with decision nodes (two or more branches) and leaf nodes (a classification or decision). Random Forests can be assimilated to random decision trees that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of individual trees.

From 244 Million measurements, decision tree rules are built from 40% of the entire dataset, which is denoted by the *training dataset* and the left 60% is used to validate the obtained rules, denoted by the *validation dataset*.

Satisfaction degrees	**Loading times (ms)**
Instant response	< 100
Seamless response	100 - 1000
Average response	1000 - 3000
Critical response	3000 - 10000
Bad response	> 10000

Table 14: State of the art satisfaction degrees

Decision Trees based on state of the art satisfaction degrees

In order to identify these parameters' thresholds we have firstly taken into account state of the art satisfaction degrees[41] based on end-user sociological perceived feelings coupled to Mean Opinion Score (MOS) studies [76] [84] [85] [36]. These different satisfaction degrees are assimilated to the time to render the visible portion of a web page without scrolling, i.e the TFVR, and are represented in Table 14 where:

- *Instant response*: indicates a high responsiveness from the visited web page,
- *Seamless response*: indicates that the end-user is happy with the overall experience,
- *Average response*: indicates that the end-user feels the delay for the web page to load acceptable,
- *Critical response*: indicates that the end-user strongly feels the bad side effects of the long web page loading,
- *Bad response*: indicates that the end-user is not happy at all and can give up the browsing.

Our decision tree has been built from the training dataset, taking into account for each measurement the TFVR value and satisfaction degrees illustrated in Table 14. The obtained decision tree is represented by 184 different nodes where for each satisfaction degree:

- *Instant response*: our measurements do not have any TFVR loading time lower than 100 ms and make this class unnecessary.

[41] https://www.nngroup.com/articles/website-response-times

- *Seamless response*: 24.32% of the training dataset has a TFVR in this range and the main impacting factors for this classification are the main web page located in North America or Europe, a low RTT value for the main web page, QUIC Repeat as requested protocol and a Fiber network access.
- *Average response*: 30.67% of our training values belongs to this class and the main factors are the main web page located in North America or Europe, the requested Internet protocol being HTTP2, QUIC or QUIC Repeat, a Wi-Fi network access, the use of an ad blocker and RTT value between 18 ms and 101 ms.
- *Critical response*: 43.78% of the training measurements have a TFVR in this class, impacted by the main web page located in Europe, South America or Asia, the visible portion between 22.76% and 41.39% (these web pages have a larger scroll height), the objects served by various content servers mainly located in Asia and Wi-Fi or ADSL as network access.
- *Bad response*: 1.23% of the training dataset is classified as Bad response, mainly because of the use of Wi-Fi network access and HTTP/1.1 as requested protocol.

When going through the obtained decision tree, as an example, the corresponding rules defining the *Seamless response* class is as follows:

$$\{(2\% \leq quicProtocolShare \leq 63.94\%) \wedge (requestedProtocol = Quic\ Repeat)\} \\ \wedge \{(alexaRanking \leq 42) \wedge (mainWebPageRTT \leq 18ms)\}$$

or

$$\{mainWebPageLocation = (NA \vee EU)\} \wedge \{requestedProtocol = Quic\ Repeat\} \\ \wedge \{networkAccess = Fiber\} \wedge \{downloadedObjects \leq 21\}$$

When taking into account the parameters identified in Table 13 from section 4.3, following different state-of-art satisfaction degrees, the Table 15 illustrates which of these parameters contribute the most to a specific satisfaction degree. While the network access plays an important role for 80% of the classes, a *seamless response* is generally met with a Fiber access network, an *average response* for Wi-Fi or Fiber and *critical* or *bad response* when making use of an ADSL network access. The visible portion only counts for the *average response* when being less or equal than 29.25% and for critical response when less or equal than 15.95%.

Parameters	**Instant Response**	**Seamless Response**	**Average Response**	**Critical Response**	**Bad Response**
Main web page location		✓		✓	✓
Requested protocol		✓	✓	✓	✓
Web server location			✓	✓	✓
Round Trip Time		✓	✓	✓	✓
Website category				✓	✓
Number of objects		✓	✓	✓	✓
Main HTML page size					✓
Size of objects		✓		✓	✓
Types of objects		✓		✓	✓
Visible portion			✓	✓	
Number of domains			✓	✓	✓
Use of ad blockers		✓			
Time of day				✓	✓
Network access		✓	✓	✓	✓

Table 15: Parameters impacting QoE based on state-of-art satisfaction degrees

Once this decision tree has been built based on the training dataset, we evaluated it with the validation dataset. The Table 16 depicts the obtained classification confusion matrix, where the diagonal represents the percentage of measurements correctly predicted. We can see that 84.79% of the predictions are good, but more than 15% of the validation dataset is wrongly predicted. Looking at the matrix, the values identified in Table 14 and measured TFVR values, we can say that the *Instant Response* class is unnecessary (no measurement in this class) and that the prediction error rate for the *Critical response* class is important (about 10%) because the loading time range is too wide. New ranges of satisfaction degrees are thus needed to identify better the sets of rules which could decrease the prediction error rate.

	Actual class				
Predicted class	**Instant**	**Seamless**	**Average**	**Critical**	**Bad**
Instant	0	0.15%	0	0	0
Seamless	0	22.21%	1.60%	0	0
Average	0	1.96%	29.01%	8.07%	0
Critical	0	0	0.04%	32.35%	0.01%
Bad	0	0	0.02%	3.36%	1.22%

Table 16: Classification confusion matrix based on user satisfaction degrees

Decision Trees based on estimated satisfaction degrees from clustering

Having seen that the proposed classification of the satisfaction degrees from literature is not optimal for current Web Browsing experience, we decided to define a new one, based on our huge dataset of web browsing measurements on the top 10,000 Alexa websites. We thus look for the best satisfaction degrees using clustering (*K-Means*) to identify the different classes.

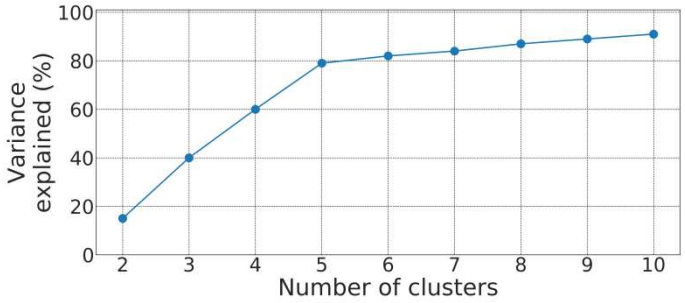


Fig. 55: Explained variance for TFVR values

Satisfaction degrees	**Loading times (ms)**
Good response	< 1232
Fair response	1232 – 3486
Moderate response	3486 – 6715
Worse response	6715 - 9281
Poor response	> 9281

Table 17: Estimated satisfaction degrees derived from our measurements

The *K-Means* algorithm divides a set of samples X into K disjoint clusters C, where each cluster is described by the mean μ of the different samples in the cluster. In order to determine the value K, we use the *Elbow* method where we assess the percentage of explained variance (ratio between-group variance to the total variance) as a function of the number of clusters. As shown in Fig. 55, the value $K = 5$ seems to be the good Elbow Criterion, and

using it, we can identify the 5 clusters representing the different satisfaction degrees, illustrated in Table 17.

The first and second satisfaction degrees, i.e *Instant* and *Seamless* from Table 14 have been merged into one degree and the *Critical response* satisfaction degree is distributed and split into two degrees. The estimated satisfaction degrees illustrated in Table 17 are firstly more uniformly distributed compared to Table 14 and secondly regarding the *Instant response* satisfaction degree, the estimated values are more realistic since even with a Fiber access network, use of caches (local and remote) and HTTP/2 Internet protocol distribution, the home page of *wikipedia.org* TFVR is 102 ms for a web browser viewport 1920x1080.

Our decision tree is then re-built from the training dataset, taking into account for each measurement the TFVR value and estimated satisfaction degrees illustrated in Table 17. The obtained decision tree is represented by 176 different nodes where for each satisfaction degree:

- *Good response*: 23.32% of the training dataset has a TFVR in this range and the main implicated parameters are the main web page located in North America or Europe, a low RTT and number of domains serving contents from North America or Europe, with replies delivered over the QUIC protocol.
- *Fair response*: 27.31% of the training measurements are classified in this class and the main impacting factors are namely the main web page located in North America, a RTT value less than 105.5ms, a Wi-Fi or Fiber network access, resources downloaded from North America, Europe or Asia in HTTPS.
- *Moderate response*: 31.26% of our training dataset leads to *Moderate response*, mainly based on the main web page located in Europe or Asia, the requested Internet protocol being HTTP/1.1, HTTP/2 or QUIC, the visible portion of the websites less than 13.25% and objects delivered from Asia or South America.
- *Worse response*: 12.09% of the training dataset has a TFVR in this range with the main factors being the main web page located in South America or Asia, the number of downloaded resources between 45 and 85 (mainly downloaded from Asia), RTT value greater than 185.5 ms and the number of domains from Asia greater than 4.
- *Poor response*: 6.02% of the training measurements lead to a poor quality, mainly because of the main web page located in Asia, HTTP/1.1 used as requested protocol, a high number of downloaded objects and the use of an ADSL network access.

When taking into account the parameters identified in Table 13 from section 4.3, following different estimated satisfaction degrees, the Table 18 illustrates which of these parameters contribute the most to a specific satisfaction degree.

Parameters	**Good Response**	**Fair Response**	**Moderate Response**	**Worse Response**	**Poor Response**
Main web page location	✓	✓		✓	✓
Requested protocol	✓	✓	✓		✓
Web server location	✓	✓	✓		✓
Round Trip Time	✓	✓	✓	✓	✓
Website category	✓				
Number of objects	✓	✓	✓	✓	✓
Main HTML page size			✓		✓
Size of objects	✓			✓	✓
Types of objects					✓
Visible portion			✓		
Number of domains	✓	✓	✓	✓	✓
Use of ad blockers	✓	✓			
Time of day			✓	✓	✓
Network access	✓	✓	✓	✓	✓

Table 18: Parameters impacting QoE upon estimated satisfaction degrees

	Actual class				
Predicted class	**Good**	**Fair**	**Moderate**	**Worse**	**Poor**
Good	22.19%	0.04%	0	0	0
Fair	1.13%	25.17%	0.82%	0	0
Moderate	0	2.10%	29.22%	0.43%	0
Worse	0	0	1.20%	11.63%	0.06%
Poor	0	0	0.02%	0.03%	5.96%

Table 19: Classification confusion matrix based on estimated satisfaction degrees

	Actual class				
Predicted class	**Good**	**Fair**	**Moderate**	**Worse**	**Poor**
Good	6.51%	0.16%	0	0	0
Fair	0.76%	21.09%	0.72%	0	0.02%
Moderate	0.01%	1.96%	35.02%	0.12%	0.01%
Worse	0	0	3.60%	17.74%	1.03%
Poor	0	0	0	1.21%	10.04%

Table 20: Classification confusion matrix to validate the accuracy of our rules-based model

Once this decision tree has been built based on the training dataset, we evaluated it with the validation dataset. The Table 19 depicts the obtained classification confusion matrix, where the diagonal represents the percentage of measurements correctly predicted. We can see, looking at the diagonal representing the percentage of measurements correctly predicted, that our rules can efficiently predict 94.17% of the validation dataset. Only 5.83% of the validation dataset is wrongly predicted, which is 2.61 times less when compared to the rules-based model obtained previously. We can then think that this model is better.

Takeaways: Websites are all different among them. Although we have identified the main factors which can impact web browsing quality, we have made use of Decision Trees along with new satisfaction degrees to identify the thresholds of these parameters which qualify and quantify good or bad web browsing experience. Since some parameters are strongly linked among themselves, the identified sets of rules qualifying QoE can efficiently predict 94.17% of the validation dataset.

5.1.2 Accuracy of our rules-based model

In order to verify the correctness of the obtained decision tree from estimated satisfaction degrees, we have performed in February 2019 measurements on the Alexa websites ranging from rank 10,000 to 15,000. The dataset sums up to the measurement of 4861 never-assessed before distinct websites, which represent 2.7 Million different measurements. The Table 20 illustrates the obtained classification confusion matrix for these measurements when applying our rules-based model.

Among our measurements, 6.51% of the measurements provide a *Good response* (low RTT and small number of objects), 21.09% for *Fair response* (low RTT and main web page in Europe), 35.02% for *Moderate response* (large number of domains serving contents from Europe and Asia), 17.04% for *Worse response* and finally 10.04% of the measurements yield a *Poor response* (high HTTP/1.1 reply distribution and objects mainly delivered from South America and Asia).

In short, 90.4% of the overall new dataset was correctly predicted. This is a bit less but it is related to websites which have never been measured before. We can then estimate that our rules-based model can efficiently predict web pages' visible portion quality for any website an end-user located in France browses.

	Actual class					
Predicted class	**Good**	**Fair**	**Moderate**	**Worse**	**Poor**	**Bad**
Good	3.02%	3.2%	2.07%	0	0	0
Fair	0.82%	7.68%	10.78%	4.02%	0	0
Moderate	0	0.69%	17.21%	5.79%	0.44%	0
Worse	0	0.09%	6.03%	11.9%	1.7%	0.06%
Poor	0	0	0.15%	2.46%	8.02%	1.63%
Bad	0	0	0	0.07%	2.1%	10.07%

Table 21: Classification confusion matrix for the PLT-HAR on never measured websites

5.2 Quality prediction of the entire web page

We focus in this section on the quality prediction following the entire web page. Compared to our study in section 5.1 where our study was related to the visible portion of web pages, we now pay particular attention on the ability to predict the overall quality of a web page, i.e the QoE for the visible and non-visible portion of a web page. When comparing the time frame following the TFVR versus the PLT, a greater number of factors can be implicated for the PLT time frame. As an example, with a web browser viewport 1920x1080 and web browsing the website *www.stackoverflow.com*, the TFVR represents the time to load 7% of the entire web page. The parameters (and their thresholds) involved during the PLT time lapse have to be re-evaluated. We focused on PLT-HAR timings and identified different quality degrees, followed by decision trees to identify the rules and then predict the quality. We have noticed in our study that the error prediction rate is high, i.e 42.1%. The error in prediction is not uniform among the different satisfaction degrees, and we have identified that the error prediction rate is high for websites where content is regularly renewed which creates fluctuations in the loading times. The TFVR quality prediction is not impacted since only a small portion of these websites are visible. These fluctuations must be taken into account and we have used a HDP-HMM with GMM emissions distributions to identify them and enrich our model. This technique increases the quality prediction correctness for a whole web page up to 25%.

5.2.1 Decision Trees based on quality degrees

We have gone through the same process identified in section 5.1.1 to build our rules-based model. Since the state of the art satisfaction degrees focus on the visible portion of web pages only, we have applied clustering (*K-Means*) to identify the different classes on the Top 10,000 Alexa websites. For the PLT-HAR, 6 classes are identified and these quality degrees have been used along with the training dataset to obtain our rules-based model. When this model is applied to the validation dataset, the error in prediction rate is 31.4%. The model applied on never assessed websites, i.e Alexa websites ranged 10,000 – 15,000, yields an error prediction rate of 42.1% which is highlighted in Table 21.

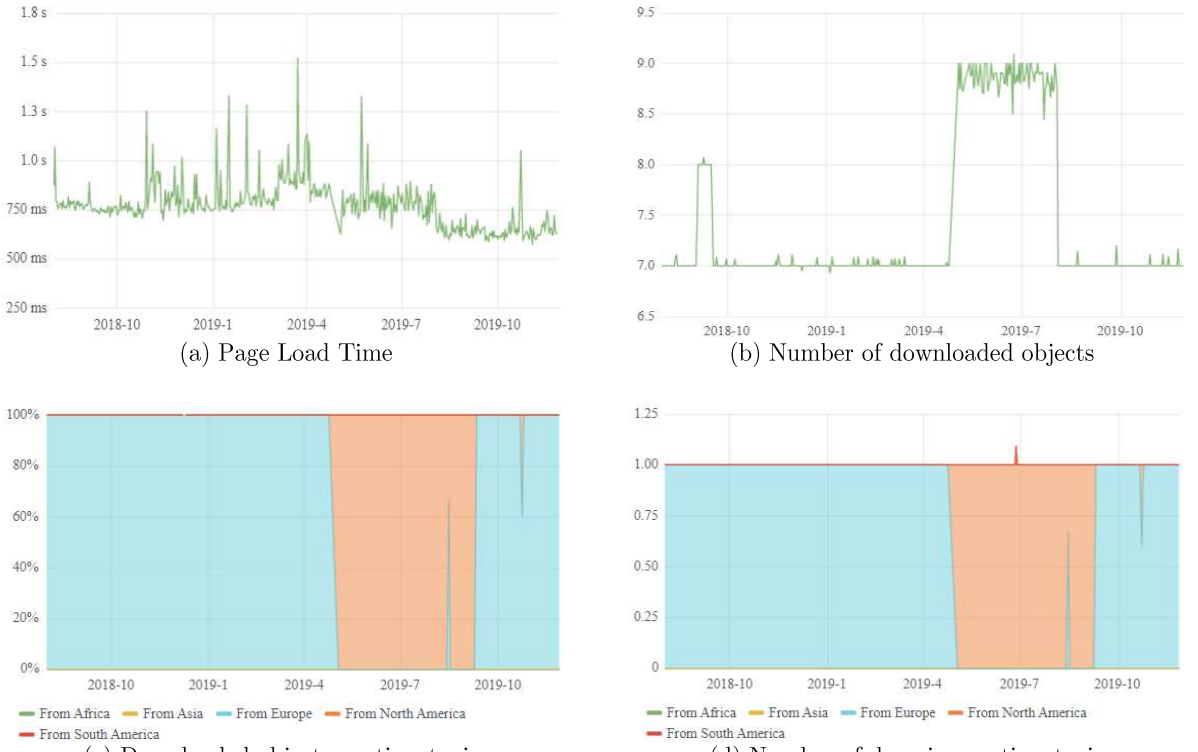
The error in prediction rate is important for two classes, i.e *Moderate* and *Worse quality* and we wanted to understand why error rates are concentrated between the two classes. Following the parameters we have identified in Table 13 in section 4.3, we noticed that the thresholds of the parameters *Round-Trip-Time, Number of objects, Web Server Location* and *Requested Internet Protocol* are more important. That means that the location of servers delivering content might change regularly and thus increase the overall networking time. The amount of content composing the web page might also change regularly and impact our quality prediction. To obtain ground proof of this, we have used the monitoring website of our tool. We analyzed several web pages' Page Load Time belonging to different classes and we hereunder expose our findings related to websites having smooth loading times

(e.g *wikipedia.org, google.com, etc.*) versus websites changing constantly (e.g *baidu.com, tumblr.com, etc.*).

Smooth loading times over time

As discussed in section 4.2.3 through Fig. 54, fluctuations in the loading times of the website *wikipedia.org* are mean at different times of the day. The Fig. 56 and Fig. 57 illustrate the web delivery and loading time (PLT-HAR) of this website between August 2018 and November 2019 when requesting HTTP/1.1 and HTTP/2 through a Google-Chrome web browser v.63. When requesting HTTP/1.1 from a Web View probe in Europe, the average loading time for this period is 771 ms (Fig. 56(a)) where an average number of 7.4 objects (Fig. 56(b)) are downloaded. Although the PLT-HAR ranges most of the time between 600 ms and 900 ms, we can identify through Fig. 56(b) that the number of downloaded objects from web servers located in North America is increased by 2. This increase in the number of downloaded objects has no impact on the PLT-HAR since their average size is 27 ko and networking receiving time being 25 ms. But we can also identify that when requesting HTTP/1.1, the number of outliers in the PLT-HAR is more important (compared to requesting HTTP/2 discussed hereunder).

When requesting HTTP/2 from a Web View probe in Europe, depicted by Fig. 57, the average PLT-HAR is 807 ms (Fig. 57(a)) which is 36 ms greater than when requesting HTTP/1.1. While the average number of downloaded objects stays the same, i.e 7.4, as from September 2019, objects are downloaded from both North America and Europe. When comparing the average networking receiving time for this period with HTTP/1.1 measurements, the total receive time is on average 30 ms greater since content is downloaded from North America, hence contributing to the increase in the PLT-HAR. When requesting HTTP/2, we can notice that the loading times bear less outliers thanks to multiplexing.



