Analyzing Cacheability in the Access Network with HACkSAw

Claudio Imbrenda
Orange Labs
claudio.imbrenda@orange.com

Luca Muscariello
Orange Labs
luca.muscariello@orange.com

Dario Rossi
Telecom ParisTech
dario.rossi@enst.fr

ABSTRACT

Web traffic is growing, and the need for accurate traces of HTTP traffic is therefore also rising, both for operators and researchers, as accurate HTTP traffic traces allow to analyse and characterize the traffic and the clients, and to analyse the performance of the network and the perceived quality of service for the final users. Since most ICN proposals also advocate for pervasive caching, it's imperative to measure the cacheability of traffic to assess the impact and/or the potential benefits of such solutions. This demonstration will show a both a tool to collect HTTP traces that is both fast and accurate and that overcomes the limitations of existing tools, and a set of important statistics that can be computed in post processing, like aggregate/demultiplexed cacheability figures.

Categories and Subject Descriptors

C.2.3 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Operations—*Network monitoring*

Keywords

Cacheability; caching; high performance; http analysis; live traffic monitoring

1. INTRODUCTION AND MOTIVATION

Web content is nowadays skyrocketing, causing significant additional infrastructure costs to network operators, therefore, an accurate characterization of the traffic and the users is a fundamental prerequisite for strategic decisions. Moreover, most ICN proposals advocate for pervasive caching; a correct measure of cacheability and traffic reduction is therefore fundamental to assess the impact and/or the potential benefits of such solutions.

Characterization of web traffic usually means logging the IDs of the requested objects along with a timestamp; some further details can be collected almost effortlessly, like the content type and in some cases the size of the object. The methodology of collection varies, and the accuracy of the results also varies consequently.

The next step is analysis or post processing of the logs, in order to extract the required statistics. Cacheability and traffic reduction, as introduced in [1, 4], are very important metrics, as pointed before. In order to quickly and accurately calculate those values, a log is needed containing all the requested objects, the time of the request and the real amount of traffic generated.

There are already some tools that perform HTTP traffic analysis [3, 6, 7, 9], only some of which are publicly available. The performace and the accuracy of the publicly available tools is in general not satisfactory for some kinds of traffic analysis, especially in relation to cacheability.

Tstat [5] performs a packet-level analysis of HTTP connections, this allows it to operate quickly and with a reduced memory footprint, but on the other hand it misses many details. In most cases the Content-Length field of the object is missed because it didn't appear in the first packet of the reply.

Bro is an intrusion detection system, meant to protect a single host, and not to be executed at line speed in an operational network; it is however used in the research community to generate traces of HTTP traffic [2, 8].

Table 1 shows a comparison of HACkSAw with the other publicly available tools performed on the same hardware and on the same 1-hour packet-level trace; 0-length replies are those replies to HTTP requests where the length of the content is either zero or missing, and CPU is the cumulative runtime of all threads. It

Tool	Detected Requests	CPU [sec]	Memory [GB]	0-length replies
Tstat	2 531 210	445	0.3	1 128 109
bro	2 559 056	8033	4.2	424 355
HACkSAw	2 426 391	368	5.8	328 465

Table 1: Performance of bro, Tstat and HACkSAw.

can be seen that bro is too slow and tstat is inaccurate. In both cases both bro and Tstat only report the Content-Length field from the HTTP headers. There are many cases in which such header is unavailable; in cases where the chunked HTTP Transfer-Encoding was used, for example, the total size of the object is not known in advance. HACkSAw manages to be both fast and accurate thanks to its simple, yet effective, implementation; some key features include:

- the full TCP stream is reconstructed, therefore the full HTTP headers are available and the real size of the downloaded object is available.
- no per-packet pattern matching is performed to identify HTTP requests, the payload is skipped, thus significantly lowering the CPU consumption

ICN'14, September 24–26, 2014, Paris, France.
ACM 978-1-4503-3206-4/14/09.
http://dx.doi.org/10.1145/2660129.2660135 .

- multithreading; can scale up almost linearly with more processors

While our tool does require a significant amount of memory, it was successfully used uninterruptedly over a period of one month to collect statistics on a real access network (see [4]).

Only clear-text connections can be analysed, as obviously no dissection of SSL traffic is possible. Although the amount of HTTPS traffic is rising with time, especially since popular websites like Facebook or YouTube started to push in that direction, and therefore potentially rendering this approach useless in the long run, we measured in our observations that, currently, only approximately 15% of the total HTTP traffic is HTTPS.

