Trouble Shooting Interactive Web Sessions in a Home Environment

Heng Cui, Ernst Biersack
Eurecom
Sophia Antipolis, France
{firstname.lastname}@eurecom.fr

ABSTRACT

Home clients can use their access to the Internet for different purposes such as file sharing via P2P applications, gaming, or Web browsing; the last one is the focus of this work. When browsing the Web, the time elapsed between the click on a URL and the rendering of the Web page, referred to as page load time, is the key performance metric. When the page load time is higher than a few seconds, the user experience suffers significantly. We have developed a three-tier system that (i) captures in the browser the events necessary to measure the page load time (ii) captures at the network access all incoming and outgoing packets, and (iii) correlates the measurements made at different machines. The capture at packet level allows us to compute the contribution of the various steps that affect the page load time such as DNS resolution, server response time, data transfer time. Correlating the observations made at different machines that share a major part of the network elements can help identifying the root causes for high page load times. We will present the architecture of our system and some examples that illustrate its use.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies, Measurement techniques

General Terms

Measurement, Experimentation

Keywords

Web Browsing, Performance Evaluation, Troubleshooting

1. INTRODUCTION

"The only way to know how customers see your business is to look at it through their eyes."

Daniel R. Scroggin

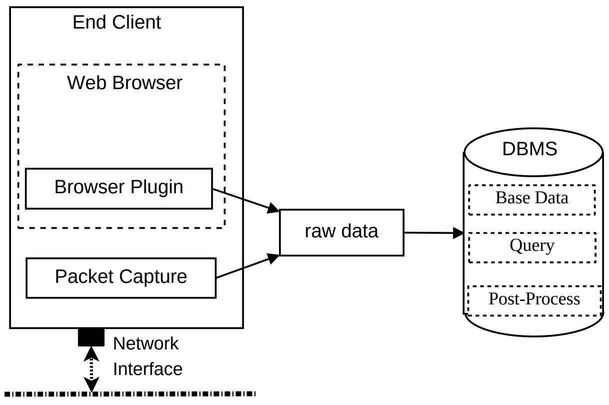


Figure 1: Architecture

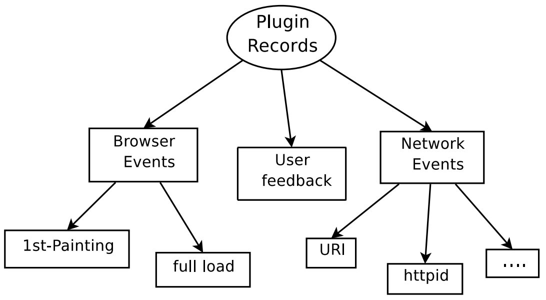
Web browsing is a very common way of using the Internet access as it allows to access to a wealth of information. Examples for Web browsing are consulting a Wikipedia entry, accessing a news site, doing some on-line shopping, or viewing user generated content such as YouTube or Dailymotion. In all these cases a low response time is key for good user experience. Low level metrics such as packet loss or round trip times are not able to measure user experience, but may be useful to explain performance problems. For this reason, we propose a methodology that puts an '*eye*' on the user's screen while browsing the web page, and combines information obtained from a web browser plugin with a low level packet capture system.

While the measurements at browser level allow us to *detect* problems, the measurements at packet level allow us to *explain* problems if they are caused by network elements along the path from the client to the server or by the server itself.

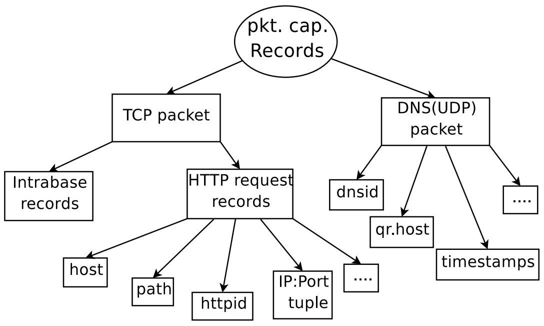
2. TROUBLE SHOOTING SET-UP

2.1 Browser Plugin

As shown in Fig. 1, our troubleshooting setup is composed by three parts: a plugin for the web browser; a packet-level capture and a database repository (DBMS). We use Firefox as our web browser due to its rich APIs [3]. The browser plugin tracks some critical events during a web session that are related to the user's perceptual experience. We track paint events of the browser to measure the **waiting time** that elapses between the user clicking on a Web page and