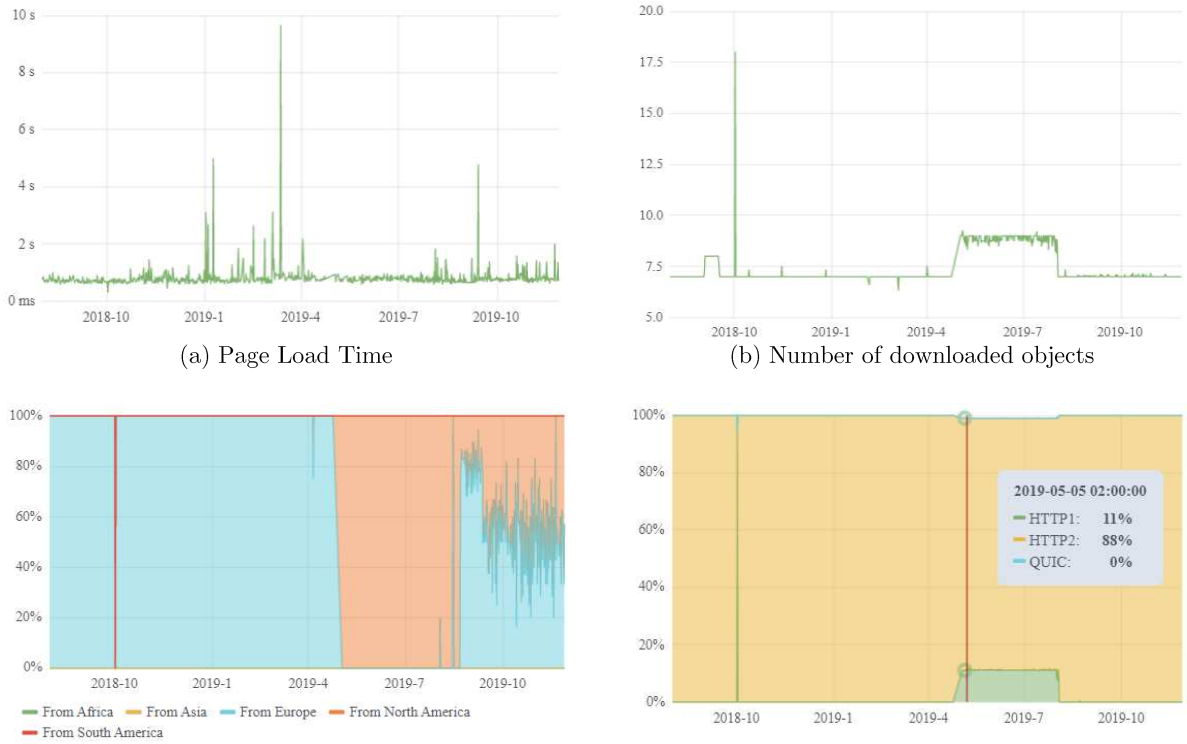
(a) Page Load Time

(b) Number of downloaded objects

(c) Downloaded objects continent-wise

(d) Number of domains continent-wise

Fig. 56: Website *wikipedia.org* delivery and loading time when requesting HTTP/1.1



(a) Page Load Time

(b) Number of downloaded objects

(c) Downloaded objects continent-wise

(d) Protocol reply distribution

Fig. 57: Website *wikipedia.org* delivery and loading time when requesting HTTP/2

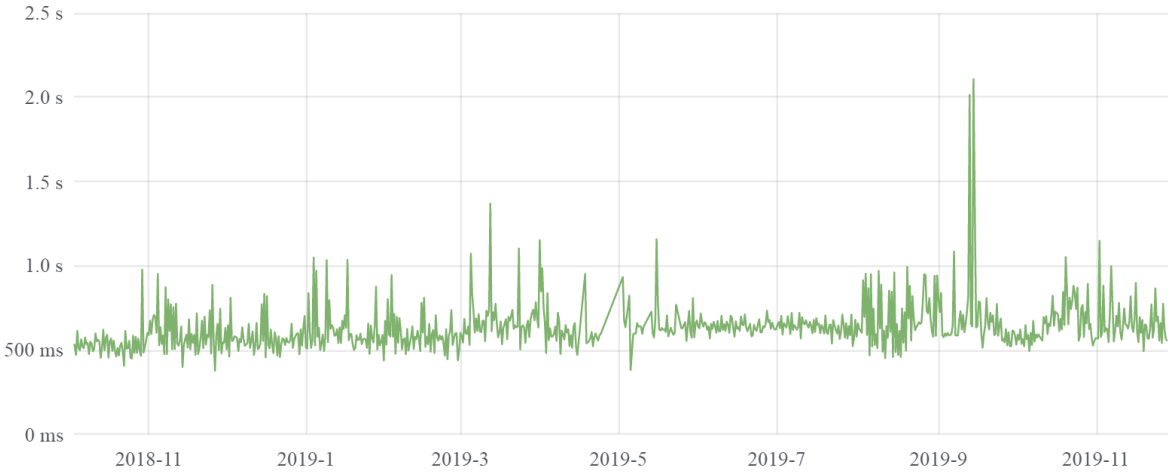


Fig. 58: Website *wikipedia.org* PLT from October 2018 to November 2019

When taking into account measurements performed between October 2018 and November 2019 (Fig. 58), we firstly calculate the number of clusters through the *Elbow* method as per the PLT-HAR values for this website only. As shown in Fig. 59, the value $K = 3$ seems to be the good *Elbow Criterion*, and using it, we can identify the 3 clusters representing the different quality degrees, illustrated in Table 22. When going through the obtained decision tree, the corresponding rules defining the *Good quality* class is as follows:

$$\{networkAccess = Fiber\} \Lambda \{HTTP2 - Replies = 100\%\} \Lambda \{downloadedObjects \leq 7\} \\ \Lambda \{(requestedProtocol = HTTP/2) \vee (requestedProtocol = QUIC)\}$$

and the *Bad quality* class:

$$\{\text{networkAccess} = \text{ADSL}\} \Lambda \{HTTP2 - Replies \neq 100\%\} \Lambda \{downloadedObjects \geq 8\}$$

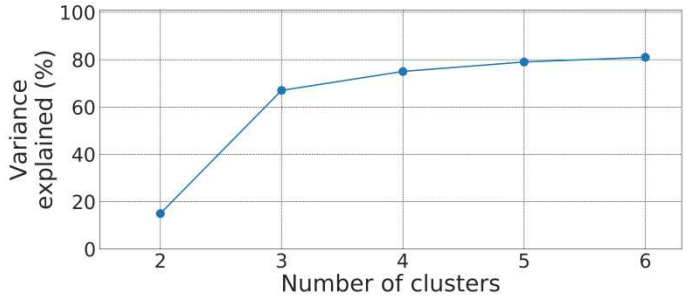


Fig. 59: Explained variance for *wikipedia.org* PLT-HAR values

Quality degrees	Loading times (ms)
Good quality	< 511
Average quality	$511 - 610$
Bad quality	> 610

Table 22: Estimated quality degrees for *wikipedia.org*

	Actual class		
Predicted class	**Good**	**Average**	**Bad**
Good	63.98%	0	0
Average	0	27.67%	0.11%
Bad	0	0.01%	8.22%

Table 23: Classification confusion matrix for *wikipedia.org*

For *wikipedia.org*, the training dataset is composed of measurements from October 2018 and November 2019 and the validation dataset are measurements performed in December 2019. The obtained confusion matrix is illustrated through Table 23 where the prediction correctness is 99.88%. Regarding the website *wikipedia.org*, the perceived web browsing quality is very regular due to the web page design being simple and optimized. Fluctuations in the PLT are rare which contribute to a mean error prediction rate of 0.12%.

Fluctuations in loading times

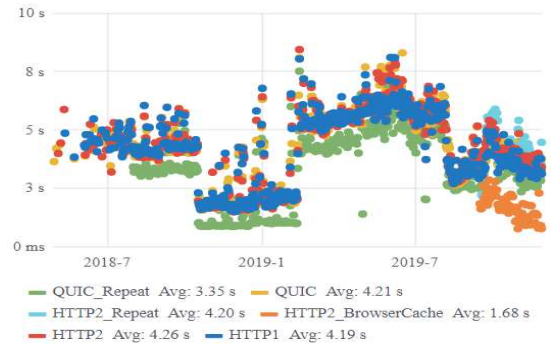
When analyzing websites where fluctuations happen in the loading time regularly, the Fig. 60 depicts the web delivery and PLT-HAR of the website *baidu.com* between April 2018 and November 2019 for a European end-user. Web pages loading times might increase (or decrease) over time and it is important to identify why and which web browsing parameters contribute to this situation.

As shown in Fig. 60(a), irrespective of the requested protocol, the average PLT-HAR for this website changes upon 4 different periods through time:

- Period 1: 4.20s between the 28th April and 14th October 2018,
- Period 2: 2.15s between the 15th October 2018 and 10th February 2019
- Period 3: 5.38s between 11th February and 06th August 2019
- Period 4: 3.48s between 07th August and 28th November 2019

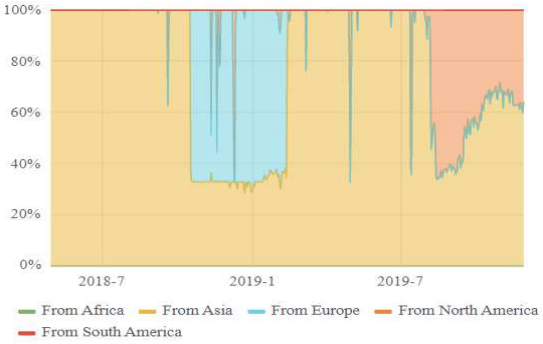
From our monitoring website, we can identify that for *Period 1*, an average number of 17.87 objects are downloaded (Fig. 60(b)) from web servers in Asia (Fig. 60(c)) to render the web page. For the *Period 2*, an average number of 17.92 objects are downloaded from web servers located in Asia and Europe. For the *Period 3*, an average number of 18.71 objects are downloaded mainly from Asia which leads to an average PLT of 5.38s. This increase in the PLT is mainly due to a higher number of downloaded objects which are downloaded from Asia mainly through HTTP/1.1. For the *Period 4*, the average PLT is 3.48s where an

average number of 18.84 objects are downloaded from Asia and North America. We can also identify through Fig. 60(d) that content is downloaded through a higher HTTP/2 distribution.

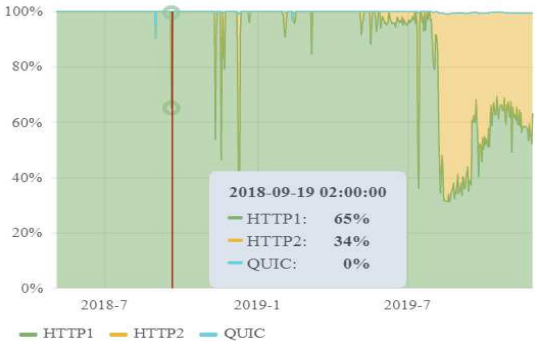


(a) Page Load Time

(b) Number of downloaded objects



(c) Downloaded objects continent-wise



(d) Protocol reply distribution

Fig. 60: Website *baidu.com* delivery and loading time

When focusing on the location of the different web servers delivering content, Web View allows identifying which domains deliver content through CDNs. The Table 24 shows the different web servers delivering contents to an end-user in Europe. For the *Period 2*, the domain *baidu.com* which is a *Same-Origin* domain delivers resources from Asia and *ss1.bdstatic.com* is a *Non-Origin* domain where content is generally served by the CDN provider *Amazon Cloudfront*. For *Period 2* and *Period 4*, *Amazon Cloudfront* serves content either from North America or from Europe.

Periods	**Domains**	**Web server estimated location**	**Average number of downloaded content**
Period 1	*baidu.com*	Hong Kong	16
	panpic.baidu.com	Beijing	2
Period 2	*baidu.com*	Hong Kong	12
	ss1.bdstatic.com	Netherlands	6
Period 3	*baidu.com*	Hong Kong	17
	ss1.bdstatic.com	Hong Kong	1
Period 4	*baidu.com*	Hong Kong	10
	ss1.bdstatic.com	San Francisco	8

Table 24: Content servers for *baidu.com*

The website *baidu.com* web browsing delivery and PLT-HAR is illustrated in Fig. 61 where fluctuations in the loading times happen regularly. The training set is composed of measurements performed between July 2018 and January 2019 and the validation dataset are measurements performed between February and September 2019. The number of clusters through the *Elbow* method for the PLT-HAR values is shown in Fig. 62, where the value $K = 4$ seems to be the good Elbow Criterion, and using it, we can identify the 4 clusters representing the different quality degrees, illustrated in Table 25.

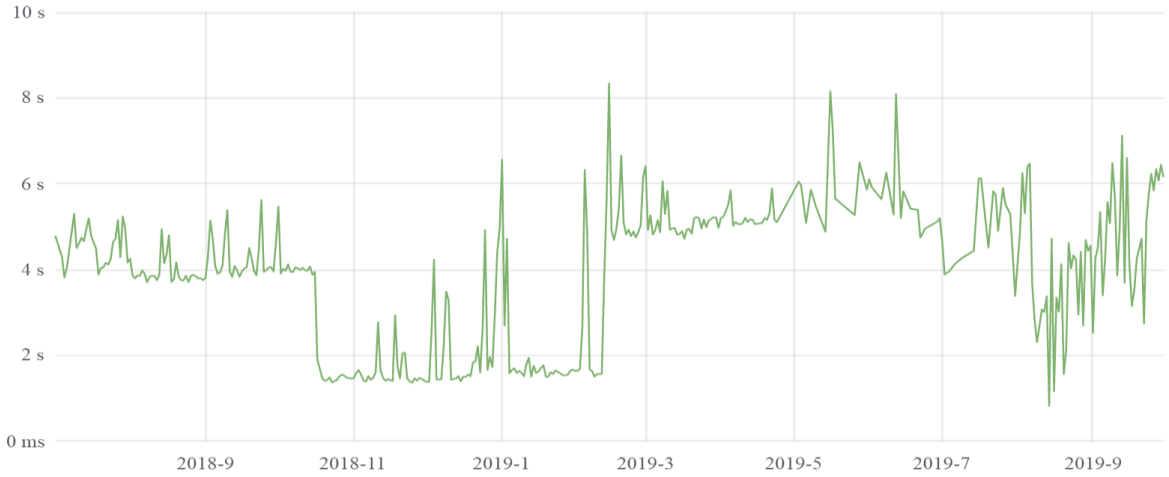


Fig. 61: Website *baidu.com* PLT from July 2018 to September 2019

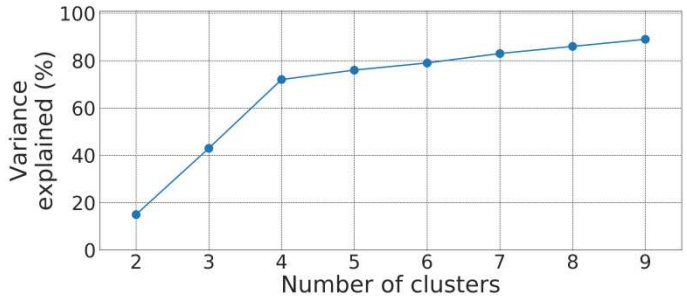


Fig. 62: Explained variance for *baidu.com* PLT-HAR values

Quality degrees	Loading times (ms)
Good quality	<1900
Average quality	1900 – 4200
Bad quality	4200 – 5220
Worse quality	>5220

Table 25: Estimated quality degrees for baidu.*com*

	Actual class			
Predicted class	**Good**	**Average**	**Bad**	**Worse**
Good	16.23%	2.98%	0.01%	0
Average	4.02%	9.16%	10.71%	0
Bad	0	11.22%	21.3%	0.86%
Worse	0	0	4.28%	18.23%

Table 26: Classification confusion matrix for *baidu.com*

When going through the obtained decision tree, the corresponding rules defining the *Good quality* class is as follows:

$$\{networkAccess = Fiber\} \Lambda \{2 \geq domainsInEurope \geq 1\} \Lambda \{downloadedObjectsFromEurope \geq 12\} \Lambda \{downloadedObjectsFromAsia \leq 6\}$$

The identified rules lead to an error prediction rate of 25.08%, as illustrated in Table 26, since during the month of October and November 2019:

1. The PLT values for the website *baidu.com* have changed (Fig. 63(a)), oscillating between 983ms and 5.50s. These values belong to the 3 classes *Good*, *Average* and *Bad response* at the same time and this is why we have a total error prediction rate of 24.22%.
2. The HTTP/2 reply share (Fig. 63(b)) has increased on average by 29% (from 34% to 63%), thus reducing the PLT values.
3. The total size of downloaded contents (overall web page size) illustrated in Fig. 63(c) has decreased from 1.2 Mb to 700 Kb (mainly due to 1 image in *gif* format whose size has decreased from 497 Kb to 705 bytes).
4. Content is intermittently delivered from web servers located in North America and Asia (Fig. 63(d)), which causes the PLT to fluctuate.

(a) Page Load Time

(b) Protocol reply distribution

(c) Overall web page size

(d) Downloaded objects continent-wise

Fig. 63: Website *baidu.com* delivery and loading time for October & November 2019

Furthermore, when taking into account measurements performed on the website *baidu.com* by Web View probes in Europe and web servers delivering content are mainly from North America and Asia, we have large amounts of outliers in the different loading times (confirmed by an Isolated Forest, through Fig. 64, which is an outlier detection technique to identify anomalies instead of normal observations). Due to a high number of outliers, and hence fluctuations, our rules-based model taking into account 4 clusters does not predict web browsing quality for the entire web page efficiently.

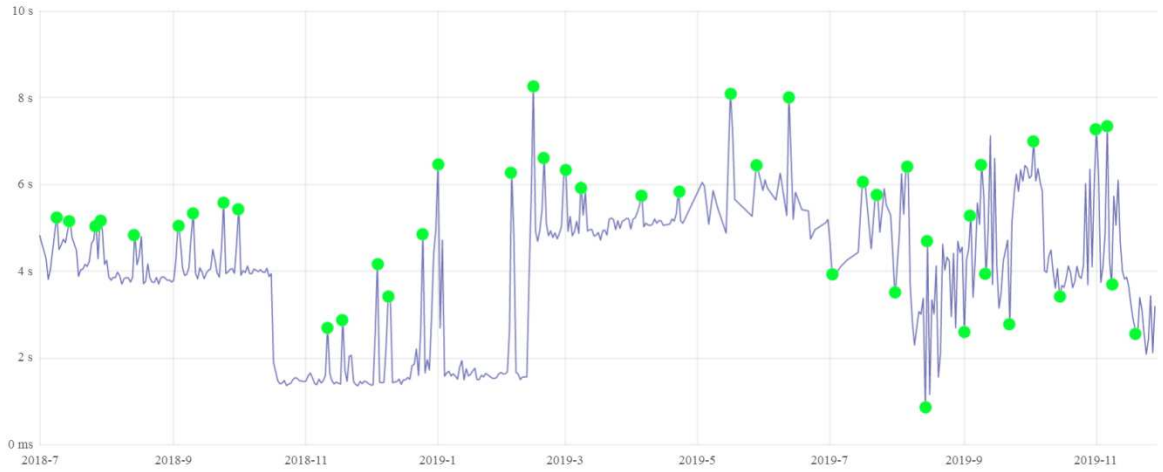


Fig. 64: Website *baidu.com* PLT outliers from July 2018 to November 2019

On a wider scope, based on the 84 websites monitored by Web View visualization website, 14 websites bear a high number of outliers and hence identified rules lead to high prediction error rates (between 21% and 34%). Other techniques, taking into account important fluctuations in the web pages' overall loading times are needed.

Takeaways: In order to identify the rules qualifying and quantifying Web browsing quality, the joint usage of the Elbow method to identify quality degrees and Random Forest provide good predictions when fluctuations are mean in loading times. For websites where important fluctuations happen, this method can yield high error prediction rate. Other techniques, taking into account important fluctuations in the web pages' overall loading times are needed in order to provide an accurate rules-based model when trying to predict web browsing quality for the entire web page

5.2.2 Application of HDP-HMM with GMM emissions distributions on loading times

We have used a Hierarchical Dirichlet Process Hidden Markov Models (HDP-HMM) [126] [127] [128] with a Gaussian Mixture Model (GMM) emissions distributions which helps in identifying finely an infinite number of states with time dependency. The main advantage of this model is to learn the order of the model from the measurements themselves, e.g PLT-HAR values, where outliers are taken into account to identify more finely quality degrees values.

From measurements of the website *baidu.com* from July 2018 to September 2019, HDP-HMM with GMM emissions distributions is applied and the result is depicted in Fig. 65. 9 different states are identified (instead of 4 states in section 5.2.1) and for the different states, the minimum and maximum PLT values are represented through Table 27. Four states, namely $State_2$, $State_4$, $State_7$ and $State_9$, are once again identified compared to Elbow method process in section 5.2.1 where the number of clusters were previously identified as $K = 4$. But from Fig. 65, five additional states (being mainly outliers identified by an Isolated Forest in Fig. 64) are detected where:

$$State_1 \in \{State_2 \vee State_7\}$$
$$State_3 \in \{State_2 \vee State_4 \vee State_7\}$$
$$State_5 \in \{State_4 \vee State_9\}$$
$$State_6 \in \{State_7 \vee State_9\}$$
$$State_8 \in \{State_7 \vee State_9\}$$

Fig. 65: HDP-HMM with GMM emissions distributions applied to *baidu.com* time series

State	Color	PLT(ms)		Time fluctuations(ms)
		Min	Max	
1	*Burlywood*	3.7	4.6	0.9
2	*Light blueviolet*	1.9	4.2	2.3
3	*Yellow*	1.8	5.3	3.5
4	*Beige*	0.8	1.9	1.1
5	*Pink*	0.9	5.2	4.3
6	*Brown*	4.8	9.1	4.3
7	*Aquamarine*	4.2	5.2	1.0
8	*Green*	0.9	7.4	6.5
9	*Dark blueviolet*	2.1	4.2	2.1

Table 27: Inferred states of *baidu.com*

When taking into account the state transition matrix for these five states, we obtain additional information on the parameters creating fluctuations into the loading times but also the underlying relation between the 9 states. Getting hold of the parameters impacting web browsing quality is handy, but obtaining their thresholds is also important to be able to predict finely the perceived quality. When focusing on $State_5$, we want to identify the rules quantifying that state. We thus take into consideration each *Pink* state start and end date time. From the corresponding measurements, we look for the number of clusters through the *Elbow* method for these PLT values. As shown in Fig. 66, the value $K = 4$ seems to be the good *Elbow Criterion*, and using it, we can identify the 4 clusters representing the different clusters' loading times, illustrated in Table 28.

