Measuring Web Speed From Passive Traces

Martino Trevisan
Politecnico di Torino
martino.trevisan@polito.it

Idilio Drago
Politecnico di Torino
idilio.drago@polito.it

Marco Mellia
Politecnico di Torino
marco.mellia@polito.it

ABSTRACT

Understanding the quality of web browsing enjoyed by users is key to optimize services and keep users' loyalty. This is crucial for Internet Service Providers (ISPs) to anticipate problems. Quality is subjective, and the complexity of today's pages challenges its measurement. `OnLoad` time and `SpeedIndex` are notable attempts to quantify web performance. However, these metrics are computed using browser instrumentation and, thus, are not available to ISPs.

PAIN (PAssive INdicator) is an automatic system to observe the performance of web pages at ISPs. It leverages passive flow-level and DNS measurements which are still available in the network despite the deployment of HTTPS. With unsupervised learning, PAIN automatically creates a model from the timeline of requests issued by browsers to render web pages, and uses it to analyze the web performance in real-time. We compare PAIN to indicators based on in-browser instrumentation and find strong correlations between the approaches. It reflects worsening network conditions and provides visibility into web performance for ISPs.

1 INTRODUCTION

Metrics related to users' Quality of Experience (QoE) are key to understand how end-users enjoy the web. Such metrics are of prime importance to all actors involved in the service delivery. From Content Providers, which need to monitor users' satisfaction to maintain or increase their user base, to Internet Service Providers (ISP), which need to be aware of web performance to actuate when network factors affect web browsing experience [6].

Users' QoE is is intrinsically subjective, thus hard to be fully assessed and quantified. Previous works [1, 4, 5] have proposed objective metrics that have been shown to be correlated with users' Mean Opinion Score (MOS), even if a model to predict MOS is still hard to get [3]. These metrics however either are computed at the server-side (i.e., available only to Content Providers) or require ground truth from in-browser instrumentation (i.e., not scalable for the monitoring of a large number of sites at ISPs). Passive solutions to provide visibility into web performance are rare, and generally based on traffic payload to reconstruct web pages [6].

2 THE PAIN SYSTEM

We introduce PAIN (PAssive INdicator), a system to monitor web page performance using passive traffic logs, as typically available in ISP networks. The deployment of encryption (e.g., HTTPS) makes solutions that reconstruct web sessions from payload [1, 4, 6] no longer effective. PAIN relies only on L4-level statistics (e.g., Netflow), annotated with DNS information [2] to compute a synthetic indicator of the web page rendering time.

The intuition behind PAIN is very simple. Once users reach a website, their browsers open many flows to different servers to fetch HTML objects, scripts and media content. We call the Fully Qualified Domain Name (FQDN) associated with the first contacted server the *Core Domain* and the remaining contacted FQDNs the *Support Domains*. An example is provided in Fig. 1, which illustrates with colored arrows the moment in which flows to support domains appear after a visit to the core domain *www.nytimes.com*. Given core domains of interest, PAIN automatically learns contacted support domains, as well as the typical order in which such flows appear in the network, creating *groups of support domains*. In the example, PAIN learns 4 groups from the observed network traffic. PAIN then considers the delay to observe flows of each group a performance indicator. It uses visits to the website from all users to (i) observe probable patterns; (ii) identify checkpoints that model the download process; and (iii) measure the delay to pass checkpoints, i.e., automatically building a benchmark.

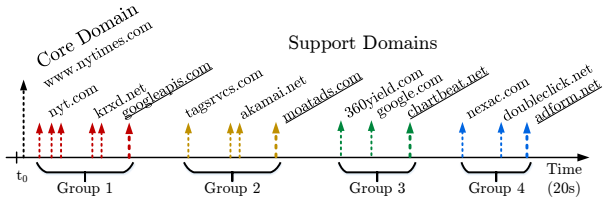

Figure 1: Flows in a *nytimes.com* visit. We use the time to contact support domains to monitor performance.

PAIN is an unsupervised system composed by two blocks (see Fig. 2). The *Model Learning* module analyzes flow records exported by monitoring devices and creates a model for each core domain of interest, i.e., it discovers and clusters support domains associated to specific websites. It must be periodically updated (e.g., monthly), to cope with changes in web-page structure. The *Metric Extraction* module extracts the actual performance metrics using the previously built models. All algorithms are $O(n)$ with respect to the input size (i.e., number of flow records) and are implemented in Apache Spark for scalable processing.

2.1 Input data

PAIN expects two input:

(i) Core domains: It is a user-defined input containing core domains the ISP is interested in monitoring, such as popular websites accessed by users of the network, or generic lists (e.g., Alexa). PAIN operates with up to L4-level measurements (e.g., source and destination IP addresses as well as TCP port numbers), and thus ISPs must specify only the domains to be monitored, and not full URLs. This allows PAIN to work with encrypted traffic, where domain names are still visible;

