Live Evaluation of Quality of Experience for Video Streaming and Web Browsing

Louis Plissonneau
Orange Labs, France
louis.plissonneau@orange.com

Heng Cui
Eurecom, France
heng.cui@eurecom.fr

Ernst Biersack
Eurecom, France
erbi@eurecom.fr

***Abstract*—Web browsing and Video download are two major applications of the Internet. To evaluate Quality of Experience (QoE) of these applications, we have developed two tools: (i) pytomo, a tool that executes video download and emulates playback as performed by the Flashplayer plugin. A live web interface for visualisation also gives user information about main indicators for troubleshooting. (ii) CPM, a Firefox plugin that records main events during Web page download to perform a "post-mortem" Critical Path Analysis, indicates to the end-user or Web developer which objects of a given Web page determines the overall page load time.**

I. Introduction

Nowadays, web-driven content represents the majority of Internet traffic with the surge of video sharing sites, such as YouTube. This traffic is usually interactive and studies show that the perception of the service by the user influences a lot how web browsing or video streaming are used. Moreover typical QoS metrics such as loss rate or RTT are not sufficient: one needs to get as "close as possible" to the client in order to evaluate the Quality of Experience (QoE) as seen by the end-user.

In this demo, we focus on the evaluation of Quality of Experience (QoE) for Web Browsing and Video Streaming which are two main Internet applications. The approach taken is to perform measurements from an end-user computer so that end-to-end QoE is evaluated. The tools presented here allow the evaluation of the main factors influencing QoE but also offer methods for self-diagnostic of users' Internet access.

For video streaming, we focus on the number of video stalls during visualisation. We also provide main QoS metrics in order to investigate the selection of video cache servers used by video streaming sites. As for web browsing, an analysis of the objects downloaded on the page is performed to derive a critical path with the overall page load time as main QoE indicator. The results of this analysis can be used to ask What-IF questions such as the impact of changing the round-trip time on the overall page load time.

II. Video Streaming

A. Video Streaming QoE

The main QoE indicator in the case of HTTP video streaming is the number of interruptions in the playback of a video. Indeed, this type of streaming is not adaptive: in case of insufficient throughput, the video will stall and the download continues until there is a sufficient amount of data for resuming playback.

To address the challenge of measuring HTTP video streaming QoE, we have developed an active monitoring tool called `pytomo` [2]. The core part of our tool consists of a model of video playback obtained through reverse engineering of the flash video player used by YouTube and DailyMotion (two of the most popular video streaming sites). We not only collect QoE indicators but also QoS indicators that allow us to figure out the delivery policies of the video streaming site analysed. The tool has been in operational use within an ISP's network for several months. We previously used this tool to better understand the cache selection of YouTube videos and its impact on QoE [1].

B. Demonstration

Our demonstration will consist of multiple live pytomo crawls using the Internet accesses of different ISPs and a visualisation of results for each of the accesses. The visualisation interface allows us to also dig into the redirection process for viewing requests. We will perform live crawls of the most popular videos of YouTube and DailyMotion and we will show the evolution over time of QoE across different access types (wired and wireless), see Fig. 1.

III. Web Browsing

A. Web Page Download Performance Analysis and Prediction

We propose a novel method for the performance analysis web page downloads based on the idea of finding **Critical Path** of a given web page. Our method preforms a passive analysis of multiple instances of browsing experiences for the same web page. This analysis helps explain page rendering performance and predict changes to both, the dependencies of the objects in that Web page or the network conditions affecting download performance of the objects.

There are two main steps of our methodology: first, we use some basic header and timestamp information to infer possible dependencies between each downloaded objects; second, we use the delay information to draw the "critical path" on the inferred dependency graph.

Dependency graph in our algorithm is represented by a Directed Acyclic Graph (DAG) where node means the downloaded objects and arrow is parental relationships between each other. To draw such graph of a given web page, we