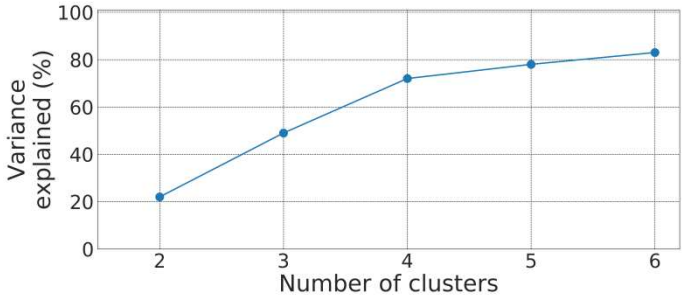


Fig. 66: Explained variance for 5th hidden state

Clusters	Loading times (ms)
C_1	1200 – 1900
C_2	1900 – 2700
C_3	2700 – 3900
C_4	> 3900

Table 28: Estimated clusters for the 5th hidden cluster

From the obtained decision trees ($State_5$ for measurements between July 2018 and September 2019), the rules qualifying the clusters C_1, C_2, C_3 and C_4 through Table 28 are:

$C_1 \rightarrow \{\text{RTT} \leq 18\text{ms}\} \wedge \{(downloadedEuropeObjects \geq 12) \wedge (downloadedAsiaObjects \leq 6)\}$
$C_2 \rightarrow \{18\text{ms} \leq \text{RTT} \leq 27\text{ms}\} \wedge \{downloadedEuropeObjects \geq 12\}$
$C_3 \rightarrow \{27\text{ms} \leq \text{RTT} \leq 41.7\text{ms}\} \wedge \{downloadedNorthAmericaObjects \geq 12\}$
$\wedge \{downloadedAsiaObjects \geq 6\}$
$C_4 \rightarrow \{41.7\text{ms} \leq \text{RTT} \leq 44\text{ms}\} \wedge \{downloadedAsiaObjects \geq 17\}$

As identified before, the $State_5 \in \{State_4 \vee State_9\}$ and from the obtained clusters' rules, we can decompose as:

$$State_5 \in \{State_4\}$$
$$(C_1, C_2) \in \{State_4\}$$

and

$$State_5 \in \{State_9\}$$
$$(C_3, C_4) \in \{State_9\}$$

From the previously obtained rules for $State_4$ and $State_9$ in section 5.2.1, they are enriched with identified rules for C_1, C_2, C_3 and C_4. As an example the $State_4$ is thus represented by:

$$\begin{aligned} Enriched\ Good\ Response = {} & (C_1\ rules) \cup (C_2\ rules) \cup (Good\ Response\ Rules) \\ & = \{networkAccess = Fiber\} \\ & \wedge \{2 \geq domainsInEurope \geq 1\} \\ & \wedge \{downloadedObjectsFromEurope \geq 12\} \\ & \wedge \{downloadedObjectsFromAsia \leq 6\} \\ & \vee \{RTT \leq 18ms\} \end{aligned}$$

The different quality degrees identified through Table 25 are all enriched following the above identified states. From the obtained rules qualifying a corresponding quality degree, the PLT is then predicted for measurements of the validation dataset (February to September 2019). The Table 29 exposes the obtained confusion matrix for the website *baidu.com*. The success in quality prediction is 96.42% which is an increase of 21.5% compared to the prediction process identified through Table 26.

	Actual class			
Predicted class	**Good**	**Average**	**Bad**	**Worse**
Good	19.17%	0.67%	0	0
Average	1.16%	21.07%	0.36%	0
Bad	0	0.84%	26.02%	0.07%
Worse	0	0.21%	0.63%	26.22%

Table 29: Classification confusion matrix for *baidu.com* from hidden states

Websites	**Prediction success (%)**		**Number of clusters/states**	
	Decision Trees with K-Means	Decision Trees and HDP-HMM with GMM	K-Means clusters	HDP-HMM with GMM
amazon.com	63.89	72.98	7	13
bbc.com	83.02	92.27	3	10
cdiscount.com	87.51	89.23	2	3
chinatimes.com	76.90	91.22	3	9
csdn.net	84.12	92.60	6	11
irishtimes.com	89.08	92.65	3	5
leboncoin.fr	81.29	97.55	6	10
lefigaro.fr	74.99	89.10	5	11
reddit.com	81.88	94.16	4	8
spiceworks.com	82.59	91.65	3	4
taobao.com	94.22	95.17	4	7
tumblr.com	71.25	86.95	5	6
twitch.tv	60.56	89.23	5	16
yahoo.co.jp	61.13	88.27	6	13

Table 30: Prediction success rates and identified states

5.2.3 Accuracy of our solution

The Web View public visualization websites exposes the measurements of 84 websites which we have taken into consideration to assess the accuracy of our solution. As discussed into section 5.2.1, among these 84 websites, 14 websites have their PLT-HAR value which fluctuates regularly. For the other 70 websites, Random Forests coupled to K-Means to identify quality degrees provide good prediction rates since the number of outliers (identified through Isolated Forests) are mean. From the obtained rules-model, we can predict finely the different loading times.

For websites whose loading times vary regularly, we have applied both methods, i.e the usage of Random Forests where quality degrees are identified by the K-Means algorithm solely versus decision Trees where the numbers of states are deduced from HDP-HMM with GMM emissions distributions. The quality correctness prediction rates of these 14 websites are illustrated in Table 30. Together with the prediction success rates, we also expose the number of clusters identified from the K-Means algorithm and number of states identified by HDP-HMM with GMM emissions distributions.

Websites are all different among them, composed of content of different flavors which are delivered by web servers from all around the globe. For some websites, especially those belonging to the *News* and *Shopping* category, the usage of both prediction techniques do not increase a lot the success in quality prediction since content is renewed very regularly, being served at different times of the day by different CDNs or origin servers. An example of such particularity is the website *amazon.com* where the classic correctness prediction rate through Random Forests and K-Means is 63.89% and with the use of decision Trees coupled to HDP-

HMM with GMM, the prediction correctness rate is 72.98%. Although the use of HDP-HMM with GMM detects 13 states (7 clusters with K-Means), the correctness in prediction is only increased by 9.09%.

The accuracy of our solution is focused on Web View probes in Europe and when a main web page is in Asia, 87% of the time, the network receiving time is the main parameter causing fluctuations in the loading times. On a general perspective, from Table 30, the use of HDP-HMM with GMM instead of defining the number of clusters from the Elbow method do help in decreasing the error prediction rates. But for some cases, e.g *twitch.tv*, the web pages change very often and also the number of states identified is important, decision Trees rules are not enough to predict correctly the corresponding perceived web browsing quality for the entire web page.

5.3 Conclusion

Web pages' loading times may fluctuate upon a wide set of factors and these loading times may belong to different satisfaction degrees. It is important to identify why and which parameters contribute to different ranges of loading times, along with the thresholds of these parameters.

In order to predict the perceived web browsing quality of the visible portion of web pages, we have used Decision Trees to identify these parameters' thresholds and from the obtained rules-based model performed web browsing quality prediction on never assessed websites where 90.4% of the overall new dataset was correctly predicted. Generally speaking, when taking into account the visible portion at first glance of web pages, TFVR timings do not fluctuate regularly. Our model has proven to be correct for this configuration but it has to be adapted to other situations, e.g end-user's geographic location. Parameters to be re-assessed will be the Round-Trip-Time (path taken by network packets will be different for an end-user connected to a specific network operator at a specific geographic location), number and types of objects (mainly due to advertising), the number of domains (specifically the location of web servers delivering content), time of the day and corresponding network access (the throughput offered by network operators from different regions are different).

When the same process is used to predict the entire web page quality, the above method works well if and only if websites do not have important and regular fluctuations in their PLT time. These fluctuations are generally created by different types of web servers delivering content from different geographic locations. Different servers are involved since the content is regularly renewed for some websites. When websites are known to renew content very often and fluctuations happen in the PLT, making use of HDP-HMM with GMM emissions distributions helps in identifying an infinite number of states which may be hidden to classical segmentation techniques. Thanks to the application of HDP-HMM, we have been able to enrich identified rules-based models and increase the correctness in predicting the quality for the entire web page.

The Web browsing eco-system is complex and relying on a unique method to predict the quality during different phases of the loading progression may provide bad quality prediction. The quality prediction when making use of K-Means to identify quality degrees and decision trees works well most of the time for the First Paint, Time for Full Visual Rendering or Above-The-Fold time lapse. When focusing on the quality prediction of the visible surface area of Web pages, our rules-based model can be applied on any website and underlying correctness in quality prediction is good. When focusing on the quality prediction for the entire Web page, our rules-based model can be applied on any web page if and only if fluctuations in loading times are mean. But if important fluctuations happen, a specific set of rules for every web page is needed.

Part III

Conclusion

Conclusion

In this dissertation, our work has dealt with characterizing and quantifying web browsing quality along with the prediction of the underlying quality. The global Internet traffic has continuously been increasing over the last decade where the digital transformation of our society is accentuated by Web browsing. In a world where connectivity is of prime importance, it is important to better understand the Web browsing eco-system and hence enlighten end-users, network operators, and the research community how perceived Quality of Experience can be leveraged and Quality of Service increased.

When an end-user performs a Web browsing activity, we have three main boundaries. Firstly there is the environment of the end-user, i.e the type of used web browser or the device computing capabilities. Secondly is the network operator to which the device is attached where a wide range of networking technologies, e.g ADSL, Wi-Fi or Fiber are available to offer increased bandwidth. Thirdly we have remote websites composed of various web pages offering a wide set of services thanks to new Web technologies. These web pages can be hosted by different types of web servers which deliver the content through different Internet protocols. Our study is two-fold; understand the implications of the different factors which can lead to an enhanced or decreased web browsing experience and be able to predict the Web browsing quality during the different phases of a web page loading progression.

Over the past years, the computing power of end-users' devices has been increasing and network operators have been offering increasing bandwidth to their customers. New Web technologies are used by Web developers to enhance the graphical representation of their website. In order to deliver content faster and closest to end-users, Content Delivery Networks have been introduced in the process. New Internet protocols, such as HTTP/2 and QUIC have also been introduced to reduce content delivery time and promote end-users' privacy. From past research studies, the Web browsing eco-system has been studied in a standalone boundary-manner. In order to provide fine-grained information on the Web browsing eco-system, we have studied it as a whole following a real end-users' environment.

With the aim to quantify and qualify Web browsing into the most end-user representative manner, we have studied different tools meant to measure Web browsing activity. While some tools do not use real updated web browsers, the offered functionalities are mean and do not allow a fine-grained assessment of the Web browsing eco-system. We have thus designed, developed and deployed a new user-representative measurement tool, Web View. The latter is deployed on end-user devices, e.g desktops and laptops, being connected to residential access networks and makes use of real updated web browsers. Any website can be measured on-the-fly with a wide set of input parameters such as type and version of the web browser, requested protocol, the use or not of local caching, viewports, etc. Eighty-four monitoring parameters are offered among which Web View exposes all available web metrics and sheds light on the factors implicated in content delivery, i.e location and types of web servers, Internet protocol distribution, types of downloaded objects, etc. All measurements are represented in real-time on a public visualization website for specific websites where end-users, service companies or researchers can better understand the Web browsing eco-system.

In our work, we have studied all available Web metrics meant to measure web pages' loading time, which correlates to Quality of Experience. We have been able to identify that actual web metrics are either not supported by available web browsers or do not take into consideration new Web technologies such as JavaScript usage or Progressive Web Applications. Following studies on end-users' behavior during Web browsing, psychological studies show that the visible portion of web pages at first glance should be primarily measured to better qualify Web browsing experience. To circumvent the inefficiencies of

actual Web metrics, we have introduced the Time for Full Visual Rendering which measures the time to load the visible portion of a Web page at first glance. This measurement technique makes use of web browsers' exposed networking logs to provide fine-grained loading times, as well as an overview of the rendering process. This technique has proven to be lightweight, calculated with a mean extra computational time and takes into account new Web technologies.

To better understand the Web browsing eco-system but also its evolution over years, more than 18 Trillion measurements have been performed over 2.5 years. Measurements' analysis has shown that over the past 15 years, web pages have shifted from being static to dynamic and the amount of content composing web pages has doubled. Although new Internet protocols, such as HTTP/2 and QUIC, have been introduced to reduce the download time of content and promote privacy, old Internet protocols coexist with them. Content Delivery Networks are more and more adopted by Web developers to deliver content closest to end-users which reduce the overall delivery time. We have also identified that although we are in a privacy era, main web pages promote security but services embedded into web pages provide a large amount of content into an unsecured manner.

When paying attention to which factors can impact Web browsing experience, we have identified that upon a network access to which an end-user is attached, web pages' loading times decrease proportionally when bandwidth increases, i.e moving from an ADSL to Fiber network access can reduce up to one third loading times. The use of Content Delivery Networks jointly with new Internet protocols can decrease the entire (or visible portion) web page loading time. For some web pages making use of new Web technologies such as JavaScript or Progressive Web Applications, we have also identified that web developers do not prioritize the delivery of content to be rendered into the visible portion of web pages which can have drawbacks on end-users' rendering perception.

From measurements performed by our tool over the last 2.5 years, we have made use of machine learning techniques in order to predict the perceived Quality of Experience. We have firstly paid attention to the prediction of the quality as per the visible portion of web pages. Along with newly estimated satisfaction degrees from our large dataset of measurements, we have used decision trees to identify through a rules-based model which parameters (with their thresholds) qualify the best these different satisfaction degrees. This rules-based model has then been used to predict the web browsing quality of the visible portion of web pages. On never measured websites, the correctness in quality prediction for the visible portion is 90.4% which proves that our model is efficient. Secondly we have focused our study on the prediction of web browsing quality for the entire web page (visible and non-visible part), where the correctness in prediction rates are more than 95% if and only if important fluctuations do not happen in the loading times. To circumvent fluctuations which can yield high prediction error rates, we have made use of the Hierarchical Dirichlet Process Hidden Markov Models (HDP-HMM) with a Gaussian Mixture Model (GMM) emissions distributions to detect these fluctuations and enrich our rules based model. For websites where fluctuations happen regularly, this solution has increased the correctness in prediction rates up to 25% for the entire web page. When focusing on the quality prediction of the visible surface area of Web pages, our rules-based model can be applied on any website and underlying correctness in quality prediction is good. When focusing on the quality prediction for the entire Web page, our rules-based model can be applied on any web page if and only if fluctuations in loading times are mean. But if important fluctuations happen, a specific set of rules for every web page is needed.

Based on our work where we have introduced a new automated Web browsing tool, a new Web browsing measuring technique, and analyzed measurements performed on a wide range of websites over 2.5 years, we draw the following conclusions from our work. Firstly, to

be able to qualify finely web browsing experience, the visible portion of web pages at first glance should be primarily measured through web metrics making use of web browsers' networking logs and taking into account new Web technologies. Secondly, the Web browsing eco-system must be studied as a whole since our work has shown that factors qualifying web browsing experience are strongly linked among them. Thirdly, web developers should favor the use of Content Delivery Networks offering content to end-users via new Internet protocols which can decrease web pages' loading times and strengthen customer experience which is the key to retention and loyalty. Last but not the least; with more than 1.7 Billion websites in 2019, monitoring all of them is not feasible and quality prediction through machine learning techniques have proven to be efficient. A unique method for quality prediction is hard to define since a web page goes trough different phases rendering phases. As per these loading phases different solutions should be privileged to take into account fluctuations of loading times.

Perspectives

Our work opens several perspectives in the field of Internet measurements. We firstly go through the potential of our tool to follow up the adoption and impact of new Web technologies on perceived QoE. Secondly we expose how rules-based models can be extended to other scenarios to better predict experience.

Following the field of Internet measurements, Web View depicts the overall real-time Web browsing eco-system through a public visualization website where measurements are performed round the clock with real updated web browsers. With the addition of Web View probes at different geographic position connected to different network operators, this can help end-users to have an overview of the perceived QoE and prefer the use of a specific web browser or move to a different network operator. Regarding network operators, they will be able to assess in real-time the perceived QoE of their customers and identify how they can increase their QoS. Web developers on the other end will also be able to assess improvements related to QoE when selecting different CDNs as well as modifying the web page structure of their website. Web View has been primarily designed to measure websites' homepages. The technologies used by our tool can also lead to qualify Web browsing when an end-user visits a set of web pages belonging to the website itself. This can highlight Web developers which content has to be downloaded in priority to increase QoE and at the same time highlight CDNs which content needs to be automatically stored in their caches to reduce their download time. It can also highlight different CDN providers on the added value of different architectures promoted by their competitors and adopt them or introduce new ones.

Web View is designed into different independent modules and video delivery monitoring can also be performed through the addition of video metrics. This can help in better identifying the factors which can enhance Web browsing experience when video delivery is involved. Regarding developers, they can choose different video encoding formats following the end-user's environment and network operator to speed up the delivery. With the introduction of 5G networks in 2020, our study can also be extended to assess the impact of increased bandwidth on Web browsing experience. This can help end-users, network operators and Web developers to better understand the impact of this new technology.

Being in a privacy era where encryption has an important role, the Domain Name System over HTTPS (DoH) has been lately introduced where public DNS infrastructures can be used rather than the network operator's one. As an example, an end-user located in Europe can choose to use a public DNS, e.g *Google* or *Cloudflare*, through its web browser. These servers might not be located in Europe and hence not favor the use of Content Delivery Networks being the closest to end-users. The underlying impact should be studied to better understand if privacy can decrease Web browsing experience. With end-to-end encryption between end-users and web servers, Web browsing quality should also be profiled from encrypted traffic.

Following the prediction of experience, the construction of our rules-based models has shown that following different scenarios, the prediction of experience can be correctly performed. The construction of rules-based models through the clustering of information and decision trees can be used on any dataset where fluctuations are mean. When fluctuations happen in datasets through time, the use of HDP-HMM can be used to identify an infinite number of states and hence enrich rules-based models. Following the type of data, different models can be used along with the HDP-HMM, e.g Bayesian Mixture Models. HDP-HMM can also be used to detect degradations in different datasets where the number of outliers over time are important, e.g predicting electricity consumption at peak hours. When combined with decision trees, degraded states may also be identified and predicted.

Thesis Publications

International Journals

- **A. Saverimoutou**, B. Mathieu and S. Vaton, "A 6-month analysis of factors impacting web browsing quality for QoE prediction," *Computer Networks,* vol. 164, p. 106905, 2019.

- **A. Saverimoutou**, B. Mathieu and S. Vaton, "Web View: A Measurement Platform for Depicting Web browsing Quality and Delivery", in *IEEE Communications Magazine Series*, COMMAG 2020.

International Conferences

- **A. Saverimoutou**, B. Mathieu and S. Vaton, "Which secure transport protocol for a reliable HTTP/2-based web service: TLS or QUIC?" in *2017 IEEE Symposium on Computers and Communications, ISCC 2017, Heraklion, Greece, July 3-6, 2017*, 2017.

- **A. Saverimoutou**, B. Mathieu and S. Vaton, "Web Browsing Measurements: An Above-the-Fold Browser-Based Technique" in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, 2018.

- **A. Saverimoutou**, B. Mathieu and S. Vaton, "Web View: Measuring and Monitoring Representative Information on Websites" in *IEEE International Workshop on Quality of Experience Management, QOE-MANAGEMENT Paris, France, February 18, 2019*, 2019.

- **A. Saverimoutou**, B. Mathieu and S. Vaton, "Influence of Internet Protocols and CDN on Web Browsing" in *10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019, Canary Islands, Spain, June 24-26, 2019*, 2019 (**Best Paper Award**).

- Alexis Huet, **Antoine Saverimoutou**, Zied Ben Houidi, Hao Shi, Shengming Cai, Jinchun Xu, Bertrand Mathieu, Dario Rossi, "Revealing QoE of Web Users from Encrypted Network Traffic", in *International Federation for Information Processing (IFIP), Networking 2020, Paris, France, June 22-25, 2020.*

Other Publications

- **A. Saverimoutou**, B. Mathieu and S. Vaton, "Real-Time Monitoring and Troubleshooting of Web Browsing Sessions", RIPE NCC Blog June 2018.

Part IV

Appendix

Appendix A

Résumé en Français

Contenu

La navigation Web est l'un des principaux services de l'Internet où un large éventail d'acteurs est impliqué et évolue de manière constante. Pour mieux comprendre la Qualité d'Expérience (QoE) perçue par les utilisateurs, il est donc essentiel d'identifier comment le contenu des pages est constitué et délivré et de fournir une métrique de QoE pertinente. Dans cette thèse, nous avons conçu un nouvel outil, Web View, destiné à effectuer des sessions de navigation Web automatisées et mesurant de nombreuses informations. Nous avons aussi définit une nouvelle métrique Web, le *Time for Full Visual Rendering* (TFVR). À partir de plus de 18 trillions de mesures effectuées pendant 2,5 ans sur les 10,000 sites Web les plus visités (selon la classification Alexa), nous avons utilisé des techniques de statistiques pour identifier les facteurs clés qualifiant et quantifiant la qualité de la navigation Web. Cet ensemble de facteurs a été confirmé par un processus d'apprentissage automatique, donnant en sortie un ensemble de règles pour prédire les temps de chargement des pages Web. L'évaluation de notre modèle basé sur un arbre de décision sur des sites Web jamais mesurés montre que nous pouvons prédire correctement la qualité de la navigation Web perçue par les utilisateurs. Pour certains sites Web où les temps de chargement de la page Web entière fluctuent régulièrement, nous avons utilisé des chaines de Markov cachées qui permettent d'enrichir nos règles de décision et ainsi augmenter le taux de prédiction de la QoE. Ces travaux visent aussi à permettre aux opérateurs de réseaux et aux fournisseurs de services d'augmenter la Qualité de Service (QoS) offerte à leurs clients.

Contexte

Au cours de la dernière décennie, le trafic Internet global a augmenté de plus de 40% et les utilisateurs de l'Internet représentent à l'heure actuelle 57% de la population mondiale. Un grand nombre de services sont offerts aux utilisateurs à travers un grand nombre de sites Web. Afin d'offrir une Qualité d'Expérience (QoE) augmentée aux utilisateurs, il est important de mieux comprendre comment sont composées ces pages Web, tout en identifiant comment le contenu est délivré aux utilisateurs. La navigation Web possède un écosystème complexe où sont impliqués le périphérique de l'utilisateur, le type de navigateur, l'accès réseau, les protocoles de l'Internet, les serveurs Web délivrant le contenu et finalement les pages Web [1].

Le contenu est téléchargé via le protocole HTTP (*HyperText Transfer Protocol*) [32], plus particulièrement via HTTP/1.1 (*HyperText Transfer Protocol Version 1*) [5], HTTP/2 (*HyperText Transfer Protocol Version 2*) [6] et plus récemment, QUIC (*Quick UDP Internet Connections*) [7] qui est en cours de standardisation en HTTP/3 (*HyperText Transport Protocol Version 3*) à l'IETF (*Internet Engineering Task Force*). Les fournisseurs de services délivrent le contenu au plus proche des utilisateurs à travers des serveurs caches [8], CDNs (*Content Delivery Networks*) [9] et architectures à base de proxy [10].

Le nombre d'acteurs impliqués dans la navigation Web n'a cessé d'augmenter et d'évoluer et cela peut diminuer ou augmenter la QoE perçue des utilisateurs. Tandis que certaines études sont consacrées à identifier les profils d'utilisation du Web [13] [14] [15] [16] [17], d'autres études se focalisent sur l'apport des nouveaux protocoles de l'Internet [18] [19] [20] [21]. Le processus de *caching* des contenus [22] [23] contribue aussi à l'amélioration de la qualité d'expérience des utilisateurs. De nouvelles technologies Web [25] [26] ont aussi été introduites dans le but d'améliorer le téléchargement de contenus.