(a) Plugin Records



(b) Packet-Level Capture Records

Figure 2: Information Logged

the content of the Web page being displayed. We use the plugin to add event-listeners inside the browser:

- The **first painting event** by the browser tells how long the user needs to wait for the *first visual impact.* We call the time interval between the user clicking on the URL until the first painting event the **first impression time**. A large first impression time means that the user has to wait for a long time until he sees anything of the Web page, which will result in a poor user experience.
- The **full page load event** measures how long it takes until the entire content of a Web page, which may consist of several tens of different elements is displayed. We call the time interval between the user clicking on the URL until the full page load event the **full load time** T_{full}.

We use the term **waiting time** to refer to either the first impression time or the full load time and we use the term **web session** to the time interval between the user clicking on a URL and the requested Web page being fully loaded.

For each Web element that is loaded, the plugin records a certain number of additional information such as URI etc. that we do not explain here for space reasons.

2.2 Packet level capture

We perform a network level capture using wireshark or tcpdump to record all the data packets exchanged during a web session. For convenience we write all the raw data packets into a database management system. We use PostgreSQL [4] as our DBMS, the PL/PgSQL as its programming language, and the Intrabase system shown in Fig. 2(b) for TCP connection analysis is developed by Siekkinen [15].

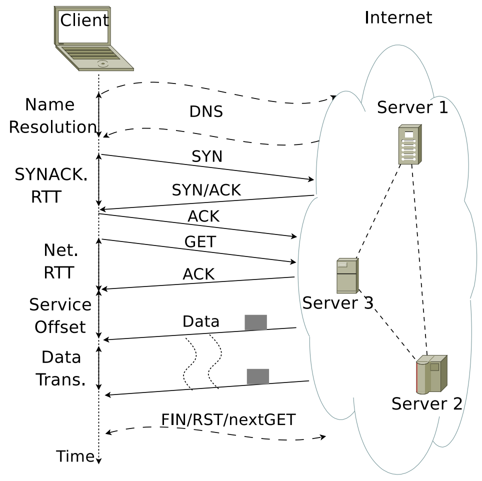


Figure 3: Metrics for downloading one Web element.

Since the plugin and packet capture are working simultaneously but independently, we need a way of "splicing together" the observations made: each time an HTTP request is initiated by the browser, the plugin randomly generates a number called **httpid** that is inserted into the HTTP request header. Since the *httpid* value will be recorded both, by the plugin and packet capture, it can be used to correlate the information obtained from plugin and packet trace. The details of how to reconstruct a web session from the packet trace are discussed in the next section.

3. MEASUREMENT

We first explain which performance measures we extract from a packet level trace and then report on two different experiments where we repeatedly access the same Web page (`www.youtube.com`) during one day.

3.1 Metrics and Methodology

A typical web page normally contains tens, up to hundreds of elements that make up the whole page. To fully render the Web page, the browser needs to load all these elements; in the extreme case, this will require as many name resolutions and TCP connections as there are elements. In the case of YouTube, the whole web page consists of about 30 elements stored on about 10 distinct servers. The typical procedure for loading one element is shown in Fig. 3: The time elapsed can be broken down into the following metrics:

- Name Resolution (DNS): The time elapsed between the first DNS query and first corresponding response with valid IP address(es) defines t_{DNS} .
- TCP Handshake RTT (SYNACK RTT): Since our measurement point is at the client side and all of TCP connections are initiated by the client, we define t_{TCPRTT} as the time between the first SYN packet sent by the client and its corresponding first SYN-ACK packet received from the server.
- Network RTT (Net.RTT): The time between the first data packet (normally carrying a GET request) sent by the client and its first corresponding ACK is referred to as t_{NetRTT}.