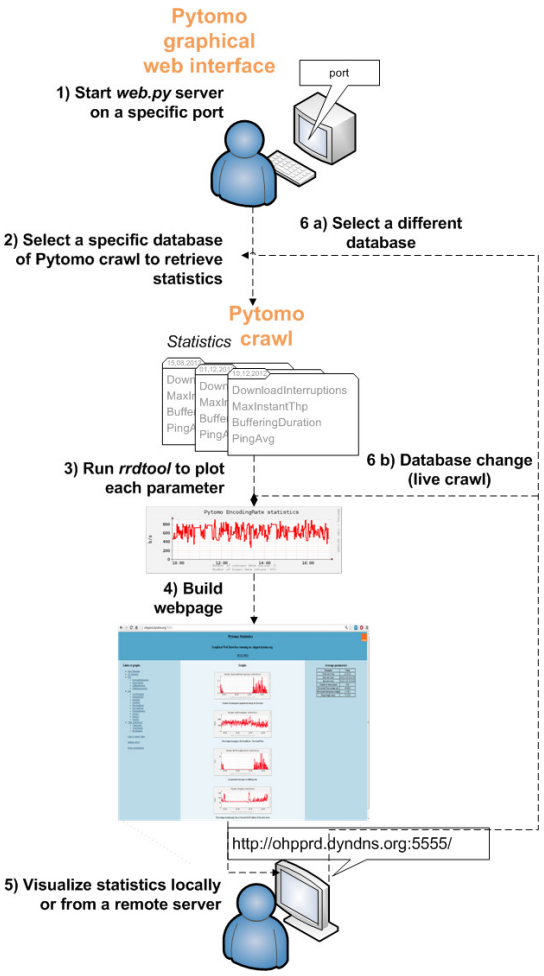

Fig. 1: Workflow of the Web Streaming Demo

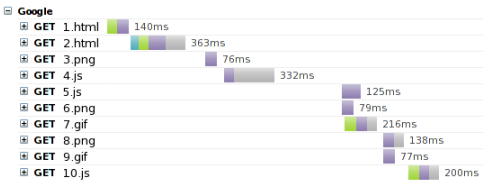

Fig. 2: Timeline of Object Downloads for www.google.com Page

define several rules to infer the parental relationships between each object. For example, we use basic HTTP headers such as `Referer` or `Location` to infer the obvious referenced parental relationships; we also use statistical method to check the objects' starting and ending download time.

B. Demonstration

We show an example by `google` home page. We use Firefox as the browser with a customized plugin to record each objects downloading activity. Fig. 2 shows an example of the object downloads of the Google page generated by the HAR Viewer (http://www.softwareishard.com/har/viewer/). Fig. 3 shows the results by our CPM algorithm for the same

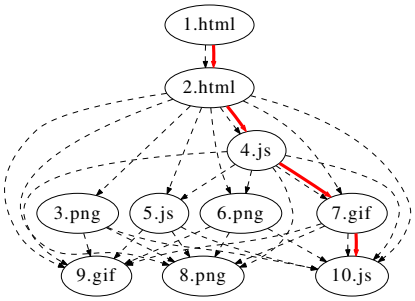

Fig. 3: Critical Path Analysis Example for www.google.com Page

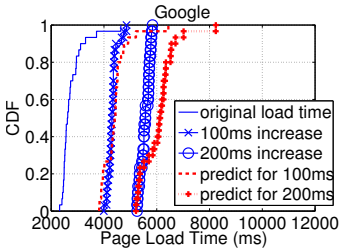

Fig. 4: CDF of Page Load Time

page (red thick lines indicate the selected critical path).

We see that the page consists of 10 different objects with diverse content types. The first request is always for www.google.com (1.html), then, redirect to the main file www.google.fr (2.html) of this current page. In this case, from Fig. 3, 1.html is always the "root" of the page and has direct parental relationship to 2.html. When the browser receives enough data from the main file, it downloads other objects. In this case, we can always see a parental relationship between 2.html and other embedded objects. For the whole web page, objects with relative larger downloading delays are normally selected in the critical path which is useful to discover the page performance bottlenecks.

We will also demonstrate Web page performance *prediction* based on updating the inferred critical path. We will demonstrate how we can predict the impact of a change in RTT on the overall page load time. We use one Linux PC to automatically browse the Google home page for multiple times and measure their page full load time as original load time; for each browsing, we update the object delays by the measurement of the browser plugin; we then compute a new critical path as shown in Fig. 3 and we use total delay traversing the new path as the estimation of the performance under different *what-if* scenarios. As a validation, we use netem to manually increase client outgoing packet delays and browse the same page under different scenarios. Fig. 4 shows that the predictions pretty closely match the measured page load times.

REFERENCES

[1] L. Plissonneau, E. Biersack, and P. Juluri. Analyzing the impact of youtube delivery policies on user experience. In *ITC 24*, 2012.
[2] Pytomo. http://code.google.com/p/pytomo/.