Problématique

La qualité perçue [29] [30] [31] lors de la navigation Web est la plupart du temps corrélée au temps nécessaire pour afficher une page Web (ou certaines parties de la page). Une augmentation dans le temps de chargement de certaines pages peut mener les utilisateurs à abandonner la page visitée et ainsi créer une baisse de revenus pour les sociétés de services. L'importance d'identifier les paramètres contribuant à une QoE améliorée peut être qualifiée à deux niveaux. Premièrement cela peut aider les développeurs de pages Web à améliorer les temps de chargement de leurs pages et ainsi augmenter la fidélité de leurs clients. Deuxièmement, ces paramètres peuvent être utilisés comme valeurs de références et permettre aux fournisseurs de services d'adapter leur Qualité de Service (QoS).

La navigation Web est activement étudiée par le monde de la recherche mais un intérêt particulier est consacré à l'impact des périphériques des utilisateurs, les types de navigateurs utilisés, l'apport des nouveaux protocoles de l'Internet ou encore l'impact des dégradations sur les utilisateurs finaux. Ces travaux de recherche ont permis de mieux cerner la complexité de la navigation Web mais comportent un grand nombre de limites que nous détaillons ci-après. Les navigateurs Web sont mis à jour régulièrement (en moyenne tous les 20 jours) et les études réalisées dans le passé prennent en compte un nombre limité de navigateurs. Lors de la navigation Web, un grand nombre de contenus est téléchargé de serveurs Web répartis tout autour du globe et les études effectuées qui sont orientées vers l'apport de nouveaux protocoles de l'Internet, ne prennent pas en compte le taux de réception de ces protocoles. Les différentes études ne prennent pas en compte les types de serveurs délivrant le contenu, qui sont très souvent sources d'une dégradation; l'utilisation de réseaux de diffusion de contenus n'est pas toujours privilégiée. Les études réalisées sont en moyenne effectuées sur une durée de 3 mois, ce qui n'est pas suffisant pour suivre de façon objective l'évolution du Web. Pour étudier la qualité de la navigation Web, plusieurs outils peuvent être utilisés mais leurs fonctionnalités sont limitées et ne permettent pas de tenir compte de l'écosystème du Web dans son intégralité.

Contributions de la thèse

Afin de répondre aux problématiques ci-dessus, nous avons étudié l'écosystème du Web dans sa globalité. Premièrement, nous avons conçu et déployé un nouvel outil automatisé de navigation Web (Web View) qui permet d'identifier tous les acteurs impliqués lors de la navigation Web. Les mesures effectuées par les sondes sont représentées en temps réel sur un site Web public (https://webview.orange.com). Deuxièmement, face à l'inefficacité des métriques Web actuelles, nous avons défini une nouvelle métrique web, le *Time for Full Visuel Rendering* (TFVR) faisant abstraction des types et versions de navigateurs, et permettant ainsi de mesurer finement la qualité d'expérience des utilisateurs lors du chargement de la surface visible d'une page Web. Troisièmement, nous avons appliqué des techniques de statistiques sur les mesures effectuées pendant 2,5 années ; ce qui nous a permis de mieux comprendre l'écosystème du Web mais aussi d'identifier les paramètres qui peuvent contribuer à une augmentation ou dégradation de la QoE. Enfin, nous avons utilisé des techniques issues de l'Intelligence Artificielle pour représenter œs paramètres sous une forme de règles. Ces dernières permettent de prédire la qualité d'expérience perçue par les utilisateurs pour le téléchargement de la surface visible et entière des pages Web.

A.1 Acteurs de l'écosystème de la navigation Web

Un grand nombre d'acteurs sont impliqués lors de la navigation Web. Nous les introduisons, soulignons comment ils ont évolué ces dernières années et les différentes études qui leurs sont consacrées pour identifier leur apport à la qualité de la navigation Web.

Protocoles Internet

Les protocoles de l'Internet sont utilisés pour demander et télécharger le contenu à afficher dans une page Web à partir de différents serveurs Web. Le protocole HTTP/1.1 (*HyperText Transfer Protocol Version 1*) [5] a été introduit en 1999 afin qu'une connexion puisse être réutilisée pour plusieurs requêtes et est couplé au protocole TCP (*Transmission Control Protocol*). Ce protocole permet d'envoyer une requête (jusqu'à six connexions parallèles) avant même de revoir la réponse. Une problématique révélée par le protocole HTTP/1.1 est que les réponses doivent être reçues dans l'ordre demandé, ce qui augmente la latence. Le protocole HTTP/2 (*HyperText Transfer Protocol Version 2*) [6] couplé à TCP a été introduit en 2015 avec deux axes d'amélioration majeurs : la rapidité et sécurité de la navigation. Parmi les améliorations on peut mentionner la compression d'entêtes des requêtes et des réponses, le multiplexage des requêtes au serveur, et le fait qu'un serveur Web puisse pousser automatiquement les ressources nécessaires sans que le navigateur les demandent. Le protocole QUIC (*Quick Internet UDP Connections*) [7] a été introduit en 2016, couplé à UDP (*User Datagram Protocol*), et est en phase de standardisation à l'IETF (*Internet Engineering Task Force*). Le protocole QUIC fonctionne sur UDP et permet de réduire la latence, chaque segment de données d'une connexion reçoit son propre numéro de séquence, les paquets perdus ne posent pas de gros problèmes grâce à un simple système de correction d'erreurs et le protocole possède un contrôle de surcharge. Pour améliorer la sécurité de transfert de données, le protocole TLS (*Transfer Layer Security*) est utilisé actuellement par HTTP/2-TCP et la version 1.3 sera utilisée par HTTP/3-UDP.

Les protocoles de l'Internet ont évolué sans cesse ces vingt dernières années avec pour objectif de délivrer le contenu aux utilisateurs plus rapidement. Comme il est souligné à travers le Tableau A-1, le protocole HTTP/1.1 a permis de réduire (par rapport à HTTP/1.0) les temps de latence. HTTP/2 n'est pas déployé à grande échelle malgré sa standardisation en 2015, mais permet de réduire les temps de téléchargement de contenus. QUIC est en phase de standardisation et est actuellement déployé majoritairement sur les serveurs de *Google*. QUIC permet de réduire la latence car fonctionne sur UDP et met en avant la vie privée de l'utilisateur grâce à un chiffrement des échanges de bout-en-bout.

Protocoles	Étude	Littérature	Résumé	
			Avantages	**Inconvénients**
HTTP/1.0	Mécanisme et efficacité	[32] [50] [51]	- Permet d'échanger des informations entre un navigateur et serveur	- Latence élevée - Connexion fermée après chaque requête / réponse - Augmente le temps de téléchargement
HTTP/1.1	Mécanisme et efficacité	[5] [51] [52]	- Envoyer une requête avant de recevoir la réponse - Six connexions parallèles - Réduit la latence	- Latence toujours considérable (TCP) - Impacte le temps de téléchargement - Connexions non-sécurisées
HTTP/2	Mécanisme, taux d'adoption et efficacité	[6] [19] [52] [33] [35] [34] [35] [36] [53] [54] [20]	- Réduit le temps de téléchargement - Priorisation des flux - Mécanisme de contrôle des flux - Compression des *headers* - Pousser le contenu automatiquement - Sécurité bout-en-bout	- Taux d'adoption moyen - Latence toujours considérable (TCP)
QUIC (HTTP/3)	Mécanisme, taux d'adoption et efficacité	[7] [37] [38] [20] [21] [39] [40] [55] [37] [41]	- Latence réduite (UDP) - QoE amélioré en environnement à forte latence - Sécurité bout-en-bout	- Principalement déployé sur les serveurs *Google* - Taux de déploiement faible (hormis *Google*)
Sécurité de la couche transport	TLS 1.2 / 1.3	[45] [46] [47] [48] [49]	- TLS 1.3 réduit la latence - Sécurité bout-en-bout	- Faible taux d'adoption

Tableau A-1 : Etudes liées aux protocoles de l'Internet

TLS 1.3 a été standardisé en 2018 et est plus robuste que TLS 1.2 mais très peu déployé actuellement.

Serveurs Web

Lors de la navigation Web, un grand nombre de ressources est téléchargé à partir de plusieurs serveurs Web. Ces serveurs peuvent être les serveurs d'origine, des caches ou CDNs (*Content Delivery Networks*). Le but principal des caches et CDNs est de délivrer le contenu au plus proche des utilisateurs, et ainsi réduire les temps de chargement de pages Web. Tandis que les caches sont destinés à délivrer du contenu statique, c.à.d. du texte ou des images, les CDNs peuvent délivrer tous types de contenus, c.à.d. des images, de la vidéo, des scripts, etc. Les CDNs fonctionnent selon plusieurs architectures où le contenu sera poussé automatiquement par les serveurs d'origine (*Push* CDNs), le CDN récupère du serveur d'origine le contenu suivant le temps de vie des différents objets (*Pull* CDN) et dispose d'un mécanisme de *Pull*, *Push* et stockage. Les CDNs et caches peuvent réduire la latence, les temps de chargement de pages web et la consommation de bande passante [9] [23] [57] [58] [59] [60]. Les CDNs ont été grandement utilisés dans le passé pour délivrer du contenu statique mais avec l'évolution constante du Web, des contenus dynamiques sont aussi délivrés [63]. Les caches [64] délivrent principalement des contenus statiques.

Métriques Web	Littérature
PLT	[69] [70] [71] [72] [73] [27] [28] [11] [12] [74] [75] [54] [76] [52] [77] [78] [79] [80] [81] [82] [55] [20] [21] [35] [36] [83] [84] [85]
Resource Timing	[69] [70] [71] [27] [76] [83] [84] [85]
Paint Timing	[69] [27] [76] [81] [83] [84] [85] [75]
SpeedIndex	[69] [76] [81] [83] [85] [75]
RUM SpeedIndex	[69] [83] [84]
ATF	[27] [81] [83] [84] [86] [85]

Tableau A-2: Métriques Web utilisées par le monde de la recherche

Pages Web

Les pages Web sont composées d'objets de types différents. Lors de la navigation Web, une page principale est premièrement téléchargée du serveur d'origine et est compilée par le navigateur. Lors de la phase de compilation, les octets sont convertis en caractères, jetons, nœuds, pour finalement obtenir le DOM (*Document Object Model*). Une page Web peut aussi être composée de plusieurs feuilles de styles (*Cascading Style Sheets*) qui sont téléchargées et compilées pour obtenir un CSSOM (*Cascading Style Sheet Object Model*). Pour afficher la page Web correspondante dans le navigateur, ce dernier utilise le DOM et CSSOM.

Métriques Web

Pour pouvoir mesurer les temps de chargement de pages Web de façon uniforme, les organismes de standardisation comme le W3C (*World Wide Web Consortium*) en collaboration avec des sociétés de services ont défini des métriques Web.

Le *Navigation Timing API* [65] met en avant le *Page Load Time* (PLT) qui est le temps de chargement d'une page Web entière. Le PLT mesure le temps pour télécharger tout le contenu nécessaire pour télécharger une page Web dans son intégralité et aussi le temps nécessaire pour la création du DOM et CSSOM. Le PLT est la métrique la plus utilisée pour qualifier la QoE des utilisateurs.

Le *Resource Timing API* [65] permet d'avoir des informations supplémentaires sur chaque objet téléchargé, c.à.d le type de l'objet, le protocole Internet par lequel il a été téléchargé, le temps nécessaire pour le télécharger, etc.

Le *Paint Timing API* [66] comprend quatre métriques: le *First Paint* (FP) qui est le temps pour afficher le premier pixel à l'écran, le *First Contentful Paint* (FCP) qui est le temps nécessaire au navigateur pour afficher une première image ou du texte, le *First Meaningful Paint* (FMP) qui est le temps nécessaire pour afficher dans la surface visible le texte et les images, et finalement le *Time To Interactive* (TTI) qui est le temps nécessaire pour afficher la partie visible de la page Web et que l'utilisateur puisse interagir avec elle.

Le *SpeedIndex* [67] est un score représentant le temps nécessaire pour afficher (sans scroller) la surface visible de la page Web. Le *SpeedIndex* est principalement calculé en effectuant une vidéo lors de la phase de chargement de la surface visible, convertie en images et en analysant les pixels pour identifier le moment où la surface visible est intégralement affichée.

Le *RUM-SpeedIndex* (*Real User Monitoring SpeedIndex*) permet de calculer le *SpeedIndex* en utilisant différents métriques W3C citées juste avant.

Le *Above-The-Fold* (ATF) [86] est une métrique utilisée pour mesurer le temps de chargement de la surface visible (sans scroller) d'une page Web. Le calcul de cette métrique peut se faire en effectuant une vidéo de progression de chargement de la partie visible de la page (comme pour le SpeedIndex) ou en utilisant les métriques W3C (comme pour le RUM-SpeedIndex).

Pour mieux qualifier la qualité de navigation Web, plusieurs travaux de recherche ont utilisé ces métriques Web, comme il est indiqué dans le Tableau A-2. Parmi ces métriques web, le PLT est la métrique la plus utilisée, mais cette métrique bien qu'elle soit

standardisée, est implémentée de différentes manières par les concepteurs de navigateurs, c.à.d. *Google, Mozilla, Opera*, etc. Avec l'introduction de nouvelles technologies Web, l'objectivité du PLT est contesté [27] car cette métrique ne s'est pas adaptée. De plus, d'après un grand nombre d'études réalisées pour mieux comprendre l'activité des utilisateurs lors de la navigation Web [89] [14] [90] [15] [91] [92] [93] ou leurs comportements [94] [16] [95] [96] [97], la mesure du temps de chargement de la partie visible (sans scroller) d'une page Web doit etre privilégiée [77].

	Outil	**Plugin**		**Navigateur Web utilisé**				**Ecosystème**	**Mécanismes Web**	**Qualité Web**	**Vie privée Web**	**Comportement utilisateurs**
		Chrome	Mozilla	Chrome	Mozilla	Moteur	Internet Explorer					
AdFisher [107]	✓				✓						✓	
Chameleon Crawler [101]	✓			✓				✓				
CoLab [108]	✓								✓			
FourthParty [99]			✓					✓			✓	
FPDetective [98]	✓			✓				✓			✓	
Gaze [75]	✓			✓						✓		✓
InspectorGadget [63]			✓		✓			✓				
OpenWPM [70]	✓			✓	✓			✓				
PageSpeed Insights [102]	✓			✓	✓					✓		
SiteSpeed.io [104]	✓			✓	✓		✓			✓		
WebPageTest [103]	✓			✓			✓			✓		
WebXRay [106]	✓					✓					✓	
WProf [54]	✓			✓						✓		
XRay [100]			✓					✓			✓	
YSlow [105]		✓	✓							✓		

Tableau A-3: Outils pour étudier l'écosystème du Web

Outils d'automatisation pour étudier la navigation Web

Un grand nombre d'outils a été introduit dans le monde de la recherche dans le but d'étudier l'écosystème du Web. Tandis que certains outils se focalisent sur l'identification de paramètres utilisés par les fournisseurs de services [63] [98] [70] [99] [100] [101], d'autres sont orientés vers la mesure de la qualité de la navigation Web [54] [75] [102] [103] [104] [105] ou encore l'étude de la vie privée des utilisateurs [99] [106] [100] [107]. Pour mieux comprendre le comportement des utilisateurs lors des sessions de navigation Web, d'autres outils ont été introduits [75] [80] [15]. Le Tableau A-3 illustre les différents outils utilisés et le domaine étudié.

A.2 Un nouvel outil automatisé de navigation Web : Web View

Pour mieux comprendre l'écosystème de la navigation Web, un grand nombre d'outils existent. Ces outils sont destinés à étudier les configurations de serveurs Web, la vie privée des utilisateurs lors de la navigation sur des sites Web, le comportement des utilisateurs et mesure de la qualité de navigation Web. Parmi ces outils, un nombre limité d'entre eux s'intéresse à la qualité de la navigation Web.

Lors de notre phase de prise en main du sujet, nous avons utilisé ces outils destinés à mesurer la qualité de la navigation Web et avons pu remarquer que les fonctionnalités offertes sont limitées et ne permettent pas d'étudier la navigation Web dans son intégralité. Nous détaillons ci-après trois outils. L'outil *PhantomJS* pilote le moteur du navigateur *Chrome* pour accéder à différents sites Web. En utilisant uniquement le moteur uniquement, nous n'avons pas la représentation graphique du navigateur habituellement utilisée. De plus, les requêtes sont toutes envoyées aux différents serveurs Web via le protocole HTTP/1.1, ce qui engendre le téléchargement de contenus uniquement via HTTP/1.1 ; cet outil ne permet pas de suivre les évolutions et apports des nouveaux protocoles de l'Internet. De plus cet outil n'est plus maintenu depuis mai 2018. Un autre outil est *WebPageTest* qui offre un environnement de mesure plus réaliste, comparativement à *PhantomJS*, car l'outil utilise de vrais navigateurs Web. Les mesures peuvent être effectuées à partir de sondes réparties tout autour du globe. Mais l'outil *WebPageTest* n'offre pas la possibilité d'effectuer les mesures via un grand nombre de navigateurs et les sondes effectuant les mesures ne sont pas rattachées à des accès réseaux résidentiels.

L'outil *SiteSpeed.io* a été introduit en 2018, deux ans après le début de notre étude. Cet outil permet d'effectuer des mesures de qualité de navigation Web selon un environnement utilisateur réel, c.à.d. que l'outil peut être déployé sur des périphériques représentatifs de ceux des utilisateurs, étant rattachés à un accès réseau résidentiel. Cet outil utilise les logs réseaux offerts par les navigateurs pour calculer les différents indicateurs de qualité. Malgré l'environnement plus réaliste pour effectuer les mesures, plusieurs indicateurs doivent être calculés après avoir effectué les mesures. Cela ne permet pas de représenter en temps réel l'écosystème du Web.

A.2.1 Web View : Architecture, Infrastructure et Fonctionnalités

En 2017, face à l'inefficacité des outils existants ne permettant pas d'étudier l'écosystème du Web dans son intégralité, nous avons introduit un nouvel outil de mesure automatisé, Web View[42] [69] [121]. Cet outil est déployé sur des ordinateurs de bureau ou portables et utilise des navigateurs Web réels. Nous privilégions des réseaux d'accès résidentiels pour mieux représenter l'environnement des utilisateurs. Web View permet de mesurer la qualité de navigation d'un site Web unique, une liste de sites Web prédéfinie ou les sites Web Alexa[43] représentant les pages Web les plus visitées au quotidien. Web View met en avant toutes les métriques web définies par le W3C mais effectue aussi leur calcul à partir des logs réseaux des navigateurs. Notre outil permet de mieux comprendre en temps réel l'écosystème du Web en offrant diverses informations sur le protocole Internet par lequel les objets sont téléchargés, les types et nombres d'objets, les types de serveurs Web délivrant le contenu et aussi leurs localisations géographiques. Web View offre aussi un site Web de surveillance[44] où sont représentées les mesures effectuées par les sondes réparties en Europe, Asie ou Afrique.

[42] L'outil dans ses débuts de développement et déploiement était nommé MORIS (*Measuring and Observing Representative Information on webSites*)
[43] https://www.alexa.com/siteinfo
[44] https://webview.orange.com

L'architecture de Web View

Web View est composé de six modules indépendants, qui sont destinés à configurer les paramètres des mesures, piloter des navigateurs Web réels, effectuer des calculs à partir des logs réseaux offerts par les navigateurs et finalement représenter en temps réel ces mesures sur un site Web public. L'architecture de notre outil est illustrée par la Figure A-1.

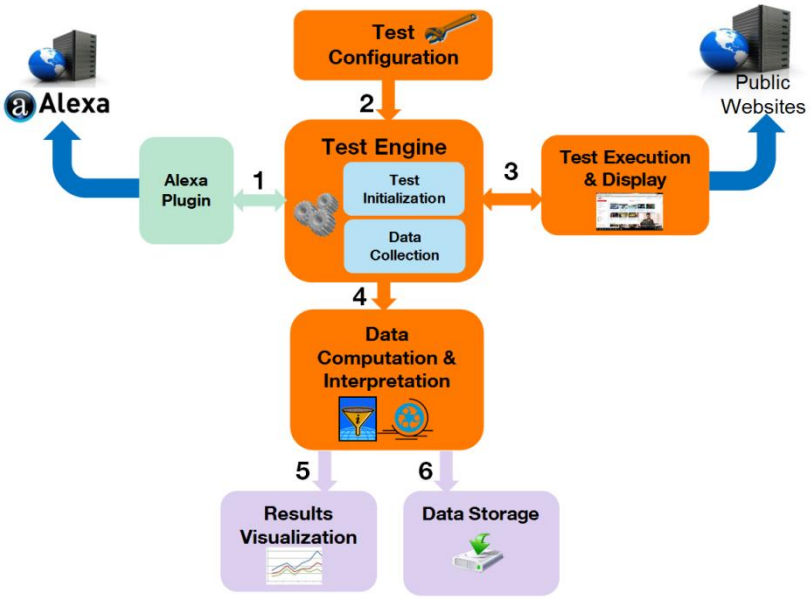


Figure A-1: L'architecture de Web View

Web View a été développé via le langage de programmation *Python* et utilise l'outil *Selenium*[45] qui permet de piloter les différents navigateurs Web. Lors de la phase de mesure, nous pouvons configurer le site Web à mesurer, le protocole Internet privilégié ou encore la taille de fenêtre du navigateur.