- Service Offset: When the ACK packet from the server in response to the data packet that carried the GET request does not contain any payload, there will be a time-gap between this ACK and the first data packet from the server, which is referred to as t_{Offset}. Since this gap between client request and data reception is normally caused a delay due to server processing, we call this gap as service offset.
- Data Transfer Period: The time between the reception of the first and last data packet containing the data of a Web element is referred to as $t_{DataTrans}$.

We measure two types of RTT, namely the TCP handshake RTT and the Net.RTT, which is useful in situations such as the following:

- In a whole web session, one TCP connection can be reused to fetch multiple elements, in which case we will not obtain a sample for t_{TCPRTT} for all of the element downloads.
- If the client is not directly communicating to the server because of an intermediate proxy, the SYNACK packet is usually come from the proxy and not the server and t_{TCPRTT} does not measure the round trip time between client and server.

When we analyze the the packet trace for one Web session we get a time series of values t^i_m where m denotes any of the metrics defined above (e.g. DNS, TCP Handshake RTT, NetRTT, etc.) and i refers to the *i-th* element loaded in that Web session, Given such a time series, we define the weight w_m of metric m as:

$$\frac{\sum t^i_m}{(\sum t^i_{DNS}+\sum t^i_{TCPRTT}+\sum t^i_{NetRTT}+\sum t^i_{Offset}+\sum t^i_{DataTrans}}$$

Based on the above weight definition, we use the full page load time T_{full} measured by the plugin to compute the **weighted breakdown** for metric m as:

$$breakdown_m = T_{full} \times w_m$$

where m is a metric such as DNS, Net.RTT, etc. From this breakdown definition, we can easily visualize the total 'weighted' contribution of each metric. This breakdown definition is similar to the one in [11]. However, [11] uses different metrics and solely relies on the packet trace.

3.2 Measurement Results

In this section, we illustrate our approach via two experiments.

3.2.1 *Wireless vs. Wired Client*

The experiments are done in a private home on both, a wired and wireless Linux PC sharing the same network access. We run the tests on the two PCs simultaneously. We use firefox version 3.6.12, and configure the DNS query with IPv4 and leave the other settings of firefox as default ones. In order to make the client fetch the web content from the server, we automatically clear the browser cache after each browsing.

Load times of the YouTube sessions, where the main page of YouTube is requested, are shown in Fig. 4(a)-(b). We see that most of the YouTube web sessions in both, the wireless and wired cases, achieve load times of two seconds or less; However, in the wireless case, there are some outliers with very high values: The first impression time sometimes being larger than 5 seconds and the page full load time as large as 10 seconds. In Fig. 4(c)-(d) (best viewed in color), we see that data transfer period dominates in these wireless sessions and some of them also have large handshake time. The explanation for these large values can be obtained looking at Fig. 4(e) and Fig. 4(f), which depict the Net.RTTs. For the wired case we see very stable values in the range between 20 – 100 msec. However, in the wireless case these values increase to 3 seconds. There are two possible explanations: (i) the wireless access point is overloaded or (ii) the quality of the wireless link is poor, which results in many link layer retransmissions.

3.2.2 *Student residence*

In the second experiment we have a client PC in a student residence. The client is connected via Ethernet and is sharing the access link from the residence to the Internet with a large number of other clients.

