Analyzing Cacheable Traffic in ISP Access Networks for Micro CDN Applications via Content-Centric Networking

Claudio Imbrenda
Orange Labs
38-40, rue du general Leclerc
92130 Issy Les Moulineaux, France
claudio.imbrenda@orange.com

Luca Muscariello
Orange Labs
38-40, rue du general Leclerc
92130 Issy Les Moulineaux, France
luca.muscariello@orange.com

Dario Rossi
Telecom ParisTech
23, Avenue d'Italie
75013 Paris 13, France
dario.rossi@enst.fr

ABSTRACT

Web content coming from outside the ISP is today skyrocketing, causing significant additional infrastructure costs to network operators. The reduced marginal revenues left to ISPs, whose business is almost entirely based on declining flat rate subscriptions, call for significant innovation within the network infrastructure, to support new service delivery.

In this paper, we suggest the use of micro CDNs in ISP access and back-haul networks to reduce redundant web traffic within the ISP infrastructure while improving user's QoS. With micro CDN we refer to a content delivery system composed of (i) a high speed caching substrate, (ii) a content based routing protocol and (iii) a set of data transfer mechanisms made available by content-centric networking.

The contribution of this paper is twofold. First, we extensively analyze more than one month of web traffic via continuous monitoring between the access and back-haul network of Orange in France. Second, we characterize key properties of monitored traffic, such as content popularity and request cacheability, to infer potential traffic reduction enabled by the introduction of micro CDNs. Based on these findings, we then perform micro CDN dimensioning in terms of memory requirements and provide guidelines on design choices.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*

Keywords

Information-Centric Networking; Network Traffic Measurements; Web caching; Content-Delivery Networks; Telco CDN

General Terms

Performance; Measurement

1. INTRODUCTION

In recent years, we have assisted to significant deployments of web-based content delivery solutions within ISP networks, with the aim to deliver a range of services directly managed by the ISP itself. Some examples are video on demand (VoD) and subscriber VoD (SVoD). Broadcast TV, traditionally delivered across IP multicast channels, has also been made available through web-based portals supported by various kinds of CDN technologies. The spread of web-based content delivery solutions has multiple root causes, e.g. enhanced flexibility and ease of access from a vast set of user terminals, mostly mobile.

In particular, today SVoD consumers expect to the able to access video from any device inside and outside the home (TV, HDTV, smart-phones, tablets, media players), and also ask for high levels of QoS. Traditional IP multicast fails to satisfy such a large set of requirements, leading ISPs to build their own video services on top of CDN systems. Further, CDNs usage has been fostered by the skyrocketing growth of Internet traffic volumes, mainly driven by video services. In fact, traffic volumes are no more carrying preeminently P2P file sharing applications, but video-based services with significant QoS requirements and often based on a third-party (e.g. Netflix) monthly subscription.

Today model, relying on third-parties to provide web services, whose quality depends on someone else network infrastructure (one or multiple ISPs), present various business weaknesses. Indeed, applications like file sharing or VoIP (e.g. Skype) respectively generate elastic flows with soft QoS constraints or negligible traffic, not creating additional network costs.

For video-centric services, the relation between investments and revenues tends to be unbalanced when third-party content providers do not recognize any additional revenue to the ISPs which, instead, shoulder investment and operational costs to deliver additional traffic shares at no incremental revenue. The end customer is not eager to pay more to solve the dispute among the two parties if this requires to loose Internet access flat rate. The dispute involving Netflix and Comcast, Verizon and AT&T in the USA is one good example displaying the difficulties and frictions in this market [26]. In terms of network resource utilization, IP multicast, nowadays considered as obsolete, offers a valuable advantage over web based systems for TV broadcasting. In fact, IP multicast operates within the network while web based systems are today installed in the PoP (point of presence) and assume to reach the end user via a fully transparent high capacity network that ISPs have low business incentive to guarantee.

In this paper, we propose a novel approach for ISP content delivery, referred to as *micro CDN* (μCDN), combining the service flexibility of web based CDNs with efficient redundant traffic reduction of IP multicast. A μCDN makes use of small storage that can work at high speed as can take advantage of fast memories. A few technologies suitable to achieve this goal already exist, i.e. JetStream [24], AltoBridge [5] and also Open Connect [27]. However, they suffer from the following limitations: (i) they provide no standardized architecture, (ii) do not interoperate and (iii) do not support all kinds of web content.