Type d'ordinateur	**Localisation**	**Opérateur Télécom**	**Bande passante descendante**		**Bande passante montante**	
Bureau et Portable	Lannion (France)	Orange	ADSL Wi-Fi FIBRE	10Mbps 200Mbps 800Mbps	ADSL Wi-Fi FIBRE	1Mbps 200Mbps 300Mbps
Bureau	Lannion (France)	Orange	ADSL FIBRE	10Mbps 800Mbps	ADSL FIBRE	1Mbps 300Mbps
Bureau	Vannes (France)	Free	FIBRE	800Mbps	FIBRE	300Mbps
Bureau	Paris (France)	Orange	FIBRE	1Gbps	FIBRE	300Mbps
Bureau	Paris (France)	Free	FIBRE	1Gbps	FIBRE	570Mbps
Bureau	Paris (France)	Bouygues	FIBRE	1Gbps	FIBRE	510Mbps
Bureau	Paris (France)	SFR	FIBRE	1Gbps	FIBRE	470Mbps
Bureau	Curepipe (Ile Maurice)	Mauritius Telecom	FIBRE	10Mbps	FIBRE	2Mbps
Cloud EC2-M5	Tokyo (Japon)	Amazon	10Gbps		4Gbps	

Tableau A-4: L'infrastructure de Web View

[45] https://www.seleniumhq.org/

L'infrastructure de Web View

Web View est composé d'une plateforme qui implique les sondes (Vitesse de processeur 2.5 Ghz et Mémoire 8Go), une base de données mutualisée et un site Web représentant les différentes mesures. Actuellement 17 sondes sont déployées, illustrées par le Tableau A-4. Chaque sonde mesure de façon continue les dix mille sites Web les plus visités en utilisant divers navigateurs Web rattachés à différents réseaux d'accès (ADSL, Wi-Fi et Fibre) selon plusieurs opérateurs et à divers endroits géographiques. Les différentes mesures sont automatiquement relayées à une base de données mutualisée et représentées sur un site Web public.

Les fonctionnalités de Web View

Web View permet d'effectuer de la navigation Web automatisée selon plusieurs paramètres pour étudier la navigation Web. Quatre fonctionnalités sont offertes pour effectuer des mesures :

- *lightDomains* : Effectuer des mesures de sites Web et mettre en avant différents temps de chargement, la localisation des serveurs Web, les taux de réponses selon différents protocoles de l'Internet et l'identification du type de contenu téléchargé.
- *detailedDomains* : Est complémentaire à la fonctionnalité *lightDomains* et permet d'identifier le type de serveur Web délivrant le contenu, c.à.d. serveur d'origine, caches ou CDN
- *networkPath* : Est complémentaire à la fonctionnalité de *detailedDomains* et permet d'estimer en temps réel le chemin montant suivi par les paquets réseaux entre le navigateur et leurs destinations.
- *dohMode* : Est complémentaire à *lightDomains* et permet de spécifier l'utilisation de DNS (Domain Name System) publics, e.x. *Google* ou *Cloudflare*

La configuration des mesures peut se faire selon différents paramètres en spécifiant le protocole Internet privilégié, le type et la version du navigateur, une fonctionnalité parmi les quatre citées au-dessus, l'utilisation (ou pas) de bloqueur de publicités, la taille de la fenêtre du navigateur, le type d'adressage (IPv4 ou IPv6) et finalement le(s) site(s) Web à mesurer.

Web View offre 84 indicateurs[46] de qualité qui mettent en avant les métriques W3C mais aussi leur calcul à partir des logs réseaux offerts par les navigateurs. Les résultats des mesures mettent aussi en avant une représentation fine du nombre et des types d'objets téléchargés[47]. Selon la fonctionnalité demandée lors de la mesure, nous retrouvons parmi les résultats l'identification des types de serveurs Web délivrant le contenu et leurs localisations géographiques. Si la fonctionnalité *networkPath* est demandée, des mesures de chemins réseaux pris par les paquets sont offertes.

A.2.2 Site Web de surveillance en temps réel

Les mesures effectuées par les sondes sont représentées en temps réel sur un site Web public (https://webview.orange.com). Ce site Web est destiné aux utilisateurs, fournisseurs de contenus ou au monde de la recherche afin de suivre l'évolution de certains sites Web. Toute personne peut y accéder et choisir les paramètres de mesures des sondes et suivre les temps de chargement, la distribution des protocoles de l'Internet (selon un protocole privilégié) comme le souligne la Figure A-2, la localisation des serveurs Web par continents ou encore les types d'objets téléchargés. Une deuxième page[48] permet d'identifier les différents types de serveurs Web selon leurs localisations géographiques par ville ou pays, délivrant le

[46] https://webview.orange.com/public/img/monitParam.png
[47] https://webview.orange.com/public/img/domainDetails.png
[48] https://webview.orange.com/d/UyIIcrUmz

contenu aux utilisateurs comme le montre la Figure A-3. Ce site Web est destiné à suivre l'évolution de la qualité de la navigation Web mais permet aussi d'identifier les dégradations de qualité en temps réel.

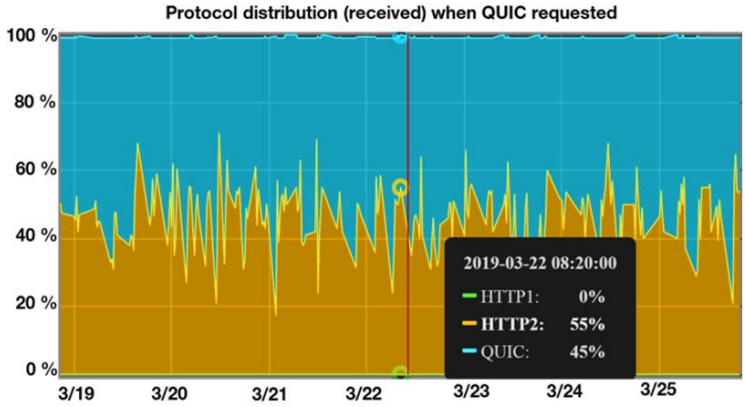


Figure A-2: La distribution des protocoles de l'Internet pour le site Web *youtube.com*

Figure A-3: Localisation et types de serveurs Web délivrant le contenu

A.3 Une nouvelle métrique Web : le *Time for Full Visual Rendering*

L'affichage d'une page Web se fait selon plusieurs étapes, c.à.d. téléchargement de la page principale, compilation par le navigateur, téléchargement de contenus additionnels et finalement affichage dans la fenêtre du navigateur Web. Afin d'évaluer la qualité perçue par les utilisateurs, un grand nombre de métriques Web ont été introduites pour mettre en avant le temps nécessaire pour afficher la page entière (ou certaines parties). Nous avons donc implémenté ces métriques Web dans notre outil Web View et à partir des mesures effectuées, nous avons pu identifier plusieurs inefficacités de ces métriques. Pour palier au manque d'objectivité de ces métriques, nous avons introduit une nouvelle métrique Web, le *Time for*

Full Visual Rendering (TFVR) qui est calculé à partir des logs réseaux du navigateur en temps réel. Cette métrique comparé à d'autres a démontré à être plus précise.

A.3.1 Inefficacités des métriques Web actuelles

Le *Page Load Time* (PLT) est une métrique permettant de mesurer le temps de chargement d'une page Web dans son intégralité. Elle est standardisée par le W3C, que nous référençons par PLT-W3C. Le PLT-W3C utilise le *Resource Timing API* aussi standardisé par le W3C et son implémentation dans les navigateurs n'offre pas des temps de mesures précis. Cela est principalement dû au fait que le *Resource Timing API* n'expose pas certaines informations liées au téléchargement de contenu si un serveur Web distant ne l'autorise pas. De plus avec l'introduction de nouvelles technologies Web comme le *Progressive Web Applications* (PWA), le PLT-W3C ne considère pas le téléchargement de différents objets qui sont nécessaires pour afficher la page Web correctement.

La métrique *Above-The-Fold* (ATF) est destinée à mesurer le temps de chargement de la surface visible d'une page Web. Cette métrique peut être calculée de deux façons ; l'utilisation de vidéos pour analyser la progression de chargement et l'utilisation d'informations offertes par les métriques du W3C. Dans le cas de l'utilisation de vidéos, des outils externes au navigateur doivent être utilisés et peuvent ainsi impacter la qualité des mesures. À partir des vidéos sont extraites différentes images qui sont ensuite comparées pour identifier si les pixels ne changent plus. Si tel est le cas, la dernière image où les pixels ne changent plus indique le temps pour afficher la surface visible. Le temps de calcul de l'ATF est important. Une autre façon de calculer l'ATF est l'utilisation des informations offertes par le *Resource Timing API* et comme nous l'avons indiqué précédemment, diverses informations sur les objets téléchargés ne sont pas offertes.

La métrique *Time-To-Interactive* (TTI) est le temps nécessaire entre le début de chargement d'une page Web et le moment auquel un utilisateur peut interagir avec la page. Cette métrique Web est implémentée (en 2019) uniquement dans le navigateur *Chrome* et est calculée si et seulement si aucune requête n'est effectuée par le navigateur ni aucune réponse reçue de différents serveurs Web pendant 5 secondes. Les pages Web sont de nos jours dynamiques et l'état du réseau rarement au repos, ce qui rend le calcul de cette métrique complexe.

Ces métriques Web ne sont pas efficaces pour fournir des temps de chargement précis car ils ne sont pas implémentés dans tous les navigateurs ou encore des outils externes pour effectuer des vidéos sont nécessaires. Face aux nouvelles technologies du Web, ces métriques ne mesurent pas finement les temps de chargement. Une nouvelle métrique Web est donc nécessaire pour mesurer finement des temps de chargement en utilisant les logs réseaux offerts par les navigateurs, tout en prenant en compte les nouvelles technologies du Web.

A.3.2 TFVR : Design, Mécanique et Efficacité

Le *Time for Full Visual Rendering* (TFVR) [83] est une métrique Web que nous avons définie et qui permet de mesurer le temps de chargement de la surface visible d'une page Web. Elle fait abstraction des types et versions de navigateurs et est entièrement écrite en langage de programmation *JavaScript*.

Design et Mécanique

Le calcul du TFVR se fait à partir des logs réseaux offerts par les navigateurs. Pour récupérer ces logs réseaux dans leur intégralité, nous avons développé deux *plugins* pour *Chrome* et *Firefox*, prenant en compte la complexité des pages Web dynamiques. A partir de ces logs réseaux couplés aux informations de temps de compilations offerts par le navigateur, nous identifions les objets qui sont affichés dans la surface visible du navigateur. Le temps de chargement de la surface visible est alors calculé en prenant en compte le temps pour

télécharger chaque objet ainsi que le temps nécessaire pour l'affichage. Lors de la phase de calcul du TFVR, nous identifions aussi pour chaque objet affiché dans la surface visible, son type, le protocole Internet par lequel il a été téléchargé et la localisation des serveurs Web distants, comme illustré par la Figure A-4.

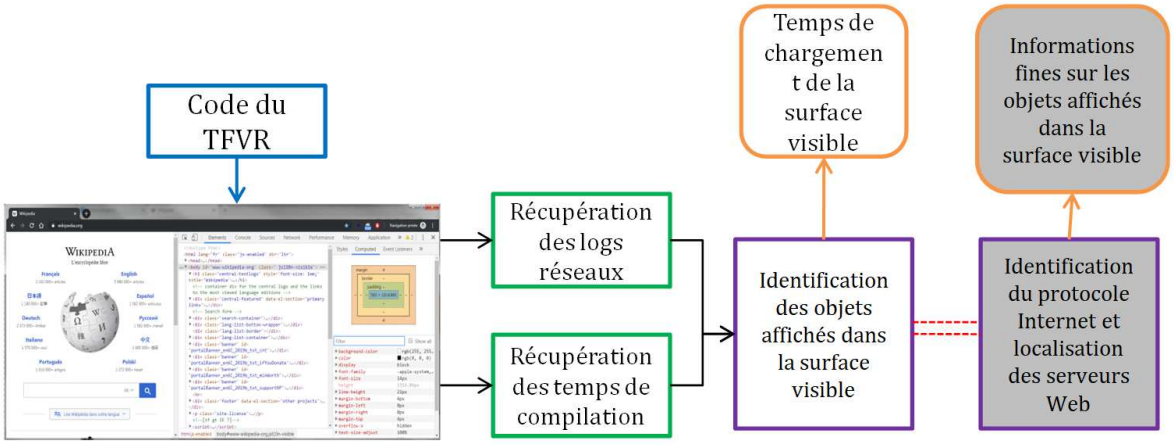
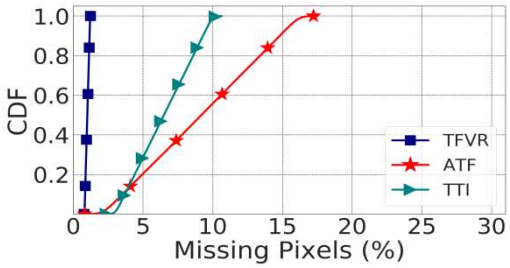
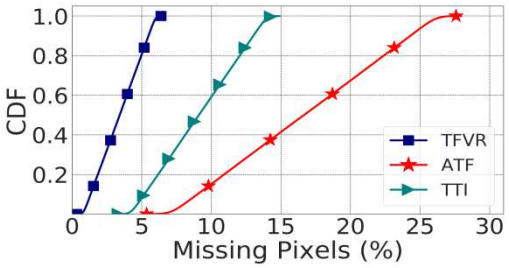


Figure A-4: Design and mécanique du TFVR

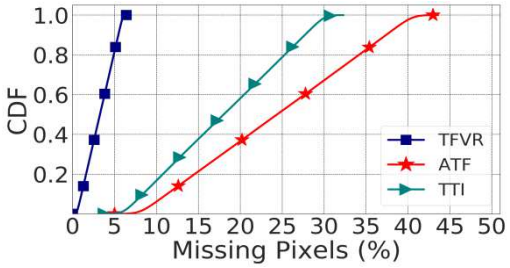
Le calcul du TFVR est effectué en temps réel selon un temps moyen de 0.156 secondes pour les 10,000 sites Web d'Alexa les plus visités. Cette métrique prend en compte les nouvelles technologies Web comme le *JavaScript* ou les *Progressive Web Applications*.



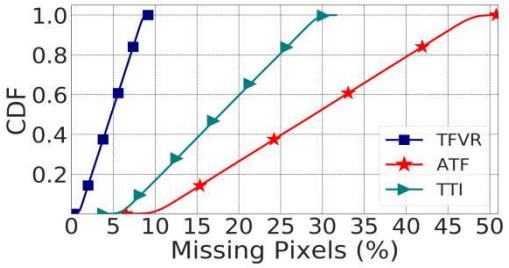
(a) Avec bloqueur de publicité and fenêtre de navigateur 768x1024



(b) Sans bloqueur de publicité and fenêtre de navigateur 768x1024

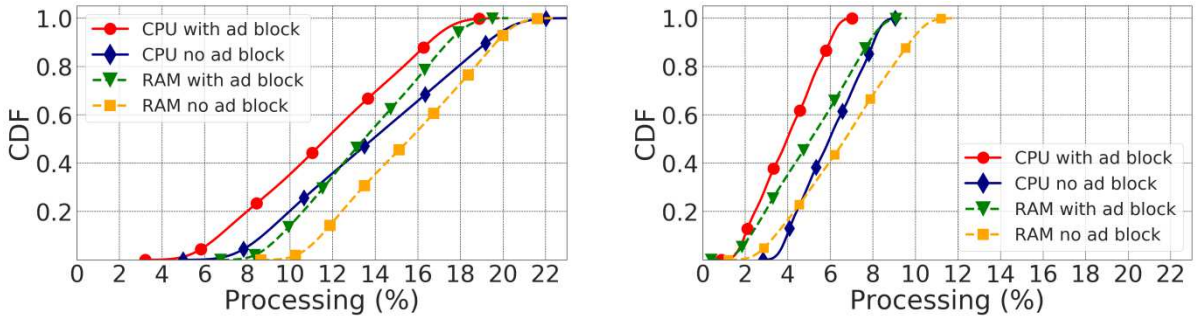


(c) Avec bloqueur de publicité and fenêtre de navigateur 1920x1080



(d) Sans bloqueur de publicité and fenêtre de navigateur 1920x1080

Figure A-5: Pixels manquants de la surface visible quand la page entière est affichée



(a) Calcul de l'ATF en utilisant des vidéos

(b) Calcul de l'ATF en utilisant les informations du navigateur

Figure A-6: Augmentation de puissance de calcul et mémoire comparé au TFVR

Efficacité

En comparant différentes métriques mesurant le temps nécessaire pour afficher la surface visible d'une page Web, la précision du TFVR est démontrée dans la Figure A-5 où moins de 1% de pixels sont manquants au moment où la page entière est affichée, tandis que pour le TTI 7.3% de pixels sont manquants et pour le ATF 10.6%. En calculant le temps nécessaire pour afficher la surface visible, nous avons remarqué que la métrique ATF consomme 8% de puissance de calcul et 9% de mémoire supplémentaires comparativement au TFVR, illustré par la Figure A-6. Comparativement à d'autres métriques, le TFVR prend en compte les nouvelles technologies Web et expose des temps de chargement fins. Le TFVR permet aussi d'identifier les ressources bloquantes qui doivent être affichées dans la surface visible.

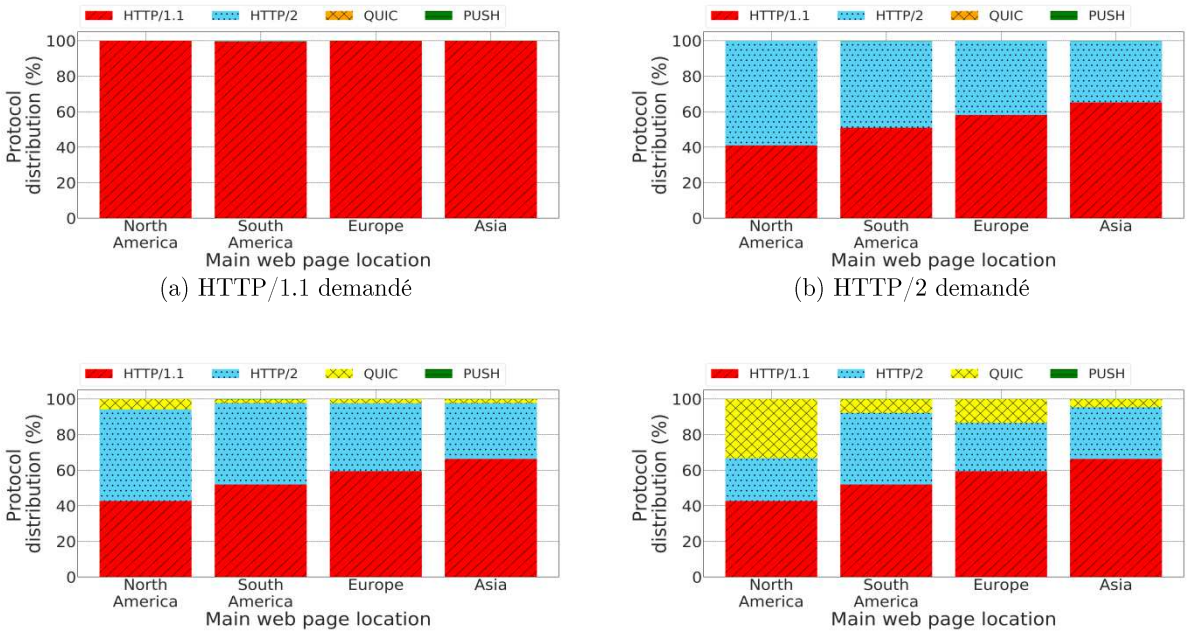
A.4 Caractérisation de la qualité de la navigation Web

Un grand nombre de facteurs est impliqué lors de la navigation Web, comme le navigateur Web utilisé, les types d'accès réseaux offrants différentes bandes passantes ou des bloqueurs de publicités et l'affichage de la page Web correspondante qui peut être statique ou dynamique. Le contenu est téléchargé à partir de plusieurs serveurs Web qui peuvent être des serveurs d'origine, des caches ou des réseaux de diffusion de contenus (CDN).

A.4.1 Ecosystème de la navigation Web

Grace aux mesures effectuées par notre outil Web View offrant 84 indicateurs de qualité pour chaque mesure, nous avons analysé 244 millions de mesures différentes via des techniques de statistiques pour mieux comprendre l'écosystème du Web. Les 10,000 sites Web les plus visités (selon Alexa) ont été mesurés à partir des sondes Web View localisées en Europe. Parmi les sites Web mesurés, 52.23% des pages principales sont délivrées par des serveurs Web en Amérique du Nord, 28.44% en Europe, 16.22% en Asie et 1.10% en Amérique du Sud.

A partir des études de la navigation Web effectuées en 2014, le nombre d'objets téléchargé a augmenté de 17% pour les sites Web classés entre le rang 1 et 2000 et augmenté de 31% pour les sites Web classés entre le rang 5000 et 10,000. Tandis que les images occupent une grande partie du contenu téléchargé, nous avons identifié que les pages Web sont composées en moyenne de 4 *css* (Feuille de styles), 5 scripts, 16 images et 2 *xml*. Nous avons aussi identifié que l'utilisation du nombre moyen de scripts et images pour ces sites Web a augmenté de 53% au cours des 15 dernières années [51] et de by 7% à partir des études récentes [11].



(a) HTTP/1.1 demandé

(b) HTTP/2 demandé

(c) QUIC demandé

(d) QUIC Repeat demandé

Figure A-7: Taux de distribution de protocoles Internet selon le protocole demandé

Quand la page principale est estimée d'être délivrée par des serveurs Web en Amérique du Nord et Europe, un plus grand pourcentage du contenu est délivré aux utilisateurs de façon sécurisée. Le protocole HTTP/1.1 a été standardisé depuis plus de 20 ans et est toujours utilisé par les serveurs Web pour délivrer le contenu aux utilisateurs, en particulier des serveurs Web localisés en Asie. Bien que le protocole HTTP/2 ait été standardisé en 2015, son déploiement est effectué à un rythme très lent et même si les utilisateurs utilisent des navigateurs Web récents, le contenu est téléchargé via le protocole Internet HTTP/1.1 et HTTP/2, comme illustré par la Figure A-7. En analysant les mesures effectuées en mars 2018 et comparées à celles effectuées en mars 2019, seulement 4% de contenus supplémentaires sont téléchargés en HTTP/2. Le protocole QUIC est principalement déployé sur les serveurs Web de *Google* et depuis le début de sa phase de standardisation en HTTP/3, les réseaux de diffusion majeurs ont commencé à l'implémenter.

A.4.2 Facteurs impactant la qualité de la navigation Web

Un indicateur de la qualité perçue lors de la navigation Web est le temps nécessaire pour afficher une page Web entière (ou certaines parties). Nous présentons dans cette sous-section les différents facteurs qui peuvent impacter ces temps de chargement. Premièrement nous analysons l'impact de la structure de la page Web et la catégorie à laquelle elle appartient. Ensuite nous analysons les facteurs impliqués dans le téléchargement des contenus et leurs impacts sur la qualité d'expérience. Finalement nous nous consacrons à l'environnement de l'utilisateur qui peut impacter les temps de chargement des pages Web.