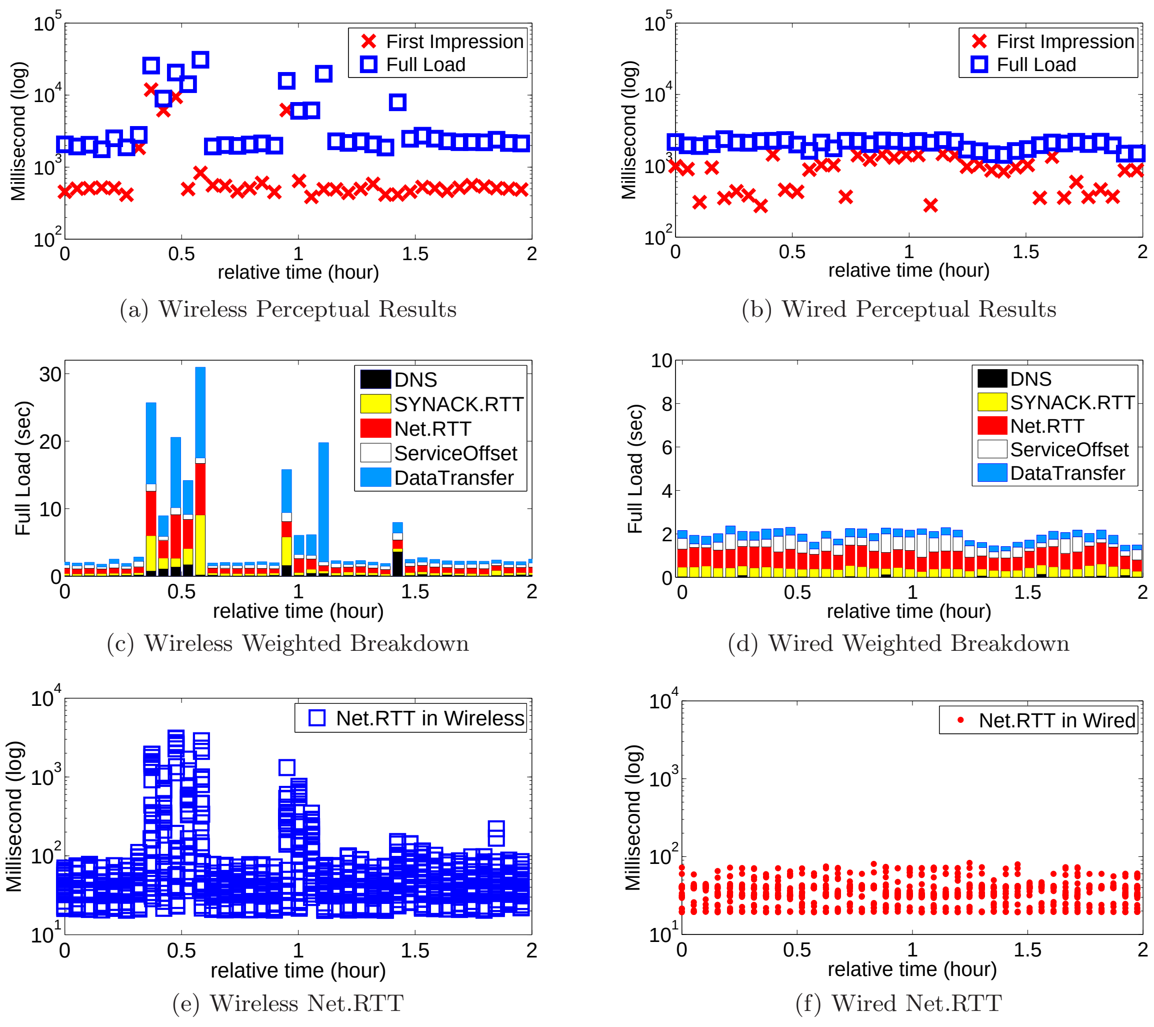
Fig. 5(a) depicts the page load times as experienced by the client. We see that the full load times are in the order of 5 seconds or higher. Some of the first impression times are also in the order of several seconds.

Fig. 5(b) shows the contributions of the different metrics that make up the load time of a Web page. What is striking are the high values for t_{DNS} and t_{TCPRTT} relative to the time $t_{DataTrans}$ it takes to download the data of the element. In Fig. 5(c) we plot the DNS response time and TCP handshake time values. We see that the values are either below 200 msec or above several seconds. Since the default timeout value for DNS and TCP SYN retransmission in the Linux client is 5 seconds and 3 seconds respectively, we can infer that the high values observed are due to packet loss and retransmission after timeout. Fig. 5(d) shows the retransmission rate for all the YouTube sessions. All sessions suffer from packet retransmissions, with a retransmission rate ranging from 1% up to 8%. Also, not shown here, most of the retransmissions are triggered by timeout and not by fast retransmit.