We believe that a promising candidate to satisfy the aforementioned goals is the content-centric networking (CCN) architecture (a.k.a. NDN) [21]. Indeed, by embedding a service agnostic content caching into the network architecture, CCN may provide a common underlying network substrate for the deployment of next generation CDN systems.

The remainder of the paper is organized as follows. Sec.2 surveys related work. In Sec.3, we introduce the network setup, the dataset and the monitoring tool, HACkSAw, developed to analyze web data traffic. In Sec.4, we provide an extensive analysis of the dataset and a thorough statistical characterization of the key performance indicators required to guide μCDN system design. In Sec.5, we gather a number of trace driven simulations of different caching systems, while Sec.6 presents the general network design, including feasibility considerations tailored to CCN. Finally, Sec.7 concludes the paper.

2. RELATED WORK

In the last few years, a significant body of work has tried to measure the relation between the amount of resources and efforts (pain) to achieve the gains promised by CCN[18, 14]. Most of these works are based on network simulations in some very specific scenarios, none of them taking into account the ISP infrastructure, but focusing on content placement and caching performance no deeper than the PoP.

Other recent works, based on network experimentation, have shown [8] the significant gains of CCN in some relatively downsized network settings. The drawbacks of computer simulation and network experimentation are that, while being valuable, they do not allow to generalize the results for realistic workloads, difficult to model and to synthesize in general scenarios. Analytical models of such systems [7, 9] allow to quickly evaluate the relation between QoS, network capacity and users' demand; however, current models either fail to provide reliable performance predictions or must be tuned, a posteriori, using a data sample [28],[34].

A common assumption to the evaluation of caching systems performance is to assume that content requests are generated under the IRM (Independent Reference Model), with a request distribution following a Zipf law. Considerable measurement efforts have been devoted to provide input to this workload in static settings, while very few consider the catalog dynamics over time. Characterization of video catalogs has especially attracted significant attention, from YouTube crawling [19], to measurement at a campus [38] or ISP [28, 34], just to cite a few. Focusing on YouTube traffic, [28], [34] show that IRM assumption may yield to a significant underestimation of the achievable caching gains. However, the main focus of [34] is to propose a fine grained characterization of the data, rather than assessing the impact on the expected CCN gain (even a lower bound), as we do in this work.

In this paper, we propose a different approach based on network measurements: we install a live web probe on some links between the access and back-haul network serving fiber customers in the Orange network in France. Then, we evaluate the amount of redundant traffic flowing through the back-haul and the required memory to install within customer premise equipments (CPEs) and edge router line cards.

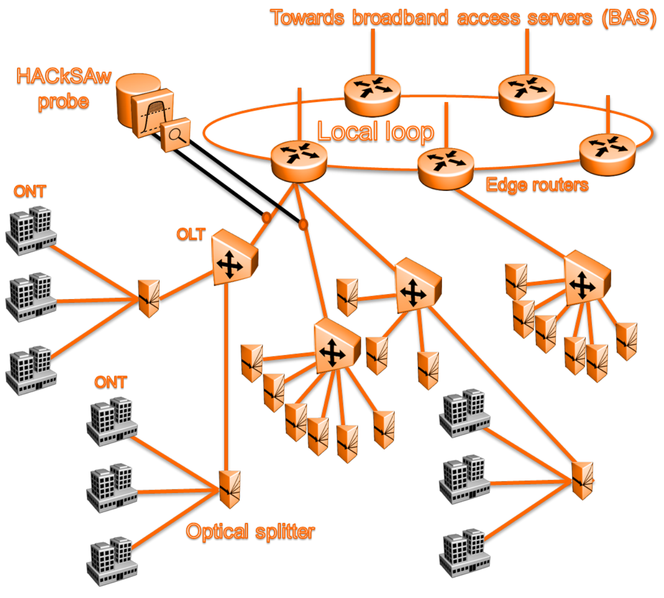


Figure 1: ISP network fiber access and back-haul.

Other work have measured cache performance for video applications [1], [2] but none of the previous work have considered also web content size which has, however, a significant impact on the amount of required storage to install in the network. Web content size is often neglected in previous work as which are based on the analysis on HTTP requests only, neglecting all HTTP replies. The most notable exceptions are [32] and [36] which use a measurement methodology similar to ours and analyze either HTTP request and replies.