Structures et catégories des pages Web

Les pages Web sont toutes différentes et même si nous les classifions par rapport à la localisation de la page principale, pour différentes métriques nous pouvons voir à partir de la Figure A-8, que les temps de chargement appartiennent à des tranches de valeurs très étendues. En comparant ces différentes catégories, nous remarquons que les temps de chargements moyens sont importants mais pour certaines très proches, comme illustré par la Figure A-9. L'élément principal contribuant à ces différences dans les temps de chargement

entre différentes catégories est le type des différents objets téléchargés à travers différents protocoles de l'Internet.

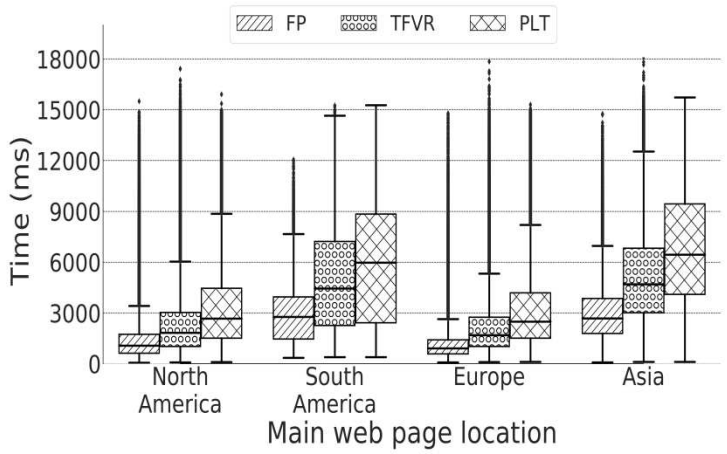


Figure A-8: Temps de chargement selon la localisation de la page principale

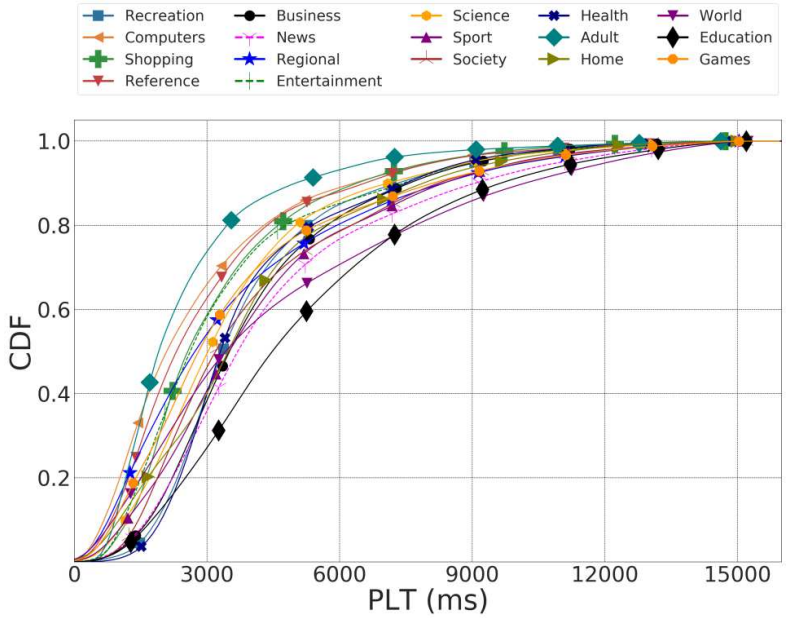


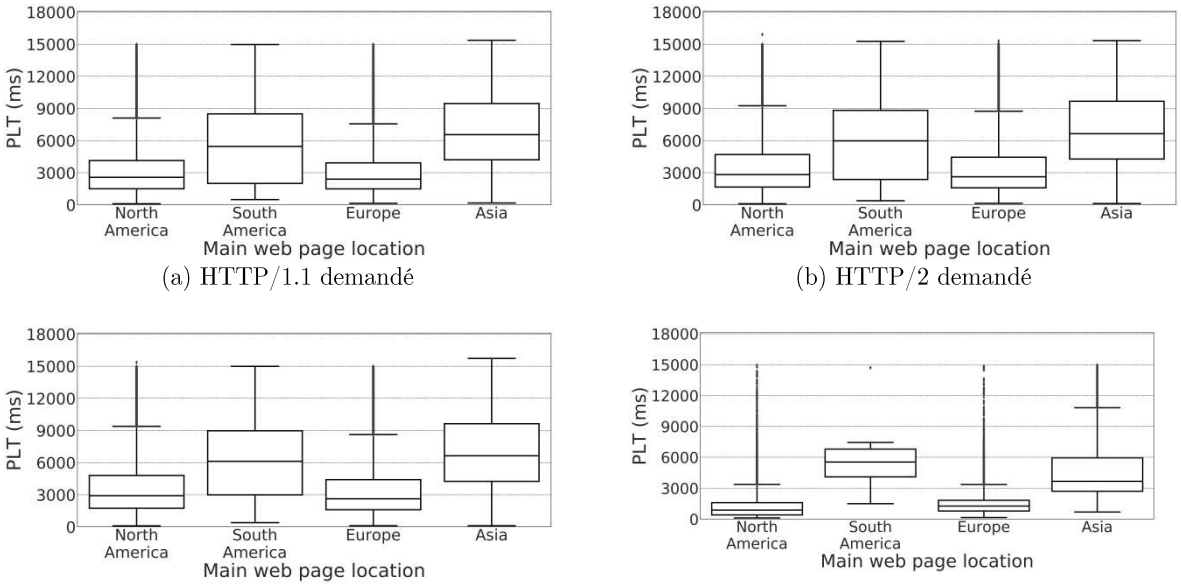
Figure A-9: Temps de chargement de la page entière selon catégories de sites Web

Facteurs impliqués dans le téléchargement de contenus

Tandis que privilégier le téléchargement de contenus à travers le protocole HTTP/1 .1, HTTP/2 ou QUIC ne permet pas de diminuer le temps de chargement de pages Web, l'utilisation du cache du navigateur est le seul élément permettant d'augmenter la qualité perçue par les utilisateurs, comme illustré par la Figure A-10. Même en privilégiant les nouveaux protocoles de l'Internet, les réponses sont généralement fournies selon différents protocoles de l'Internet qui coexistent.

Les pages Web offrent un grand nombre de services qui sont distribués par différents domaines. Le contenu téléchargé de ces domaines peut se faire à travers différents serveurs Web. Nous avons pu identifier que plus le nombre de domaines délivrant le contenu est élevé, plus les temps de chargement des pages Web sont élevés. Les réseaux de diffusion de contenus (CDNs) sont de plus en plus utilisés par les développeurs Web dans le but de délivrer le contenu au plus proche des utilisateurs. A partir de l'analyse de nos mesures [125],

nous avons identifié que si le contenu est délivré par des CDNs, le temps de chargement de l'intégralité d'une page Web peut être diminué jusqu'à 43.1% en demandant le protocole HTTP/2 et jusqu'à 38.5% en demandant le protocole QUIC. Plus d'un quart des contenus pour les 1,000 sites Web les plus visités sont délivrés par les réseaux de diffusion de contenus. Le taux d'utilisation de CDNs par des sites Web asiatiques est toutefois très bas, de l'ordre de 10%.



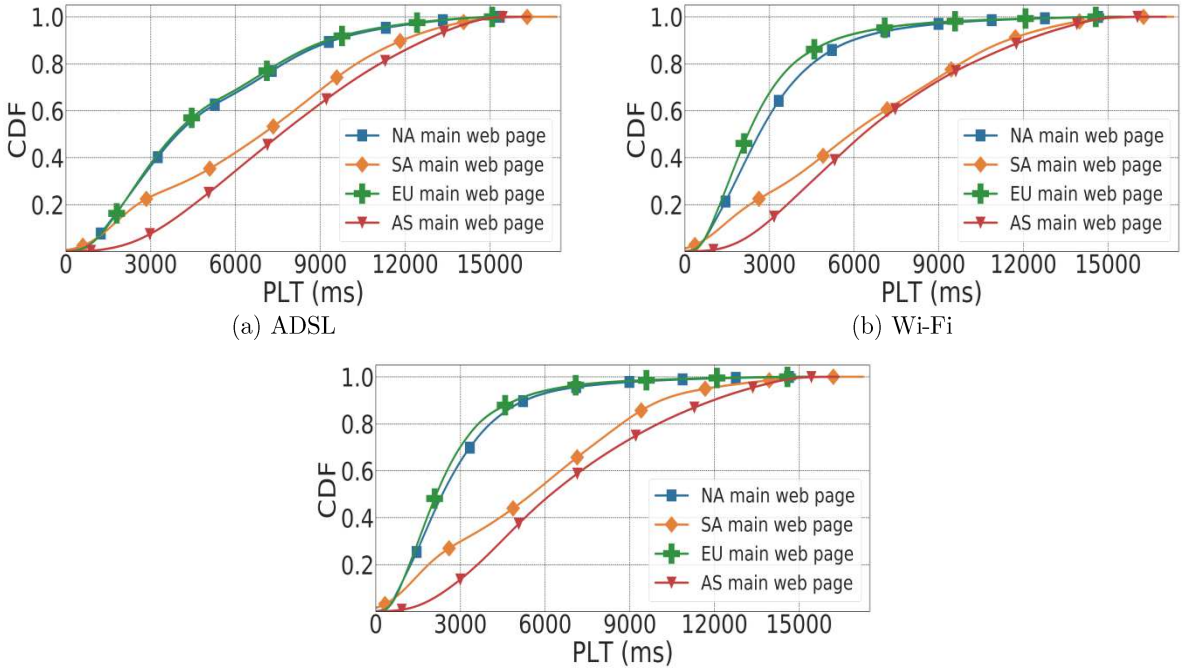
(a) HTTP/1.1 demandé

(b) HTTP/2 demandé

(c) QUIC demandé

(d) QUIC Repeat demandé

Figure A-10: Impacte du protocole Internet demandé sur le PLT



(a) ADSL

(b) Wi-Fi

(c) Fibre

Figure A-11: Temps de chargement de la page entière en fonction de la bande passante

Environnement de l'utilisateur

Nous nous sommes intéressés aux types d'accès réseaux résidentiels des utilisateurs qui peuvent offrir des bandes passantes plus importantes selon différentes technologies (ADSL, Wi-Fi et Fibre). En privilégiant le protocole Internet HTTP/2 ou QUIC, les temps de chargement de pages Web peuvent être diminués en moyenne de 19.73% en migrant de l'ADSL vers le Wi-Fi, et de 16.02% en migrant du Wi-Fi vers la Fibre. De façon générale, pour un utilisateur dont le périphérique est rattaché à un accès réseaux ADSL et demandant le protocole QUIC, la migration vers la Fibre peut diminuer les temps de chargement de pages Web visitées jusqu'à 30.25% (Figure A-11). L'heure à laquelle un utilisateur visite un site Web particulier peut aussi contribuer à diminuer la qualité d'expérience perçue qui est essentiellement due au taux de fréquentation du site Web mais aussi en lien avec l'état du réseau qui peut être chargé.

Paramètres	FP	TFVR	PLT
Localisation de la page principale	✓	✓	✓
Protocole Internet demandé		✓	✓
Localisation des serveurs		✓	✓
Temps de réponse entre navigateur et serveurs	✓	✓	✓
Catégorie du site Web	✓	✓	✓
Nombre d'objets téléchargés	✓	✓	✓
Taille de la page principale	✓	✓	✓
Taille des objets		✓	✓
Type des objets		✓	✓
Pourcentage visible de la page (sans scroller)			✓
Nombre de domaines		✓	✓
Utilisation de bloqueurs de publicités		✓	✓
Heure de la journée	✓	✓	✓
Type d'accès réseau	✓	✓	✓

Tableau A-5: Paramètres impactant la qualité de la navigation Web

De manière générale, le Tableau A-5 met en avant les paramètres qui peuvent impacter le plus les temps de chargement de pages en fonction des phases d'affichage, c.à.d. *First Paint* (le temps nécessaire pour afficher un premier pixel dans la surface visible), le *Time for Full Visual Rendering* (temps nécessaire pour afficher la partie visible de la page Web) et le *Page Load Time* (temps pour afficher l'intégralité de la page Web).

A.5 Prédiction de la qualité de la navigation Web

Avec plus de 17 Billions de sites Web que des utilisateurs peuvent visiter, les mesurer n'est pas évident selon des configurations unitaires en une seule journée. Il est dont crucial de pouvoir prédire la qualité d'expérience perçue par les utilisateurs. Grâce à nos mesures effectuées pendant 2.5 ans, nous avons pu identifier quels sont les paramètres pouvant contribuer à une QoE augmentée ou dégradée. Nous avons ensuite utilisé des techniques d'Intelligence Artificielle pour quantifier ces paramètres et ainsi pouvoir grâce à une suite de règles pouvoir prédire la QoE. Nous nous focalisons sur deux grandes étapes lors de l'affichage d'une page Web ; la partie visible de la page Web et l'intégralité de de la page. Pour la partie visible d'une page Web, nous pouvons prédire efficacement la QoE perçue à un taux de 90.4% pour des pages Web jamais mesurées. Concernant la prédiction de la QoE de l'intégralité de la page nous avons utilisé des solutions à base de chaines de Markov cachés pouvant augmenter le taux d'efficacité de prédiction jusqu'à 25%.

A.5.1 Prédiction de la qualité liée à la surface visible d'une page Web

Un grand nombre d'études a été effectué pour définir face à des utilisateurs réels les temps de chargement de la partie visible d'une page Web qui définissent différents degrés de qualité. Nous avons donc utilisé ces tranches de valeurs représentant la qualité ressentie par des utilisateurs et des arbres de décisions pour obtenir en sortie une suite de règles pouvant qualifier et quantifier la qualité d'expérience perçue. Nous regroupons 40% de nos mesures en fonction des valeurs de la métrique *Time for Full Visual Rendering* (TFVR) et des degrés de qualité. En fonction de ces différents groupes, un arbre de décision est construit et permet d'identifier pour chaque groupe les règles les qualifiant au mieux. Ces règles ont permis de confirmer notre étude statistique du Chapitre A-4. Les règles qualifiant la classe *Qualité Transparente* sont représentées sous la forme :

$$\{(2\% \leq TauxReceptionQUIC \leq 63.94\%) \wedge (protocoleDemandé = Quic\ Repeat)\} \wedge \{(RangAlexa \leq 42) \wedge (TempsReponsePagePrincipale \leq 18ms)\}$$

ou

$$\{localisationPagePrincipale = (NA \vee EU)\} \wedge \{protocoleDemandé = Quic\ Repeat\} \wedge \{accèsRéseau = Fibre\} \wedge \{nombreObjects \leq 21\}$$

En fonction de règles obtenues pour chaque classe, nous les avons appliquées sur le reste de nos données, c.à.d. 60% de nos mesures, sachant que le modèle n'a aucune connaissance de la valeur du TFVR. Pour une mesure donnée, la qualité du TFVR est alors prédite comme étant *instantanée, transparente, moyenne, critique* ou *mauvaise*. Nous vérifions ensuite la prédiction faite face à la vraie valeur du TFVR et représentons l'efficicaité des prédictions à travers une matrice de confusion représentée par le Tableau A-6. Ce tableau nous indique que seulement 84.79% des données sur lesquelles nous avons appliqué nos règles de décisions sont correctement prédites. De plus nous pouvons voir que le taux d'erreur est regroupé au sein de la classe *Critique* et qu'aucune mesure n'est prédite à appartenir à la classe *Instantanée.* Ces degrés de qualités ont été définis dans la littérature à partir d'un nombre de sites Web très limité et ne peuvent pas s'appliquer sur nos mesures des 10,000 sites Web les plus visités.

	Classe actuelle				
Classe prédite	**Instantanée**	**Transparente**	**Moyenne**	**Critique**	**Mauvaise**
Instantanée	0	0.15%	0	0	0
Transparente	0	22.21%	1.60%	0	0
Moyenne	0	1.96%	29.01%	8.07%	0
Critique	0	0	0.04%	32.35%	0.01%
Mauvaise	0	0	0.02%	3.36%	1.22%

Tableau A-6: Efficacité de prédiction du TFVR en utilisant les degrés de qualité de la littérature

	Classe actuelle				
Classe prédite	**Bonne**	**Correcte**	**Moyenne**	**Dégradée**	**Mauvaise**
Bonne	22.19%	0.04%	0	0	0
Correcte	1.13%	25.17%	0.82%	0	0
Moyenne	0	2.10%	29.22%	0.43%	0
Dégradée	0	0	1.20%	11.63%	0.06%
Mauvaise	0	0	0.02%	0.03%	5.96%

Tableau A-7: Efficacité de prédiction du TFVR en utilisant des degrés de qualité définis

Nous définissons alors nos propres degrés de qualité en fonction de nos données en effectuant un *clustering* des mesures du TFVR. À partir des ces degrés de qualité, nous fabriquons une fois de plus nos règles grâce à des arbres de décision à partir de 40% de nos mesures. Ces règles sont ensuite appliquées sur le reste de nos mesures, c.à.d. les 60%

restantes. L'efficacité de notre taux de prédiction est représentée à travers le Tableau A-7 et est de 94.17%.

	Classe actuelle				
Classe prédite	**Bonne**	**Correcte**	**Moyenne**	**Dégradée**	**Mauvaise**
Bonne	6.51%	0.16%	0	0	0
Correcte	0.76%	21.09%	0.72%	0	0.02%
Moyenne	0.01%	1.96%	35.02%	0.12%	0.01%
Dégradée	0	0	3.60%	17.74%	1.03%
Mauvaise	0	0	0	1.21%	10.04%

Tableau A-8: Efficacité de prédiction du TFVR pour des sites Web jamais mesurés

Nous vérifions par la suite la précision des règles identifiées sur des sites Web jamais mesurés auparavant. Grace à nos règles de décision identifiées à partir des sites Web de rang 1 à 10,000, nous faisons la prédiction de la qualité perçue de la surface visible des sites Web de rang 10,000 à 15,000. La prédiction est efficace à un taux de 90.4%, représenté par le Tableau A-8. Cela montre qu'à travers nos règles de décision, nous pouvons correctement prédire la qualité perçue lors du chargement de la surface visible d'une page Web.

	Classe actuelle					
Classe prédite	**Bonne**	**Correcte**	**Moyenne**	**Dégradée**	**Mauvaise**	**Absurde**
Bonne	3.02%	3.2%	2.07%	0	0	0
Correcte	0.82%	7.68%	10.78%	4.02%	0	0
Moyenne	0	0.69%	17.21%	5.79%	0.44%	0
Dégradée	0	0.09%	6.03%	11.9%	1.7%	0.06%
Mauvaise	0	0	0.15%	2.46%	8.02%	1.63%
Absurde	0	0	0	0.07%	2.1%	10.07%

Tableau A-9: Efficacité de prédiction du PLT pour des sites Web jamais mesurés

A.5.2 Prédiction de la qualité liée à une page Web entière

Nous nous intéressons maintenant à la prédiction de la qualité de la page Web entière. Nous avons appliqué le même procédé qu'identifié dans la section A.5.1. Pour les degrés de qualité en lien avec les temps de chargement de la page entière, nous les définissons à partir de 40% de nos mesures car il n'existe pas d'études impliquant des utilisateurs réels pour les définir. Nous construisons nos règles de décision et les appliquons sur 60% des mesures restantes et les taux de prédiction correctes sont de 68.6%. Ces mêmes règles de décisions appliquées sur les sites Web de rang 10,000 – 15,000 jamais mesurés auparavant engendrent un taux d'erreur de 42.1% dans la prédiction, comme illustré par le Tableau A-9.

Voulant comprendre pourquoi le taux de prédiction correct est si bas pour la page entière, sachant que pour la partie visible des pages Web, le taux de prédiction était élevé, nous nous intéressons à différents sites Web de façon unitaire. Grâce à notre site Web public, nous remarquons qu'entre le début de la navigation jusqu'à l'affichage de la surface visible, des fluctuations dans les temps de chargement pour une très grande partie des sites Web sont minimes. Par ailleurs, pour certains sites Web, les temps de chargement de la page Web entière fluctuent beaucoup.