Indeed, packet retransmissions do not only happen to YouTube sessions in the student residence. Fig. 6(a) shows the retransmission rate of browsing `www.dailymotion.fr` measured at the same time period as YouTube case. We can find a retransmission behavior similar to the one for the YouTube sessions. Fig. 6(b) also shows the CDF of retransmission rate for some other selected French web sites tested in the same student residence on another day. These measurements last over 9 hours. From the CDF, we can easily discover a similar retransmission behavior among all web sessions of different domains, and lasting over the whole measured period.

4. RELATED WORK

Our approach is inspired by the work of Agarwal et al. [6] and Siekkinen [15]. Agarwal et al. developed the WebProfiler system that is composed of a browser plugin, WebProfiler service, a local and central database to diagnose problems in the web service access. Siekkinen [15] developed the Intrabase system that uses database technology to efficiently post-process packet level traces. However, the aim of our work is different, namely we try to explain why Web access is slow, while the Intrabase system does a root cause anal-



(a) Wireless Perceptual Results (b) Wired Perceptual Results (c) Wireless Weighted Breakdown (d) Wired Weighted Breakdown (e) Wireless Net.RTT (f) Wired Net.RTT

Figure 4: YouTube Sessions of wired and wireless client.(starting around 2011-02-22 17:30)

ysis of long TCP connections. In our case the connections are typically short and we simply use Intrabase to analyze the loss and retransmission behavior of TCP connections.

Huang et al. [11] perform experiments in a controlled environment to study the performance of Web browsing on a smart-phone. There are also some related works that instrument the Web browser using a plugin such as dynatrace [1] and firebug [2]. These plugins can easily visualize the web page structure and browser's behavior in rendering the page. The concepts of first impression and full load are also proposed in dynatrace [1]. However, both systems do not perform any packet level analysis.

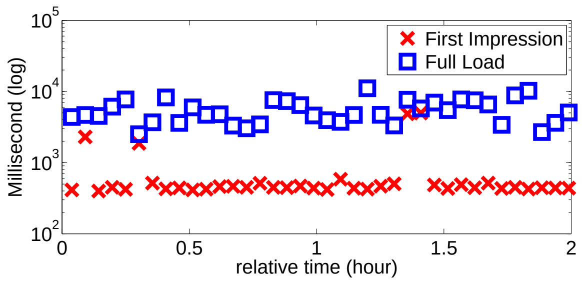
Plenty of systems are also proposed for home network management and trouble shooting. Joumblatt et al. [12] propose HostView, which collects a set of end-host data such as network traffic, basic network configurations, and subjective user feedback to study the performance degradations. Calvert et al. [8] propose their a system called HNDR for *"general-purpose"* management and trouble shooting, and describe challenges and requirements for such system. Aggarwal et al. [7] propose NetPrints and Karagiannis et al. [13] HomeMaestro. Both systems are based on sharing knowledge between home users to solve problems. NetPrints focuses on the network configuration errors. HomeMaestro addresses performance issues caused by contention within home-networks, and combines observation with control in the sense of modifying the resource allocation to the different flows.

5. DISCUSSION

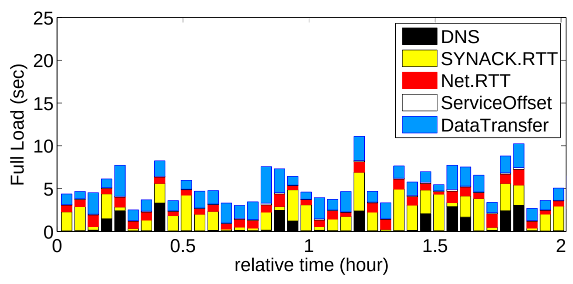
In this paper, we propose a method to combine observations captured at Web browser level with observations at packet level to detect and explain high web access times. However, so far we have done only the first step and more work is needed to better understand the limitations and the potential for extensions of the approach.