3. WEB TRAFFIC MONITORING

This section introduces our traffic monitoring infrastructure, deployed within Orange France ISP network, the dataset under study and HACkSAw, our monitoring tool.

Our methodology is based on deep packet inspection (DPI) of the HTTP protocol and does not apply to HTTPs, where all bytes transferred across a TCP connection are encrypted. In our data set we observed 15% of all traffic on HTTPs and this statistic is expected to grow in the future as HTTP 2.0 specifies encryption by default. Considering the highly predominant usage of HTTP over HTTPs in our dataset the results presented in this paper are valuable to draw significant conclusion about cache performance at the network edge.

3.1 The monitoring infrastructure

Along the path between a PoP and the end customer, the monitoring probe can be placed at different network segments, ranging from the PoP or the broadband access server (BAS), where one observes the largest user fan-out, to the access network, where traffic is captured at a finer granularity. Our initial choice has been to perform monitoring of web traffic over the network segment between access and back-haul for fiber to the home users served by a GPON (Gigabit Passive Optical Network).

The promising findings over such dataset captured between the access and the back-haul have convinced us to permanently install the probe therein. As shown in Fig.1, each GPON consists of an OLT (optical line terminal), installed within a CO (central office) and accommodating, for instance, 16 line cards in a high density non blocking chassis. Line cards usually have a line rate of 1Gbps to 10Gbps and each one typically serves up to 64 users connected to a single optical line by a X64 optical splitter terminated by an ONT (optical network terminal) in the home. Each OLT has a back-haul link towards the local loop consisting of a given number of IP edge routers. Our probe (described in Sec.3.3) is installed at two such back-haul links in Paris, arriving at the same edge router as depicted in Fig.1.

3.2 The dataset

The probe captures every packet flowing on the wire and makes L3, L4 and L7 processing to collect a wide set of statistics at the different layers. In the paper we focus on L7, namely HTTP traffic, to obtain the relevant part of the paper's dataset: all HTTP requests and corresponding replies.

	Total	Daily Average
Distinct users	1478	1218
HTTP requests	369 238 000	8 586 000
HTTP data volume	37TB	881GB
Distinct objects	174 283 000	4 956 000

Table 1: Dataset summary: April 18 to May 30 2014.

Each detected HTTP request/reply transaction is recorded with a number of fields and associated statistics; namely time-stamp, user ID, object ID, Content-Length[15], and actual transaction length over the wire. We collect 42 days of data, from midnight April 18 to midnight May 30, 2014. Tab.1 reports a summary of such dataset. Details about the methodology used to process captured traffic and to obtain relevant statistics are provided hereafter.

Ref	Duration	Clients	Requests	Distinct objects	HTTP traffic
our dataset	42 days	1500	369M	174M	37TB
[2]	14 days	20k	–	–	40TB
[33]	14 days	20k	–	–	42TB
[16]	1 day	200k	48M	7M	0.7TB
[36]	8 days	1.8M	7.7G	–	256TB
[32]	1 day		42M	–	12TB
[14]	1 day	–	6M	–	–

Table 2: Overview of the dataset in this and related work.

Dataset of work inherent to caching or workload characterization, can either be request logs from servers, [37, 4, 14], gathered via active crawling techniques of popular portals [19, 11] or via passive measurement methodology [32, 36] as we do in this work. A comparison of some basic information about the dataset considered in this and related work is given in Tab.3.2, where it can be seen that our dataset has a significant span in terms of time, requests and distinct objects seen.

3.3 HACkSAw: the monitoring tool

We describe here HACkSAw, the continuous monitoring tool we have developed to collect our dataset and to process it in order to derive the statistics presented in next sections.

We recall that the measurement probe is installed at back-haul links with line cards ranging from 1Gbps to 10Gbps. Packets on the wire must be, then, captured and timestamped accurately at any of such mentioned line rates to prevent malfunctioning of the processing of the different protocol layers: IP, TCP/UDP, HTTP. The probe also requires to be installed in a network location guaranteeing symmetric routing, to carefully capture both directions of a TCP connection and requests and replies of an HTTP transaction (possibly pipelined in a single TCP connection, i.e. HTTP/1.1).