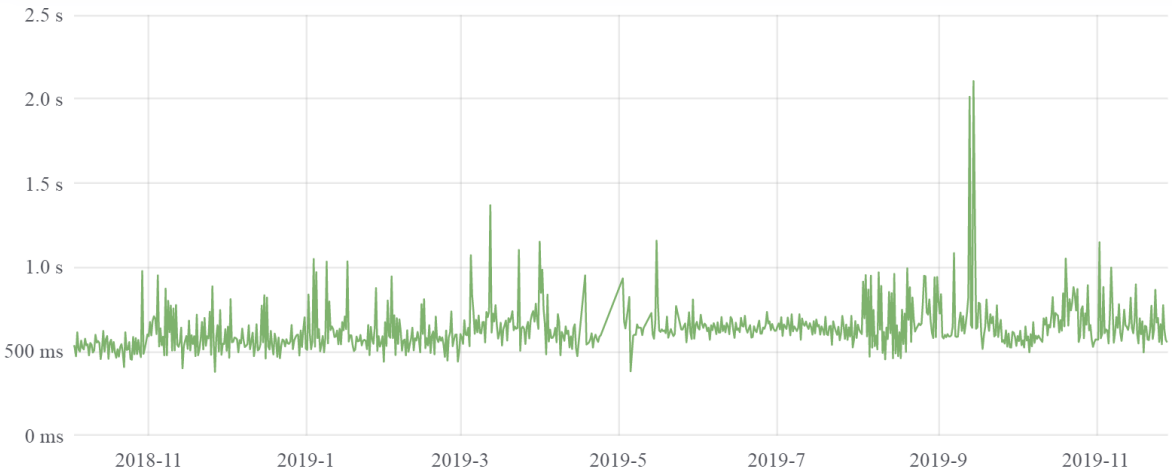


Figure A-12: PLT du site Web *wikipedia.org* entre octobre 2018 et novembre 2019

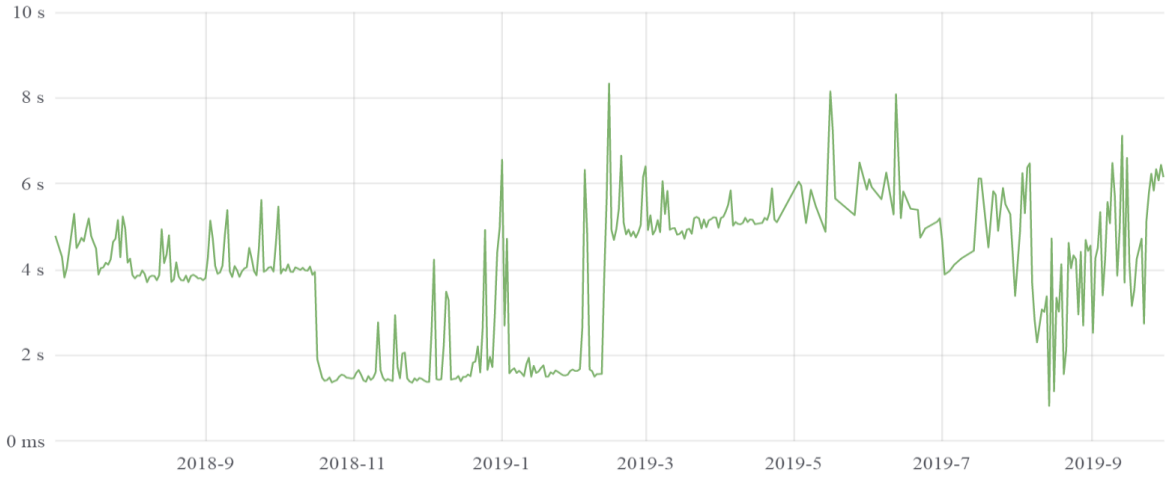


Figure A-13: PLT du site Web *baidu.com* entre juillet 2018 et septembre 2019

Nous avons alors construit nos règles de décision en se basant sur des sites Web dont les temps de chargement fluctuent très peu, e.x. *wikipedia.org*, représenté par la Figure A-12. Cette figure représente le *Page Load Time* (PLT) entre octobre 2018 et novembre 2019 et nous construisons nos règles à partir de ces mesures. Nous effectuons ensuite la prédiction de la qualité de la page entière pour le mois de décembre 2019. A notre surprise, la qualité est correctement prédite à un taux de 99.88%.

Nous nous sommes ensuite intéressés à des sites Web dont les temps de chargement fluctuent régulièrement, e.x. *baidu.com*, représenté par la Figure A-13. Cette figure représente le *Page Load Time* (PLT) entre juillet 2018 et septembre 2019. Nous construisons nos règles à partir des mesures effectuées entre juillet 2018 et janvier 2019. Nous effectuons ensuite la prédiction de la qualité de la page entière pour la période de février à septembre 2019. Le taux d'erreur dans la prédiction est de 25.08%.

Avec pour objectif de confirmer que notre modèle est inefficace face à des sites Web ayant des fluctuations importantes dans les temps de chargement pour la page entière, nous identifions 14 autres sites Web via notre site Web public ayant cette particularité et appliquons notre modèle. Les erreurs dans le taux de prédictions oscillent entre 21% et 34%. Cela démontre que pour les sites Web avec des temps de chargement qui fluctuent en permanence, d'autres solutions sont nécessaires pour augmenter l'efficacité des prédictions pour la page Web entière.

Figure A-14: HDP-HMM avec des modèles de mélange Gaussiens appliqué à la série temporelle de *baidu.com*

Sites Web	**Taux d'efficacité de prédiction (%)**		**Nombres d'états**	
	Arbres de décision et Clustering	Arbres de décision et HDP-HMM GMM	Clustering	HDP-HMM GMM
amazon.com	63.89	72.98	7	13
bbc.com	83.02	92.27	3	10
cdiscount.com	87.51	89.23	2	3
chinatimes.com	76.90	91.22	3	9
csdn.net	84.12	92.60	6	11
irishtimes.com	89.08	92.65	3	5
leboncoin.fr	81.29	97.55	6	10
lefigaro.fr	74.99	89.10	5	11
reddit.com	81.88	94.16	4	8
spiceworks.com	82.59	91.65	3	4
taobao.com	94.22	95.17	4	7
tumblr.com	71.25	86.95	5	6
twitch.tv	60.56	89.23	5	16
yahoo.co.jp	61.13	88.27	6	13

Tableau A-10: Taux de prédiction correcte et nombre d'états identifiés

Nous avons utilisé des chaines de Markov cachées suivant un processus de Dirichlet hiérarchique (HDP-HMM) [126] [127] [128] et des lois d'émissions étant des modèles de mélange Gaussien (GMM) pour mieux détecter ces fluctuations dans le temps et ainsi affiner nos règles. L'application de HDP-HMM sur la série temporelle du site Web *baidu.com* est représentée par la Figure A-14. De nouvelles tranches de degré de qualité sont alors définies, ce qui permet d'affiner nos règles de décisions. En appliquant ces nouvelles règles pour prédire la qualité de la page Web entière du site Web *baidu.com*, le taux de prédiction est augmenté de 21.5%. Nous avons appliqué cette solution sur 14 sites Web dont les temps de chargement fluctuent constamment et avons augmenté l'efficacité de prédiction jusqu'à 25%. Ces différents sites Web sont illustrés par le Tableau A-10.

Conclusion

Nous nous sommes intéressés dans cette thèse à caractériser, quantifier et prédire la qualité de la navigation Web. Le trafic global de l'Internet n'a cessé d'augmenter ces dernières années où la navigation Web occupe une place importante. Dans un monde où la connectivité est importante, il est crucial de pouvoir mieux comprendre l'écosystème du Web et ainsi éclairer les utilisateurs, fournisseurs de services et le monde de la recherche sur la façon dont la Qualité d'Expérience des utilisateurs peut être améliorée.

Pour mieux caractériser la qualité de la navigation Web, nous avons introduit un nouvel outil de mesure automatisé, Web View. Cet outil permet d'étudier l'écosystème du Web dans son intégralité en utilisant des navigateurs Web réels et est déployé sur des périphériques utilisateurs représentatifs. Le rattachement de ces périphériques à un accès réseau résidentiel est privilégié. Lors de la configuration des mesures, on peut y spécifier l'utilisation d'un navigateur particulier, le type de protocole Internet privilégié, l'utilisation de bloqueurs de publicités (ou pas) ou encore l'utilisation du cache (ou pas) du navigateur. Web View offre 84 paramètres qui permettent de qualifier la qualité de la navigation Web. Les mesures de certains sites Web prédéfinis sont représentées en temps réel sur un site Web public qui permet de suivre l'évolution du Web et aussi d'identifier en temps réel des dégradations de qualité.

Nous avons aussi étudié et utilisé différentes métriques Web et avons pu identifier un grand nombre d'inefficacités dans les méthodes de calcul, ne prenant pas en compte les nouvelles technologies du Web. Nous avons donc défini une nouvelle métrique Web, le *Time for Full Visual Rendering* (TFVR) qui est calculé à partir de logs réseau offerts par les navigateurs. Le TFVR comparé à d'autres métriques Web a montré son efficacité face à de nouvelles technologies Web et est calculé en temps réel sans l'utilisation d'outils externes qui peuvent impacter les mesures.

Grace à notre outil, Web View, et notre métrique, le TFVR, nous avons effectué des mesures de qualité de navigation Web pendant plus de 2.5 années. Ces mesures ont été analysées par différentes techniques statistiques, ce qui a permis d'identifier les acteurs impliqués dans la navigation Web. Nous avons aussi analysé l'impact de ces différents facteurs sur la qualité de la navigation Web.

Vu le grand nombre de sites Web, nous nous sommes intéressés à la prédiction de la qualité de la navigation Web liée à la surface visible ou intégralité d'une page Web. Pour prédire la qualité du temps d'affichage de la partie visible des pages, nous avons utilisé des arbres de décision pour qualifier et quantifier les facteurs contribuant à divers degrés de qualité. Ces règles de décision ont été identifiés à partir des 10,000 sites Web les plus visités et la prédiction de qualité effectuée sur les sites Web de rang 10,000 à 15,000. Le taux d'efficacité du taux de prédiction est de 90,4%. Concernant la prédiction de la qualité pour la page Web entière, ces règles de décision ont montrés qu'elles sont efficaces pour prédire la qualité (taux d'efficacité de 99.88% pour le site Web *wikipedia.org*) si et seulement si les fluctuations dans les temps de chargement sont minimes. Pour les sites Web où les temps de chargement de la page entière fluctuent grandement et régulièrement au cours du temps, nous avons utilisé des chaines de Markov cachées suivant un processus de Dirichlet hiérarchique (HDP-HMM) et des lois d'émissions étant des modèles de mélange Gaussien (GMM). Cela permet d'identifier finement ces fluctuations et ainsi affiner nos règles de décisions. L'utilisation d'arbres de décisions couplé au HDP-HMM a permis d'augmenter l'efficacité de prédiction de ces sites Web jusqu'à 25%.

Notre modèle à base de règles de décision est efficace à prédire la qualité de la surface visible de n'importe quel site Web. Pour la prédiction de la qualité de la page Web entière, si nous avons peu de fluctuations dans les temps de chargement, notre modèle peut être appliqué à n'importe quel site Web. Si les fluctuations sont importantes, une suite de règle de décision spécifique est nécessaire pour chacun de ces sites Web.

Bibliography

[1] Vicki L. Hanson, "Taking control of web browsing," *The New Review of Hypermedia and Multimedia*, vol. 10, pp. 127-140, 2004.

[2] Abu Awal Md Shoeb, "Is Private Browsing in Modern Web Browsers Really Private?," *CoRR*, vol. abs/1802.10523, 2018.

[3] Adrienne Porter Felt et al., "Measuring HTTPS Adoption on the Web," in *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, 2017, pp. 1323-1338.

[4] George Michaelson, Matthew Roughan, Jonathan Tuke, Matt P. Wand, and Randy Bush, "Rigorous statistical analysis of HTTPS reachability," *CoRR*, vol. abs/1706.02813, 2017.

[5] Henrik Frystyk Nielsen et al., "Hypertext Transfer Protocol -- HTTP/1.1," Tech. rep. June 1999. [Online]. https://rfc-editor.org/rfc/rfc2616.txt

[6] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC Editor, RFC ISSN: 2070-1721, May 2015. [Online]. http://www.rfc-editor.org/rfc/rfc7540.txt

[7] Jana Iyengar and Martin Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Internet-Draft Sep. 11, 2019. [Online]. https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-23

[8] Ali Raza, Yasir Zaki, Thomas Pötsch, Jay Chen, and Lakshmi Subramanian, "Extreme Web Caching for Faster Web Browsing," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*, 2015, pp. 111-112.

[9] Shankaranarayanan Puzhavakath Narayanan et al., "Reducing Latency Through Page-aware Management of Web Objects by Content Delivery Networks," in *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science, Antibes Juan-Les-Pins, France, June 14-18, 2016*, 2016, pp. 89-100.

[10] Stathes Hadjiefthymiades and Lazaros F. Merakos, "Using proxy cache relocation to accelerate Web browsing in wireless/mobile communications," in *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, 2001, pp. 26-35.

[11] Michael Butkiewicz, Harsha V. Madhyastha, and Vyas Sekar, "Characterizing Web Page Complexity and Its Impact," *IEEE/ACM Trans. Netw.*, vol. 22, pp. 943-956, 2014.

[12] Michael Butkiewicz, Harsha V. Madhyastha, and Vyas Sekar, "Understanding website complexity: measurements, metrics, and implications," in *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2-, 2011*, 2011, pp. 313-328.

[13] Sergio Duarte Torres, Ingmar Weber, and Djoerd Hiemstra, "Analysis of Search and Browsing Behavior of Young Users on the Web," *TWEB*, vol. 8, pp. 7:1--7:54, 2014.

[14] Fabio Gasparetti, "Modeling user interests from web browsing activities," *Data Min. Knowl. Discov.*, vol. 31, pp. 502-547, 2017.

[15] Alexey Tikhonov, Liudmila Ostroumova Prokhorenkova, Arseniy Chelnokov, Ivan Bogatyy, and Gleb Gusev, "What can be Found on the Web and How: A Characterization of Web Browsing Patterns," in *Proceedings of the ACM Web Science*

Conference, WebSci 2015, Oxford, United Kingdom, June 28 - July 1, 2015, 2015, pp. 14:1--14:10.

[16] Sharad Goel, Jake M. Hofman, and M. Irmak Sirer, "Who Does What on the Web: A Large-Scale Study of Browsing Behavior," in *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.

[17] Norikatsu Nagino and Seiji Yamada, "Future View: Web Navigation Based on Learning Users Browsing Patterns," in *2003 IEEE / WIC International Conference on Web Intelligence, (WI 2003), 13-17 October 2003, Halifax, Canada*, 2003, pp. 541-544.

[18] R. Peon and H. Ruellan, "HPACK: Header Compression for HTTP/2," RFC Editor, RFC ISSN: 2070-1721, May 2015.

[19] Jawad Manzoor, Idilio Drago, and Ramin Sadre, "How HTTP/2 is changing web traffic and how to detect it," in *Network Traffic Measurement and Analysis Conference, TMA 2017, Dublin, Ireland, June 21-23, 2017*, 2017.

[20] P. Megyesi, Z. Kramer, and S. Molnar, "How Quick is QUIC?," in *Proceedings of IEEE International Conference on Communications (ICC)*, May 2016.

[21] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better For What And For Whom?," in *Proceedings of IEEE International Conference on Communications (ICC)*, May 2017.

[22] Kaname Harumoto, Tadashi Nakano, Shinya Fukumura, Shinji Shimojo, and Shojiro Nishio, "Effective Web browsing through content delivery adaptation," *ACM Trans. Internet Techn.*, vol. 5, pp. 571-600, 2005.

[23] Faraz Ahmed, M. Zubair Shafiq, Amir R. Khakpour, and Alex X. Liu, "Optimizing Internet Transit Routing for Content Delivery Networks," *IEEE/ACM Trans. Netw.*, vol. 26, pp. 76-89, 2018.

[24] Ruwaifa Anwar et al., "Investigating Interdomain Routing Policies in the Wild," in *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, 2015.

[25] Giang T. K. Nguyen, Xun Gong, Anupam Das, and Nikita Borisov, "PnP: improving web browsing performance over tor using web resource prefetch-and-push," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, 2013, pp. 1433-1436.

[26] Shui Yu, Theerasak Thapngam, Su Wei, and Wanlei Zhou, "Efficient Web Browsing with Perfect Anonymity Using Page Prefetching," in *Algorithms and Architectures for Parallel Processing, 10th International Conference, ICA3PP 2010, Busan, Korea, May 21-23, 2010. Proceedings. Part I*, 2010, pp. 1-12.

[27] Theresa Enghardt, Thomas Zinner, and Anja Feldmann, "Web Performance Pitfalls," in *Passive and Active Measurement - 20th International Conference, PAM 2019, Puerto Varas, Chile, March 27-29, 2019, Proceedings*, 2019, pp. 286-303.

[28] Dennis Guse, Sebastian Schuck, Oliver Hohlfeld, Alexander Raake, and Sebastian Moller, "Subjective quality of webpage loading: The impact of delayed and missing elements on quality ratings and task completion time," in *Seventh International Workshop on Quality of Multimedia Experience, QoMEX 2015*.

[29] Fintan Culwin and Xristine Faulkner, "Browsing the Web: Delay, Determination and Satisfaction," in *34th Annual Hawaii International Conference on System Sciences (HICSS-34), January 3-6, 2001, Maui, Hawaii, USA*, 2001.

[30] Anirban Banerji, "How happy is your web browsing? A probabilistic model to describe user satisfaction," *CoRR*, vol. abs/0902.1104, 2009.

[31] Barbara Rita Barricelli, Marco Padula, and Paolo Luigi Scala, "Personalized web browsing experience," in *HYPERTEXT 2009, Proceedings of the 20th ACM Conference*

on Hypertext and Hypermedia, Torino, Italy, June 29 - July 1, 2009, 2009, pp. 345-346.

[32] Henrik Frystyk Nielsen, Roy T. Fielding, and Tim Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.0," Tech. rep. May 1996. [Online]. https://rfc-editor.org/rfc/rfc1945.txt

[33] Matteo Varvello et al., "Is the Web HTTP/2 Yet?," in *Passive and Active Measurement - 17th International Conference, PAM 2016, Heraklion, Greece, March 31 - April 1, 2016. Proceedings*, 2016, pp. 218-232.

[34] Kyriakos Zarifis, Mark Holland, Manish Jain, Ethan Katz-Bassett, and Ramesh Govindan, "Modeling HTTP/2 Speed from HTTP/1 Traces," in *Passive and Active Measurement - 17th International Conference, PAM 2016, Heraklion, Greece, March 31 - April 1, 2016. Proceedings*, 2016.

[35] Torsten Zimmermann, Jan Rüth, Benedikt Wolters, and Oliver Hohlfeld, "How HTTP/2 pushes the web: An empirical study of HTTP/2 server push," in *2017 IFIP Networking Conference, IFIP Networking 2017 and Workshops, Stockholm, Sweden, June 12-16, 2017*, 2017, pp. 1-9.

[36] Enrico Bocchi, Luca De Cicco, Marco Mellia, and Dario Rossi, "The Web, the Users, and the MOS: Influence of HTTP/2 on User Experience," in *Passive and Active Measurement - 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings*, 2017.

[37] Arash Molavi Kakhki, Samuel Jero, David R. Choffnes, Cristina Nita-Rotaru, and Alan Mislove, "Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols," in *Proceedings of the 2017 Internet Measurement Conference, IMC 2017, London, United Kingdom, November 1-3, 2017*, 2017.

[38] Jan Ruth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld, "A First Look at QUIC in the Wild," in *Passive and Active Measurement - 19th International Conference, PAM 2018, Berlin, Germany, March 26-27, 2018, Proceedings*, 2018.

[39] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, "How Secure and Quick is QUIC? Provable Security and Performance Analyses," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, May 2015.

[40] P. Biswal and O. Gnawali, "Does QUIC Make the Web Faster?," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1-6.

[41] Michael Seufert, Raimund Schatz, Nikolas Wehner, Bruno Gardlo, and Pedro Casas, "Is QUIC becoming the New TCP? On the Potential Impact of a New Protocol on Networked Multimedia QoE," in *11th International Conference on Quality of Multimedia Experience QoMEX 2019, Berlin, Germany, June 5-7, 2019*, 2019, pp. 1-6.

[42] Albert Levi, "How secure is secure Web browsing?," *Commun. ACM*, vol. 46, p. 152, 2003.

[43] Marc Fischlin and Felix Günther, "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2014, pp. 1193-1204.

[44] Eric Rescorla, "Diffie-Hellman Key Agreement Method," *RFC*, vol. 2631, pp. 1-13, 1999.

[45] Eric Rescorla and Tim Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," Tech. rep. Aug. 2008. [Online]. https://rfc-editor.org/rfc/rfc5246.txt

[46] Eric Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Tech. rep. Aug. 2018. [Online]. https://rfc-editor.org/rfc/rfc8446.txt

[47] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla Merwe, "A Comprehensive Symbolic Analysis of TLS 1.3," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, 2017, pp. 1773-1788.

[48] Ralph Holz, Johanna Amann, Abbas Razaghpanah, and Narseo Vallina-Rodriguez, "The Era of TLS 1.3: Measuring Deployment and Use with Active and Passive Methods," *CoRR*, vol. abs/1907.12762, 2019.

[49] Ghada Arfaoui, Xavier Bultel, Pierre-Alain Fouque, Adina Nedelcu, and Cristina Onete, "The privacy of the TLS 1.3 protocol," *PoPETs*, vol. 2019, pp. 190-210, 2019.

[50] Ivan Tam Ming-Chit, Mohan Krishna Patnam, King Tung Chan, and Foo Siang Fook, "Performance evaluation of Web browsing over hybrid fiber coaxial broad-band networks," in *Proceedings of the IEEE International Conference on Networks 1999, ICON 1999, Brisbane, Queensland, Australia, September 28 - October 1, 1999*, 1999, pp. 372-382.

[51] Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith, "Tracking the Evolution of Web Traffic: 1995-2003," in *11th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2003), 12-15 October 2003, Orlando, FL, USA*, 2003, pp. 16-25.

[52] Matteo Varvello, Jeremy Blackburn, David Naylor, and Konstantina Papagiannaki, "EYEORG: A Platform For Crowdsourcing Web Quality Of Experience Measurements," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies, CoNEXT 2016, Irvine, California, USA, December 12-15, 2016*, 2016, pp. 399-412.

[53] Kai Shuang, Tong Zhang, Zhenjiang Dong, and Peng Xu, "Impact of HTTP Pipelining Mechanism for Web Browsing Optimization," in *2015 IEEE International Conference on Mobile Services, MS 2015, New York City, NY, USA, June 27 - July 2, 2015*, 2015, pp. 415-422.

[54] Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall, "Demystifying Page Load Performance with WProf," in *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2013, Lombard, IL, USA, April 2-5, 2013*, 2013.

[55] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: an Experimental Investigation of QUIC," in *Proceedings of the 30th ACM/SIGAPP Symposium On Applied Computing (SAC)*, Apr. 2015.

[56] Antoine Saverimoutou, Bertrand Mathieu, and Sandrine Vaton, "Which secure transport protocol for a reliable HTTP/2-based web service: TLS or QUIC?," in *2017 IEEE Symposium on Computers and Communications, ISCC 2017, Heraklion, Greece, July 3-6, 2017*, 2017, pp. 879-884. [Online]. https://doi.org/10.1109/ISCC.2017.8024637

[57] Marcel Flores and Harkeerat Bedi, "Caching the Internet: A View from a Global Multi-tenant CDN," in *Passive and Active Measurement - 20th International Conference, PAM 2019, Puerto Varas, Chile, March 27-29, 2019, Proceedings*, 2019, pp. 68-81.

[58] Michael J. Freedman, "Experiences with CoralCDN: A Five-Year Operational View," in *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010, April 28-30, 2010, San Jose, CA, USA*, 2010, pp. 95-110.

[59] Kirk L. Johnson, John F. Carr, Mark S. Day, and M. Frans Kaashoek, "The measured performance of content distribution networks," *Computer Communications*, vol. 24, pp. 202-206, 2001.