5.1 Scope and limitations

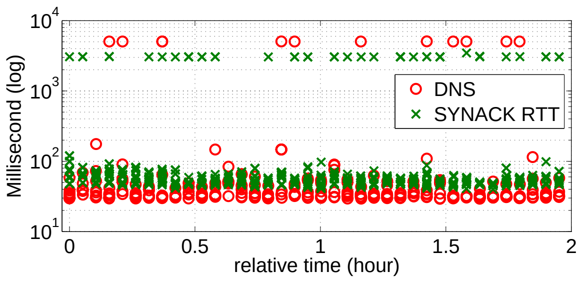
We use two browser events to estimate the user satisfaction. We need to do some end user studies to confirm that real user's satisfaction can be measured this way. Since a user already sees some of the content of a Web page before the full load time has elapsed, we need to deeper investigate the relationship between the full load time and the subjec-



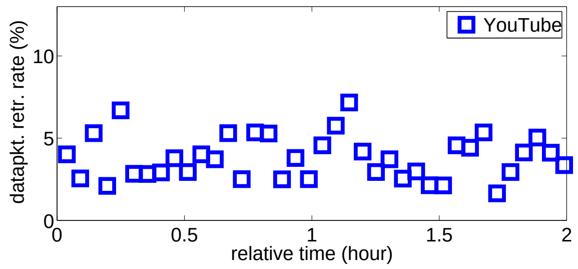
(a) Perceptual Results



(b) Weighted Breakdown

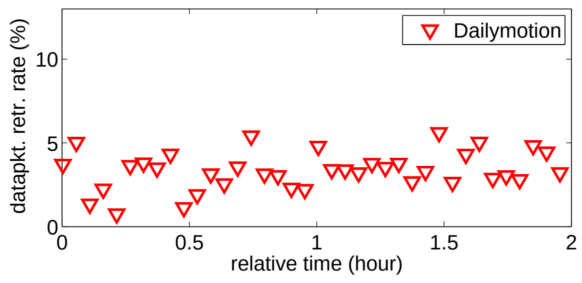


(c) DNS and TCP HS Time

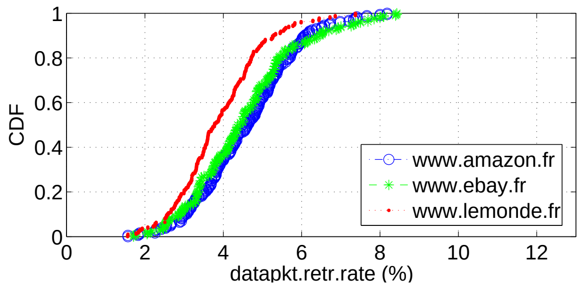


(d) Packet Retransmission Rate

Figure 5: YouTube Sessions at student residence.(starting around 2011-02-20 00:00)



(a) dailymotion.fr Measured at 2011-02-20 00:00



(b) List of French Web Sites Measured at 2011-05-05 00:00

Figure 6: Data retransmissions at student residence

tive user experience. For this purpose, we have already implemented a user feedback bottom (c.f. Fig. 2(a)) that allows the user indicate whether he is satisfied or dissatisfied with the speed at which the current Web page was displayed. With the user feedback we also record the timestamp to be able to relate the feedback to the events preceding the feedback.

We have positioned our work as trouble shooting in a *home environment*. We think the home is a very good starting point, since in many homes we have multiple networked devices such as PCs, laptops, smart phones or tablets that can all be used to access the Web. Correlating the observations made on the different devices in the same home poses not much of an issue in terms of privacy. However, sharing information for trouble shooting among devices in *different homes* requires first a careful evaluation of potential privacy violations. In this context, the work on differential privacy by Drowk [9] can provide a solid foundation for reasoning about how to share information among clients in different homes while maintaining privacy guarantees.

Our model of how a Web page is loaded is quite simple and we currently do not consider issues such as (i) caching, (ii) two clients having different browser configurations, and (iii) the impact of parallel TCP connections or pipelining.

Packet traces only allow us to "observe" the path between client and server but do not provide insights into events *at the client itself*. However, CPU overload or main memory shortage leading to disk thrashing, which are not captured in our approach, can also cause large page load times.

As pointed by Yahoo![1], the scripting style in css or javascript can affect the browser rendering behavior and impact latency. In the experiments presented in this paper, we sidestep this issue by downloading the same Web page over again.