Tool	Detected Requests	CPU [sec]	Memory [GB]	0-length replies
Tstat2.4	2 531 210	445	0.3	1 128 109
Tstat2.3	1 348 642	345	0.4	N.A.
bro	2 559 056	8033	4.2	424 355
HACkSAw	2 426 391	368	5.8	328 465

Table 3: Performance of bro, Tstat and HACkSAw.

Recent work on web caching within the radio mobile backhaul in New York metropolitan area [32] and in South Korea [36] have used proprietary tools satisfying the mentioned requirements. Unfortunately, none of these tools is publicly available.

Conversely, popular open source tools like *bro* [29] and *Tstat* [25] do not satisfy all necessary requirements. Indeed, *bro*, conceived as an intrusion detection system, is not suitable for high speed monitoring because it applies regular expressions on each packet to detect signatures of known threats, and therefore results to be very slow. *Tstat*, instead, is faster and accurate in analyzing TCP connections, but inaccurate in analyzing HTTP transactions. Consequently, both tools turn out to be not satisfactory for our needs.

For this reason, the analysis presented in this paper is based on a novel tool, called HACkSAw and that we developed to accurately and continuously monitor web traffic in a modern operational ISP network, at any line rate, for any workload.

A comparison of the performance of the different tools is reported in Tab.3, using the same one hour trace as benchmark. We experiment with two version of Tstat, since its last version (released May 6th, 2014) addresses some (though not all) shortcomings related to HTTP traffic analysis. *bro* detects about twice as much HTTP requests with respect to *Tstat* (v2.3), using 10 times more memory and running over 20 times slower, whereas HACkSAw manages to be almost as accurate as *bro*, and almost as fast as *Tstat*. *Tstat* (v2.4) catches most of the transactions, though it still fails to match the requests with the reply, and is hence unusable to reports the size of the object (at least when the Content-Length header is not within the first IP packet of the reply, which happens more than 30% of the cases in our benchmark). As such, if *Tstat* (v2.4) addresses shortcomings in terms of HTTP transactions recall, it still fails to provide a reliable measure of object size.

HACkSAw implements L3 packet collection (and reordering, when needed) and L4 flow matching and reconstruction, so that a TCP connection is properly tracked and analyzed. Scalability is achieved via efficient non-blocking multithreaded design, available in none of previous tools, enabling deployment of the probe in any access or back-haul link. At flow termination (after a TCP half close or an inactivity timeout), the payload stream is handed to one of the several consumer threads that are in charge of HTTP transactions analysis.

Accuracy is, thus, achieved via full payload analysis and full stream reconstructions (so that memory usage is close to that of *bro*). In the network setup previously described, HACkSAw runs on an IBM server with 2 quad-core Intel Xeon CPU E5-2643 at 3.30GHz with 48GB of RAM each, for a total of 96GB of RAM memory. The server is equipped with a Endace DAG7.5G4 card, allowing to capture packets from 4 Gigabit links simultaneously, allowing us to monitor 2 full-duplex Gigabit links. HACkSAw logs several information for each HTTP transaction namely:

- **request timestamp**, accurately given by the Endace DAG card;
- **user identifier**, computed as the obfuscated MAC address of their home router;
- **object identifier**, calculated as a hash of the full URL concatenated with the ETAG[15], when present;
- **reported object size**, from the HTTP Content-Length header, when present or computed from the chunk sub-headers in case of chunked transfer-encoding;
- **actual object size**, computed as the actual amount of unique (i.e., excluding TCP retransmissions) bytes on the wire, that may be lower than the reported object size in case of aborted transfer.

4. WEB TRAFFIC CHARACTERIZATION

Traffic characterization is an essential prerequisite of traffic engineering: network dimensioning and upgrading lie upon the knowledge of the relation between three entities: traffic demand, network capacity and quality of service. What makes traffic characterization a difficult task is the stochastic nature of Internet traffic, complex to synthesize via simple models.

In literature, a wide range of models exist, varying model abstraction and related complexity according to a more microscopic or macroscopic analysis of network dynamics. In this paper, we avoid a detailed representation of a network of caches, which turns out to be analytically intractable even for a simple tandem cache and simple workloads [17]. We rather prefer a simple characterization of web traffic, based on key system properties and applicable to general in-network caching systems. Such model abstraction, assuming an independent reference model (IRM), might be leveraged for the dimensioning of a micro CDN. The key factors impacting the performance of an in-network caching system and that our model takes into account are:

- the timescale at which content popularity may be approximated by an independent reference model (IRM) (Sec.4.1);
- the content popularity at the given timescale (Sec.4.2).