2. STATISTICS

In addition to the usual statistics collected by other tools, like for example the client ID, the object ID, the hostname or the User-Agent string, HACkSAw also collects many statistics that other tools neglect, like the time between the HTTP request and the HTTP reply or the first byte of content; the indication whether cookies or ETAG headers were used, the size of the headers, and the byte-range in case of range(partial) request.

The results collected by the tool allow to easily compute aggregate statistics as shown in [4]; those aggregate statistics are calculated with a simple post-processing of the output log of the tool, which is in plain text. The relevant statistics that can be calculated easily are:

- Request cacheability (the share of HTTP requests that can potentially be cached in a given timeframe)
- Traffic reduction (the percentage of actual traffic potentially saved assuming all cacheable items are pre-fetched during off-peak hours)
- Virtual cache size (the minimum cache size needed to cache all cacheable content, assuming perfect "oracle" replacement strategy)
- Average number of HTTP requests per connection (many connections perform more than one HTTP request, as per HTTP/1.1)
- Share of requests with cookies and/or ETAG (ETAG headers potentially indicate different content for the same URL; the presence of cookies generally hinders cacheability)
- download completion of requests (for each request, the percentage of bytes of the object actually downloaded by the client; values under 100% thus indicate that the object was not downloaded completely)
- average actual throughput and average latency of requests (time between the first and the last byte of content, and between start of the HTTP request and the first byte of content, respectively)

3. DEMONSTRATION

The tool will be run live on the conference network, as it is as close to the actual access network as we can get; relevant statistics will be computed from the anonymized output of the tool and shown almost live, with a small delay to allow for processing.

Some of the statistics that will be shown (apart from the classical ones like total traffic, number of HTTP requests, etc), are those presented above in section 2.

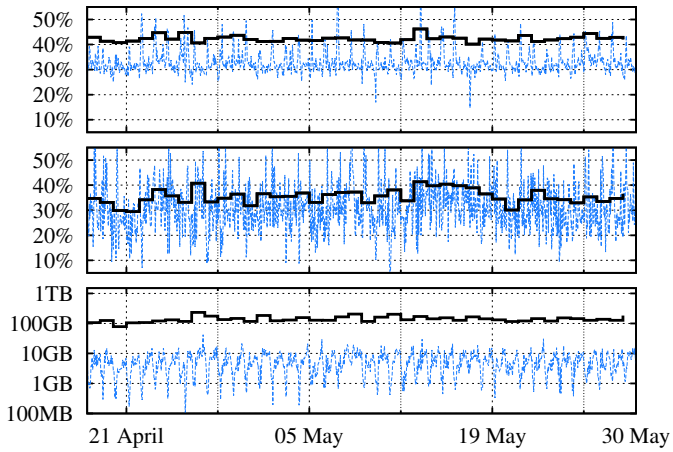

Figure 1: Timeseries of Requests cacheability (top), Traffic reduction (middle) and Virtual cache size (bottom); 1 day aggregate (thick black line) and 1 hour aggregate (thin blue line) calculated on the data gathered in [4].

All statistics will be computed over different timescales, from 1 hour to 1 day; figure 1 shows an example of some of the presentation of the statistics that will be shown in the demo.

4. REFERENCES

[1] B. Ager, F. Schneider, J. Kim, and A. Feldmann. Revisiting Cacheability in Times of User Generated Content. In *Proc.of IEEE INFOCOM*, 2010.

[2] B. Ager, F. Schneider, J. Kim, and A. Feldmann. Revisiting cacheability in times of user generated content. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1–6, March 2010.

[3] A. Finamore, M. Mellia, M. Meo, M. Munafo, and D. Rossi. Experiences of internet traffic monitoring with tstat. *IEEE Network Magazine*, May 2011.

[4] C. Imbrenda, L. Muscariello, and D. Rossi. Analyzing cacheable traffic in isp access networks for micro cdn applications via content-centric networking. In *Proc.of ACM ICN*, 2014.

[5] M. Mellia and al. `http://tstat.tlc.polito.it`.

[6] V. Paxson. `http://www.bro.org`.

[7] B. Ramanan, L. Drabeck, M. Haner, N. Nithi, T. Klein, and C. Sawkar. Cacheability analysis of HTTP traffic in an operational LTE network. In *In Proc. of WTS*, 2013.

[8] F. Schneider, B. Ager, G. Maier, A. Feldmann, and S. Uhlig. Pitfalls in HTTP Traffic Measurements and Analysis. In *Proc. of PAM*, 2012.

[9] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park. Comparison of caching strategies in modern cellular backhaul networks. In *Proc. of ACM MobiSys*, 2013.