5.2 Extensions

Our work is part of an ongoing European project called **Figaro** on the design of the home network of the future. This project foresees the deployment of a so called **home gateway** that mediates between the home and the external

[1]http://developer.yahoo.com/performance/rules.html

world. In this case it will be interesting to study how much of the data collection can be moved to the home gateway. If all the data collection could occur in the home gateway, there may be no need for browser plugins and packet capture in the end-devices; However, it remains to be determined if all the metrics defined in this paper can be measured in such a gateway centric approach.

6. CONCLUSION AND FUTURE WORK

We have presented the architecture of a tool for trouble shooting performance problems in Web access. The tool relies on measurements at browser and at network level to identify problems and explain their possible causes. The measurements of different end-devices can be correlated to help reduce the number of possible causes and to improve the explicative power. Via two experiments we have demonstrated the use of the tool and the insights it can provide.

However, much work remains to be done:

- The development of such a tool must go hand in hand with more experiments. We need more controlled experiments to validate some of our assumptions and experiments in the "wild" to evaluate the usefulness of the tool.

- Since we want to correlate observations from multiple vantage points (clients), we need to define a distributed architecture for data collection and data sharing. So far, a client collects browser events and packet traces that will be loaded into a central database for post-processing. Such an approach has obvious performance and scalability limitations; therefore, as soon as we have consolidated all the metrics needed for trouble shooting, we plan to compute these metrics "on the fly" at the client side, instead of storing packet headers. We are currently evaluating the feasibility of `tstat` [5] for this purpose.

- Given a set of performance metrics and measurements, we need to automate the diagnosis task. For this purpose, we have started using clustering algorithms to group different Web sessions and identify a small and meaningful set of classes that allow to explain all major performance problems observed. Another approach we plan to explore is the use of probabilistic rational models for root cause analysis [10] [14].

7. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive comments which are useful for improving our future work. This work was partly done as part of the Collaborative Project FIGARO, which is supported by the European Commission under the 7th Framework Program, Grant Agreement Number 258378.

8. REFERENCES

[1] Dynatrace. http://ajax.dynatrace.com/.
[2] Firebug. http://getfirebug.com/.
[3] Mozilla. https://developer.mozilla.org/en/.
[4] Pgsql. http://www.postgresql.org/.
[5] Tstat. http://tstat.tlc.polito.it/index.shtml/.
[6] S. Agarwal, N. Liogkas, P. Mohan, and V.N. Padmanabhan. WebProfiler: Cooperative Diagnosis of Web Failures. In *Proceedings of the 2nd international conference on COMmunication systems and NETworks*, pages 288–298, January 2010.
[7] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. Padmanabhan, and G. M. Voelker. NetPrints: Diagnosing Home Network Misconfigurations Using Shared Knowledge. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, April 2009.
[8] K. L. Calvert, W. K. Edwards, N. Feamster, R. E. Grinter, Y. Deng, and X. Zhou. Instrumenting Home Networks. In *HomeNets, ACM SIGCOMM Workshop on Home Networks*, New Delhi, India, September 2010.
[9] C. Dwork. Differential Privacy. In *Proceedings of 33rd International Colloquium on Automata, Languages and Programming*, 2006.
[10] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic Relational Models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, pages 129–174. MIT press, 2007.
[11] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and V. Bahl. Anatomizing Application Performance Differences on Smartphones. In *International Conference On Mobile Systems*, 2010.
[12] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft. HostView: Annotating End-host Performance Measurements with User Feedback. In *HotMetrics, ACM Sigmetrics Workshop*, New York, NY, USA, June 2010.
[13] T. Karagiannis, E. Athanasopoulos, C. Gkantsidis, and P. Key. HomeMaestro: Order from Chaos in Home Networks. Technical Report MSR-TR-2008-84, Microsoft Research, May 2008.
[14] G. J. Lee. CAPRI: A Common Architecture for Autonomous, Distributed Diagnosis of Internet Faults Using Probabilistic Relational Models. In *Proceedings of the First international conference on Hot topics in autonomic computing*, Dublin, Ireland, June 2006.
[15] M. Siekkinen, G. Urvoy-Keller, Ernst W. Biersack, and D. Collange. A Root Cause Analysis Toolkit for TCP. *Computer Networks*, 52(9):1846–1858, 2008.