Analyzing Cacheable Traffic for FTTH Users Using Hadoop

Claudio Imbrenda
Orange Labs Networks
claudio.imbrenda@orange.com

Wuyang Li
Eurecom Institute
wuyang.li@eurecom.fr

Luca Muscariello
Orange Labs Networks
luca.muscariello@orange.com

ABSTRACT

We present this year (2015) statistics about cacheable traffic in the access network of Orange in Paris for about 30,000 customers served by a fiber to the home subscription. These statistics update some of the results presented in a recent work, which considered only 2 000 fiber users in 2014. The huge amount of data to be processed in the new vantage point made necessary the usage of a hadoop cluster that we have used to process the data and report new statistics in the present paper. The aggregation level at which we observe web traffic allows to draw some conclusions about the feasibility of implementing in-network caching at wire speed.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*

Keywords

Network Traffic Measurements; Caching; Big Data

1. INTRODUCTION AND MOTIVATION

One of the fundamental questions about using in-network caches in the Internet is to show how much data can be really cached, if such data allows to save significant amount of traffic and if the total cacheable data requires a memory that can be implemented with a reasonable cost. The size of that memory would also determine the technology and then the access speed, which has a direct implication on the location of such memory in the communication path. Smaller memories can be very fast and installed in the forwarding engine of a router and accessed at wire speed, while very large memories are typically implemented with low rate technologies that can only be installed out of the data path and accessed at lower rate. The answer to this question depends on many factors but mostly on the nature of the traffic demand coming from the end users. It also depends on the network location where such demand is observed. In this paper we position our vantage point in the access networks of Orange in Paris to observe about 30 000 customers, served by a GPON (Gigabit Passive Optical Network) based FTTH (Fiber To The Home) access, with a per user maximum downlink rate spanning from 300Mbps to 1Gbps. In [3] all HTTP requests and replies were processed online in the probe itself, while the new scale that we face in this work required the use of a hadoop cluster to compute the relevant statistics online. The present work takes into account several ameliorations of the measurement methodology employed in [3], as it also takes into account identification of objects that are retrieved using HTTP chunked transfer encoding and range requests in order to detect partial data transfers or non overlapping pieces of a same object, that is identified by the same HTTP URI. Cacheability and traffic reduction, as introduced in [1, 3], are very important metrics, as pointed before. In order to quickly and accurately calculate these 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[7, 6, 2, 5], some of which are unavailable. The performance 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[4] 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.

Only clear-text connections can be analyzed, as obviously no dissection of SSL/TLS traffic is possible for us. 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, approximately 35% 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, our tool, called 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, the byte-range in case of range(partial) request, and the list of all present headers (without their values). The results collected by the tool allow to compute aggregate statistics as shown in [3]; those aggregate statistics are calculated with a non-trivial 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*

ICN'15, September 30–October 2, 2015, San Francisco, CA, USA.
ACM 978-1-4503-3855-4/15/09
http://dx.doi.org/10.1145/2810156.2812602.