[60] Balachander Krishnamurthy, Craig E. Wills, and Yin Zhang, "On the use and performance of content distribution networks," in *Proceedings of the 1st ACM SIGCOMM Internet Measurement Workshop, IMW 2001, San Francisco, California, USA, November 1-2, 2001*, 2001, pp. 169-182.

[61] Stefan Saroiu, P. Krishna Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy, "An Analysis of Internet Content Delivery Systems," in *5th Symposium on Operating System Design and Implementation (OSDI 2002), Boston, Massachusetts,*

USA, December 9-11, 2002, 2002.

[62] P. Krishna Gummadi et al., "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003, SOSP 2003, Bolton Landing, NY, USA, October 19-22, 2003*, 2003, pp. 314-329.

[63] Usama Naseer and Theophilus Benson, "InspectorGadget: Inferring Network Protocol Configuration for Web Services," in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, 2018.

[64] Sunghwan Ihm and Vivek S. Pai, "Towards understanding modern web traffic," in *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2-, 2011*, 2011, pp. 295-312.

[65] Ilya Grigorik and Al., "Navigation Timing," W3C Recommendation, Tech. rep. Sep. 26, 2018.

[66] Todd Reifsteck and Al., "Resource Timing," W3C Recommendation, Tech. rep. Oct. 11, 2018.

[67] Shubhie Panicker, "Paint Timing," W3C Working Draft, Tech. rep. Sep. 07, 2017.

[68] Speed Index. [Online]. https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index

[69] Antoine Saverimoutou, Bertrand Mathieu, and Sandrine Vaton, "Web View: Measuring and Monitoring Representative Information on Websites," in *IEEE International Workshop on Quality of Experience Management, QOE-MANAGEMENT Paris, France, February 18, 2019*, 2019.

[70] Steven Englehardt and Arvind Narayanan, "Online Tracking: A 1-million-site Measurement and Analysis," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 1388-1401.

[71] Brendan Cody-Kenny et al., "Investigating the Evolvability of Web Page Load Time," in *Applications of Evolutionary Computation - 21st International Conference, EvoApplications 2018, Parma, Italy, April 4-6, 2018, Proceedings*, 2018, pp. 769-777.

[72] David Rozado Fernandez, Ahmad El Shoghri, and Raja Jurdak, "Gaze dependant prefetching of web content to increase speed and comfort of web browsing," *Int. J. Hum.-Comput. Stud.*, vol. 78, pp. 31-42, 2015.

[73] Raimund Schatz and Sebastian Egger, "An annotated dataset for Web Browsing QOE," in *Sixth International Workshop on Quality of Multimedia Experience, QoMEX 2014, Singapore, September 18-20, 2014*, 2014, pp. 61-62.

[74] Pengfei Wang, Matteo Varvello, and Aleksandar Kuzmanovic, "Kaleidoscope: A Crowdsourcing Testing Tool for Web Quality of Experience," *The 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)*, 2019.

[75] Conor Kelton, Jihoon Ryoo, Aruna Balasubramanian, and Samir R. Das, "Improving User Perceived Page Load Times Using Gaze," in *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, 2017, pp. 545-559.

[76] Enrico Bocchi, Luca De Cicco, and Dario Rossi, "Measuring the Quality of Experience of Web users," *Computer Communication Review*, 2016.

[77] Jan-Niklas Voigt-Antons, Tobias Hoßfeld, Sebastian Egger-Lampl, Raimund Schatz, and Sebastian Möller, "User Experience of Web Browsing - The Relationship of Usability and Quality of Experience," in *Tenth International Conference on Quality of Multimedia Experience, QoMEX 2018, Cagliari, Italy, May 29 - June 1, 2018*, 2018, pp. 1-3.

[78] Robert E. Kooij, Robert D. Mei, and Ran Yang, "TCP and web browsing performance in case of bi-directional packet loss," *Computer Communications*, vol. 33, pp. S50--S57, 2010.

[79] Dominik Strohmeier, Sebastian Egger, Alexander Raake, Tobias Hoßfeld, and Raimund Schatz, "Web Browsing," in *Quality of Experience.*, 2014, pp. 329-338.

[80] Yang Yang et al., "Scout: A Point of Presence Recommendation System Using Real User Monitoring Data," in *Passive and Active Measurement - 17th International Conference, PAM 2016, Heraklion, Greece, March 31 - April 1, 2016. Proceedings*, 2016.

[81] Qingzhu Gao, Prasenjit Dey, and Parvez Ahammad, "Perceived Performance of Top Retail Webpages In the Wild: Insights from Large-scale Crowdsourcing of Above-the-Fold QoE," in *Proceedings of the 2017 Workshop on QoE-based Analysis and Management of Data Communication Networks, Internet-QoE@SIGCOMM 2017, Los Angeles, CA, USA, August 21, 2017*, 2017.

[82] Toshiko Tominaga et al., "Web-Browsing QoE Estimation Model," *IEICE Transactions*, vol. 100-B, pp. 1837-1845, 2017.

[83] Antoine Saverimoutou, Bertrand Mathieu, and Sandrine Vaton, "Web Browsing Measurements: An Above-the-Fold Browser-Based Technique," in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, 2018.

[84] Diego Neves Da Hora, Alemnew Sheferaw Asrese, Vassilis Christophides, Renata Teixeira, and Dario Rossi, "Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics," in *Passive and Active Measurement - 19th International Conference*, 2018.

[85] Tobias Hobfeld, Florian Metzger, and Dario Rossi, "Speed Index: Relating the Industrial Standard for User Perceived Web Performance to web QoE," in *Tenth International Conference on Quality of Multimedia Experience*, 2018.

[86] Ilya Grigorik. (2012, July) Above the fold time: Measuring web page performance visually. [Online]. http://conferences.oreilly.com/velocity/velocity-mar2011/public/schedule/detail/18692

[87] Zeinab Abbassi, Nidhi Hegde, and Laurent Massoulié, "Distributed content curation on the web," *SIGMETRICS Performance Evaluation Review*, 2014.

[88] Peter Snyder, Lara Ansari, Cynthia Taylor, and Chris Kanich, "Browser Feature Usage on the Modern Web," in *Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14-16, 2016*, 2016.

[89] Fabio Gasparetti and Alessandro Micarelli, "Exploiting web browsing histories to identify user needs," in *Proceedings of the 12th International Conference on Intelligent User Interfaces, IUI 2007, Honolulu, Hawaii, USA, January 28-31, 2007*, 2007, pp. 325-328.

[90] Luca Vassio, Idilio Drago, Marco Mellia, Zied Ben-Houidi, and Mohamed Lamine Lamali, "You, the Web and Your Device: Longitudinal Characterization of Browsing Habits," *CoRR*, vol. abs/1806.07158, 2018.

[91] Wojciech Jaworski, "Identifying Web Users on the Base of their Browsing Patterns," *Int. J. Comput. Intell. Syst.*, vol. 4, pp. 1062-1069, 2011.

[92] Alan L. Montgomery and Christos Faloutsos, "Identifying Web Browsing Trends and Patterns," *IEEE Computer*, vol. 34, pp. 94-95, 2001.

[93] Ganesan Velayathan and Seiji Yamada, "Investigating User Browsing Behavior," in *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops, 2-5 November 2007, Silicon Valley, CA, USA*, 2007, pp. 195-198.

[94] Eva M. Thury, "Analysis of Student Web Browsing Behavior: Implications for Designing and Evaluating Web Sites," in *The Sixteenth Annual International Conference of Computer Documentation, Scaling The Heights: The Future of Information Technology, September 23-26, 1998, Quebec City, Canada, Proceedings*, 1998, pp. 265-270.

[95] Gengsheng Zhao, Ning Zhang, Zhaoxing Liu, and Jiming Li, "A Case Investigation on the Scaling Behaviors in Web Browsing," in *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops, Toronto, Canada, August 31 - September 3, 2010*, 2010, pp. 160-163.

[96] Young-Woo Seo and Byoung-Tak Zhang, "Learning user's preferences by analyzing Web-browsing behaviors," in *Proceedings of the Fourth International Conference on Autonomous Agents, AGENTS 2000, Barcelona, Catalonia, Spain, June 3-7, 2000*, 2000, pp. 381-387.

[97] Ting-Peng Liang and Hung-Jen Lai, "Discovering User Interests from Web Browsing Behavior: An Application to Internet News Services," in *35th Hawaii International Conference on System Sciences (HICSS-35 2002), CD-ROM / Abstracts Proceedings, 7-10 January 2002, Big Island, HI, USA*, 2002, p. 203.

[98] Gunes Acar et al., "FPDetective: dusting the web for fingerprinters," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, 2013, pp. 1129-1140.

[99] Jonathan R. Mayer and John C. Mitchell, "Third-Party Web Tracking: Policy and Technology," in *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, 2012, pp. 413-427.

[100] Mathias Lécuyer et al., "XRay: Enhancing the Web's Transparency with Differential Correlation," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, 2014, pp. 49-64.

[101] Chameleon Crawler. [Online]. https://github.com/ghostwords/chameleon

[102] PageSpeed Insights. [Online]. https://developers.google.com/speed/pagespeed/insights/

[103] WebPageTest. [Online]. https://www.webpagetest.org

[104] sitespeed.io. [Online]. https://www.sitespeed.io/

[105] YSlow. [Online]. http://yslow.org/

[106] Timothy Libert, "Exposing the Hidden Web: An Analysis of Third-Party HTTP Requests on 1 Million Websites," , vol. abs/1511.00619, 2015.

[107] Amit Datta, Michael Carl Tschantz, and Anupam Datta, "Automated Experiments on Ad Privacy Settings," , vol. 2015, 2015, pp. 92-112.

[108] Guillermo Jesús Hoyos-Rivera, Roberta Lima-Gomes, and Jean-Pierre Courtiat, "CoLab: A Flexible Collaborative Web Browsing Tool," in *19th International Conference on Advanced Information Networking and Applications (AINA 2005), 28-30 March 2005, Taipei, Taiwan*, 2005, pp. 501-506.

[109] G. Malkin, "Traceroute Using an IP Option," Internet Engineering Task Force, RFC Jan. 1993. [Online]. http://www.rfc-editor.org/rfc/rfc1393.txt

[110] Brice Augustin et al., "Avoiding traceroute anomalies with Paris traceroute," in *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeriro, Brazil, October 25-27, 2006*, 2006, pp. 153-158.

[111] Brice Augustin, Timur Friedman, and Renata Teixeira, "Multipath tracing with Paris traceroute," in *Fifth IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services, E2EMON 2007, 21st May, 2007, Munich, Germany*, 2007, pp. 1-8.

[112] Kevin Vermeulen, Stephen D. Strowes, Olivier Fourmaux, and Timur Friedman,

"Multilevel MDA-Lite Paris Traceroute," in *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*, 2018, pp. 29-42.

[113] Yakov Rekhter, Susan Hares, and Tony Li, A Border Gateway Protocol 4 (BGP-4), Jan. 2006.

[114] Alexander Marder and Jonathan M. Smith, "MAP-IT: Multipass Accurate Passive Inferences from Traceroute," in *Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14-16, 2016*, 2016, pp. 397-411.

[115] Khalid Bakhshaliyev, Muhammed Abdullah Canbaz, and Mehmet Hadi Gunes, "Investigating Characteristics of Internet Paths," *TOMPECS*, vol. 4, pp. 16:1--16:24, 2019.

[116] Vasileios Giotsas et al., "Detecting Peering Infrastructure Outages in the Wild," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, 2017, pp. 446-459.

[117] Matthew J. Luckie and Robert Beverly, "The Impact of Router Outages on the AS-level Internet," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, 2017, pp. 488-501.

[118] Mahdi Jalili, "Error and attack tolerance of small-worldness in complex networks," *J. Informetrics*, vol. 5, pp. 422-430, 2011.

[119] Lun Li, David L. Alderson, Walter Willinger, and John Doyle, "A first-principles approach to understanding the internet's router-level topology," in *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA*, 2004, pp. 3-14.

[120] Andra Lutu, Marcelo Bagnulo, Cristel Pelsser, Olaf Maennel, and Jesús Cid-Sueiro, "The BGP Visibility Toolkit: Detecting Anomalous Internet Routing Behavior," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 1237-1250, 2016.

[121] Antoine Saverimoutou, Bertrand Mathieu, and Sandrine Vaton, "Web View: A Measurement Platform for Depicting Web Browsing Performance and Delivery," in *IEEE Communications Magazine (to appear)*, 2020.

[122] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Processing*, vol. 13, pp. 600-612, 2004. [Online]. https://doi.org/10.1109/TIP.2003.819861

[123] Nobuyuki Otsu, "A Threshold Selection Method from Gray-Level Histograms," , 1979.

[124] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar R. Weippl, ""I Have No Idea What I'm Doing" - On the Usability of Deploying HTTPS," in *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, 2017, pp. 1339-1356.

[125] Antoine Saverimoutou, Bertrand Mathieu, and Sandrine Vaton, "Influence of Internet Protocols and CDN on Web Browsing," in *10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019, Canary Islands, Spain, June 24-26, 2019*, 2019, pp. 1-5. [Online]. https://doi.org/10.1109/NTMS.2019.8763827

[126] Maxime Mouchet, Sandrine Vaton, and Thierry Chonavel, "Statistical Characterization of Round-Trip Times with Nonparametric Hidden Markov Models," in *IFIP/IEEE International Symposium on Integrated Network Management, IM 2019, Washington, DC, USA, April 09-11, 2019.*, 2019, pp. 43-48.

[127] Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky, "A sticky HDP-HMM with application to speaker diarization," *The Annals of Applied Statistics*, vol. 5, pp. 1020-1056, 2011.

[128] Radford M. Neal, "Markov chain sampling methods for Dirichlet process mixture models," *JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS*, vol. 9, pp. 249-265, 2000.

[129] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," Internet Engineering Task Force, RFC Jan. 2001. [Online]. http://www.rfc-editor.org/rfc/rfc3022.txt

[130] Dublin-Traceroute. [Online]. https://dublin-traceroute.net/README.md

[131] Catalin-Alexandru Avram, Kenneth Salem, and Bernard Wong, "Latency Amplification: Characterizing the Impact of Web Page Content on Load Times," in *33rd IEEE International Symposium on Reliable Distributed Systems Workshops, SRDS Workshops 2014, Nara, Japan, October 6-9, 2014*, 2014, pp. 20-25.

[132] Matthew Malloy, Mark McNamara, Aaron Cahn, and Paul Barford, "Ad Blockers: Global Prevalence and Impact," in *Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14-16, 2016*, 2016.

[133] Israel Brosh, Eliezer Shlifer, and Paul J. Schweitzer, "Generalized Markovian decision processes," *Math. Meth. of OR*, vol. 21, pp. 173-186, 1977.

[134] Simon Wimmer, "Hidden Markov Models," *Archive of Formal Proofs*, vol. 2018, 2018.

[135] I. I. I. Hal Daumé, "Fast search for Dirichlet process mixture models," *CoRR*, vol. abs/0907.1812, 2009.

[136] Pablo E. Román and Juan D. Velásquez, "A Web Browsing Cognitive Model," in *Knowledge Engineering, Machine Learning and Lattice Computing with Applications - 16th International Conference, KES 2012, San Sebastian, Spain, September 10-12, 2012, Revised Selected Papers*, 2012, pp. 31-40.

[137] Anatoli Nachev and Ivan Ganchev, "An Approach to Clustering Web Browsing Patterns by ART2 Neural Networks with General Learning Rules," in *Proceedings of the International Conference on Artificial Intelligence, IC-AI '04, June 21-24, 2004, Las Vegas, Nevada, USA, Volume 1*, 2004, pp. 126-132.

[138] Tingshao Zhu, Russell Greiner, Gerald Häubl, Kevin Jewell, and Robert Price, "Using Learned Browsing Behavior Models to Recommend Relevant Web Pages," in *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, 2005, pp. 1589-1590.

[139] Tingshao Zhu, "Clustering Web Users Based on Browsing Behavior," in *Active Media Technology, 6th International Conference, AMT 2010, Toronto, Canada, August 28-30, 2010. Proceedings*, 2010, pp. 530-537.

[140] Mamoun A. Awad and Issa Khalil, "Prediction of User's Web-Browsing Behavior: Application of Markov Model," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 42, pp. 1131-1142, 2012.

[141] Evgeniy Mokrov et al., "Modelling And Response Time Analysis For Web Browsing Under Interruptions In LTE Network," in *European Conference on Modelling and Simulation, ECMS 2017, Budapest, Hungary, May 23-26, 2017, Proceedings.*, 2017, pp. 706-712.

[142] Maxime Mouchet, Sandrine Vaton, and Thierry Chonavel, "Poster Abstract: A flexible infinite HMM model for accurate characterization and segmentation of RTT timeseries," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019*, 2019, pp. 1055-1056.

[143] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B*, vol. 39, pp. 1-38, 1977. [Online]. http://web.mit.edu/6.435/www/Dempster77.pdf

[144] Antoine Saverimoutou, Bertrand Mathieu, and Sandrine Vaton, "A 6-month analysis of factors impacting web browsing quality for QoE prediction," *Computer Networks*, vol. 164, p. 106905, 2019. [Online]. http://www.sciencedirect.com/science/article/pii/S1389128619307546

List of Abreviations

ADSL	Asymmetric Digital Subscriber Line
AS	Autonomous System
ATF	Above-The-Fold
BGP	Border Gateway Protocol
CDN	Content Delivery Network
CSSOM	Cascading Style Sheet Object Model
DH	Diffie-Hellman
DNS	Domain Name System
DoH	DNS over HTTPS
DOM	Document Object Model
FCP	First Contentful Paint
FMP	First Meaningful Paint
FP	First Paint
GMM	Gaussian Mixture Model
GQUIC	Google QUIC
HAR	Http Archive
HDP-HMM	Hierarchical Dirichlet Process Hidden Markov Models
HOL	Head-Of-Line
HTML	HyperText Markup Language
HTTP	HyperText transfer Protocol
IANA	Internet Assigned Numbers Authority
IEFT	Internet Engineering Task Force
IQUIC	IETF QUIC
ISP	Internet Service Provider
MAC	Message Authentication Code
MDA	Multipath Detection Algorithm

MIME	Multipurpose Internet Mail Extension
MOS	Mean Opinion Score
NAT	Network Address Translator
OTI	Open Transit Internet
PLT	Page Load Time
PWA	Progressive Web Applications
QoE	Quality of Experience
QoS	Quality of Service
QUIC	Quick UDP Internet Connections
RTT	Round-Trip-Time
RUM-SI	Real User Monitoring Speed Index
SI	Speed Index
SSIM	Structural Similarity Index
TCP	Transmission Control Protocol
TFVR	Time for Full Visual Rendering
TLS	Transport Layer Security
TTI	Time-To-Interactive
TTL	Time-To-Live
UDP	User Datagram Protocol
W3C	World Wide Web Consortium

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre: Métrologie de l'Internet du futur : Caractérisation, Quantification et Prédiction de la Qualité de la Navigation Web

Mots clés : Navigation Web, Intelligence Artificielle, QoE, QoS, Prédiction.

La navigation Web est l'un des principaux services de l'Internet où un large éventail d'acteurs est impliqué et évolue de manière constante. Les nouveaux protocoles de l'Internet, les réseaux de diffusion de contenu (CDN) ou encore les évolutions des différents navigateurs Web sont destinés à améliorer les temps de chargement des pages. Pour mieux comprendre la Qualité d'Expérience (QoE) perçue par les utilisateurs, il est donc essentiel d'identifier comment le contenu des pages est constitué et délivré et de fournir une métrique de QoE pertinente. Dans cette thèse, nous avons conçu un nouvel outil, Web View, destiné à effectuer des sessions de navigation Web automatisées et mesurant de nombreuses informations liées à l'écosystème du Web. Nous avons aussi défini une nouvelle métrique Web, le Time for Full Visual Rendering (TFVR). À partir de plus de 18 trillions de mesures effectuées pendant 2,5 ans sur les 10,000 sites Web les plus visités (selon la classification Alexa), nous avons utilisé des techniques statistiques pour identifier les paramètres clés qualifiant et quantifiant la qualité de la navigation Web. Cet ensemble de facteurs a été confirmé par un processus d'apprentissage automatique, donnant en sortie un ensemble de règles pour prédire les temps de chargement des pages Web. Pour les pages Web où des fluctuations dans les temps de chargement sont fréquentes, nous avons utilisé des modèles de chaines de Markov cachées suivant un processus de Dirichlet hiérarchique (HDP-HMM) pour enrichir notre modèle et ainsi augmenter le taux de prédiction correspondant. L'évaluation de notre modèle basé sur un arbre de décision sur des sites Web jamais mesurés montre que nous pouvons prédire correctement la qualité de la navigation Web. Ces travaux visent ainsi à permettre aux opérateurs de réseaux et aux fournisseurs de services d'augmenter la Qualité de Service (QoS) offerte à leurs clients.

Title: Future Internet Metrology : Characterization, Quantification and Prediction of Web browsing Quality

Keywords : Web Browsing, Artificial Intelligence, QoE, QoS, Prediction.

Web browsing is one of the main Internet services where a wide set of actors are involved and evolves constantly. New Internet protocols, Content Delivery Networks (CDN) or web browsers' evolutions are meant to improve web pages' loading times. In order to better understand the Quality of Experience (QoE) perceived by end-users, it is of prime importance to identify how web pages' content is composed and delivered as well as providing a relevant QoE metric. In this thesis, we have designed a new tool, Web View, meant to perform automatic Web browsing and measures several aspects of the Web browsing eco-system. We have also introduced a new web metric, the Time for Full Visual Rendering (TFVR). From more than 18 Trillion measurements performed over 2.5 years on the Top 10,000 Alexa websites, we have used statistical techniques to identify the key parameters qualifying and quantifying web browsing quality. This set of factors has been confirmed by a machine learning process, which gives as output the set of rules to predict web pages' loading times. For websites where fluctuations in loading times happen regularly, we have used the Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) to enrich our rules-based models in order to increase the correctness in prediction rates. The evaluation of our decision tree-based model on never assessed websites shows that we can correctly predict web browsing quality. This work aims at helping network operators and service providers to increase the Quality of Service (QoS) offered to their customers.