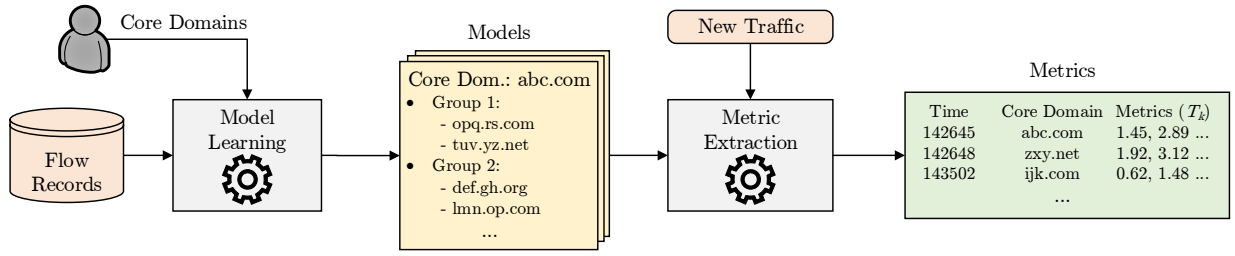

Figure 2: Architecture of PAIN. It learns and clusters support domains using flow records and a list of target core domains. The resulting groups are used to extract indicators of the website performance.

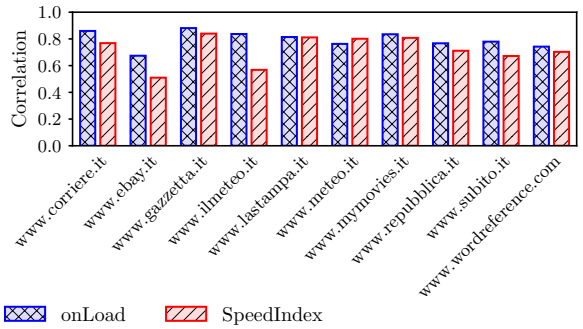

Figure 3: Spearman's correlation of PAIN index with `onLoad` and `SpeedIndex` for analyzed websites.

(ii) Flow records annotated with DNS information: they are captured in the network, by means of flow exporters. PAIN expects ordinary flow records (i.e., the usual 5-tuple), enriched with the domain names used by clients to contact servers. Different methodologies can be used to annotate flow records with that information, such as the methodology presented in [2], which leverages DNS traffic.

2.2 Model learning

The Learning Module models the arrival of flows opened by the browser to render web pages. The first task is to learn the support domains associated with each core domain. PAIN learns that by focusing on the flows *commonly* occurring after core domains. Then, given that flows to download objects while rendering a page vary from visit to visit (e.g., because of caching, persistent connections, modification in the content, etc.), PAIN analyzes the order in which *groups* of support domains ypically appear.

2.3 Performance metric extraction

The metric extraction module analyzes new traffic to provide the performance metrics. Intuitively, we measure the time at which flows in each group are observed. Visits to a group are considered completed when the last flow in the group is observed. This last visit is a *checkpoint*, and the relative visit time to checkpoints are recorded as performance metrics.

3 CORRELATION WITH CLIENT-SIDE METRICS

We validate PAIN in a controlled environment, in which a testbed is instrumented to browse web pages and collect classic performance metrics. We compare measurements collected by the two independent approaches. We show that PAIN is able to spot changes in network conditions, reporting quality degradation when the page load time increases. PAIN metrics are strongly correlated with objective metrics computed based on client instrumentation.

We formally quantify the correlation of PAIN index with `SpeedIndex`[1] and `OnLoad` time. Fig. 3 shows Spearman's rank correlation coefficients for `onLoad` time (blue) and `SpeedIndex` (red), separately for 10 Italian websites. Correlation coefficients are positive and very high (i.e., ≥ 0.5), ranging from 0.54 for *www.ebay.it* (`SpeedIndex`) to 0.90 for *www.gazzetta.it* (`onLoad`). Most values are close to 0.8 for both metrics. That is, PAIN checkpoints are strongly correlated with both objective metrics for different sites. For comparison, the correlation between `SpeedIndex` and `onLoad` ranges in [0.71,0.91]. Results reinforce that PAIN works well as a proxy to quality monitoring, providing strong indications without client-side instrumentation.

REFERENCES

[1] Athula Balachandran, Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Srinivasan Seshan, Shobha Venkataraman, and He Yan. 2014. Modeling Web Quality-of-experience on Cellular Networks. In *Proc. of the MobiCom*. 213–224.

[2] Ignacio Bermudez, Marco Mellia, Maurizio M. Munafò, Ram Keralapura, and Antonio Nucci. 2012. DNS to the Rescue: Discerning Content and Services in a Tangled Web. In *Proc. of the IMC*. 413–426.

[3] Enrico Bocchi, Luca De Cicco, Marco Mellia, and Dario Rossi. 2017. The Web, the Users, and the MOS: Influence of HTTP/2 on User Experience. In *Proc. of the PAM*. 47–59.

[4] Enrico Bocchi, Luca De Cicco, and Dario Rossi. 2016. Measuring the Quality of Experience of Web Users. In *Proc. of the Internet-QoE*. 37–42.

[5] Pedro Casas, Michael Seufert, Florian Wamser, Bruno Gardlo, Andreas Sackl, and Raimund Schatz. 2016. Next to You: Monitoring Quality of Experience in Cellular Networks From the End-Devices. *IEEE Trans. Netw. Service Manag.* 13, 2 (2016), 181–196.

[6] Sandvine. 2017. Measuring Web Browsing Quality of Experience. (2017). https://www.sandvine.com/technology/web-browsing-quality-of-experience.html.

[1] https://developers.google.com/speed/docs/insights/about