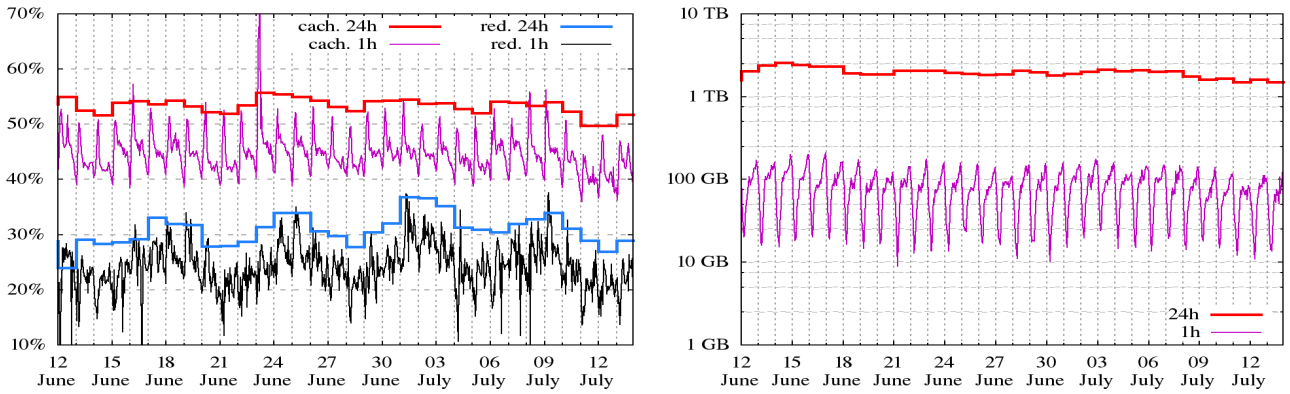

Figure 1: Cacheability, traffic reduction; (a) virtual cache size over 1h and 24h during a month from June July 2015.

cacheable content, assuming perfect "oracle" replacement);
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)*;
Average throughput and 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)*;
Share of HTTPS connections *(percentage of HTTPS connections and traffic over the total of web traffic)*.

3. HADOOP FRAMEWORK

Since the amount of data generated by HACkSAw can be huge (200GB per day for an average 10Gbps of traffic), the amount of time and memory needed to compute it on any single system is prohibitive. Hence our decision to use a Hadoop cluster. In average, on a daily basis, we use 398 maps and 60 reduce tasks which read and write 93GB and 0.8GB of data respectively. The hadoop framework employs 3.5 hours of CPU time and requires 505GB of memory. The real time processing takes only 2.4 minutes which constitutes a speedup factor of 87.5 times. The current setup involves an automatic script on the probe that compresses and sends the hourly logs towards the hadoop cluster, where they are uncompressed and injected in the HDFS (Hadoop Distributed File System). At regular intervals, a script on the hadoop cluster performs all the calculations for cacheability, traffic reduction and virtual cache size; the results are then exported to a web interface.

One important process on the data is the reaggregation of chunked objects, in order to avoid miscomputing their real hit ratio and size. Due to the complexity of the task, including the chunk reaggregation, the process uses two map-reduce steps: (i) the first step takes in the raw files produced by the probe; the map phase filters out the unused columns from the rows and the reduce phase aggregates all the records by object ID and time slot, and it merges all the object slices from range requests, using the `Content-Range` data, and finally it computes intermediate statistics. (ii) the second job takes in the intermediate results obtained in the first job; little processing is done in the map stage, while the reduce phase aggregates all records by time slot and calculates the final statistics.

4. DISCUSSION

We have reported a month of on-line statistics on a hourly and daily basis in Fig.1 that allows to draw some conclusions. Today, one third of the traffic is encrypted and politics will tell if this number will increase or not. We know, however, that a significant portion of such traffic comes from Google caches which gives a rough estimation of the hit ratio in such equipment (no statistics reported here though). Two third of the traffic in non encrypted and fine grained statistics are reported here. Almost half of the requests are cacheable and about one third of the traffic could be reduced. On a daily basis 1.2TB is roughly the size of an ideal memory installed at the vantage point to cache such data. However hourly statistics reveal that no more than 200GB would suffice. The gap between these two numbers proves that time locality is a very strong component of Internet traffic. These statistics suggest that in-network storage could be implemented in router memories and serve users' requests at wire speed. An ICN architecture would help optimizing traffic engineering by also using in-network caching in the access without having to give up content encryption, if required.

Acknowledgments

This research work has been partially funded by the Technological Research Institute SystemX, within the project on Network Architectures ARE.

5. REFERENCES

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

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

[3] C. Imbrenda, L. Muscariello, and D. Rossi. Analyzing Cacheable Traffic in ISP Access Networks for Micro CDN Applications via Content-centric Networking. In *Proc. ACM ICN*, 2014.

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

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

[6] 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.

[7] 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.