From their characterization, we later infer in Sec.5.1 the minimum useful amount of memory to embed in the home network and in edge routers in the Micro CDN architecture.

4.1 Timescale analysis

First, we define some statistics that will be used throughout the section: catalog, cacheability, traffic reduction. The *catalog* is the set of distinct objects requested in a given time interval. As introduced in [2], the *cacheability* is a statistic indicating the fraction of requests for objects requested more than once in a given time interval. The first request of an object is not considered cacheable, whereas all its subsequent requests in the same time interval are. The resulting definition of cacheability is

$$\frac{N_r - N_o}{N_r} = 1 - \frac{N_o}{N_r} \qquad (1)$$

where N_r is the number of requests observed in the given time interval, and N_o is the number of unique distinct objects observed (the cardinality of the catalog).

We also define the *traffic reduction*, a statistic measuring the maximum amount of traffic that can be potentially saved across a link over a given time interval as a consequence of content cacheability. Traffic reduction is defined as

$$\frac{R - R_u}{R} = 1 - \frac{R_u}{R} \qquad (2)$$

where R is the total traffic, and R_u is the uncacheable traffic, both measured in bytes. Before presenting observations from the network probe, we use a simple explanatory model to show that measuring content popularity at a timescale (as an example, over a large time window), where the IRM assumption does not hold, may lead to wrong predictions in terms of memory requirements.

We divide the time axis in windows of size $T > 0$, $W_i = [iT, iT + T)$, and assume that, in each time window W_i, objects in content catalog A_i are requested following a Poisson process of rate λ, with $A_i \cap A_j = \emptyset$ for all $i, j : i \neq j$. The average object size is σ bytes. A_i is Zipf distributed with parameters α, N, i.e. a content item of rank k is requested with probability $q_k = ck^{-\alpha}$, $k \in \{1, \ldots, N\}$, $|A_i| = N$.

By using the modeling framework developed in [7] for an LRU cache of size x in bytes, we know that if $T >> x^\alpha g$, with $1/g = \lambda c \sigma^\alpha \Gamma(1 - 1/\alpha)^\alpha$ the cache miss probability for an object of rank k tends to $\exp\{-\lambda q_k g x^\alpha\}$.

However, if one estimates the content popularity as the time average across m contiguous time windows, the estimated miss probability would be $\exp\{-\lambda q_k g (x/m)^\alpha\}$. Indeed, the right cache performance measured across m contiguous time windows of size T is still $\exp\{-\lambda q_k g x^\alpha\}$ resulting in an overestimation factor m of the required memory, for the same miss ratio.

In this section we estimate the timescale over which the IRM model can be used to estimate cache performance without using complex measurement-based models, e.g. [28],[34].

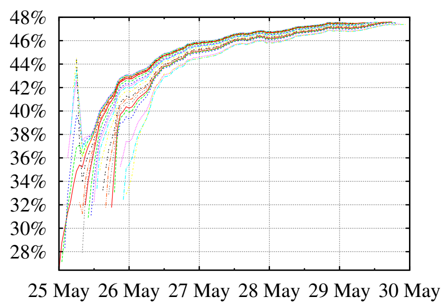
Observation 4.1. *In order to exploit a IRM cache network model for system dimensioning, one needs to estimate the smallest timescale, referred to as "cutoff" timescale at which the IRM assumption holds. As a consequence, above the cutoff timescale, every new content request gives a negligible contribution to catalog inference.*

In Fig.2(a),(b) we plot cacheability and traffic reduction as computed over our dataset at different time windows: from one hour to an entire contiguous week at incremental steps of one hour. The statistics are also computed starting at different time instants, delayed by one hour each. We observe that the two statistics have a cutoff scale above which they reach a plateau. In Fig.2(c), we report the time required for the cacheability to attain percentiles (namely, the 75%, 90%, 95% and 99%) of the long term value: it can be seen that 90% of the traffic reduction are attained in less than 24 hours, irrespectively of the start time.

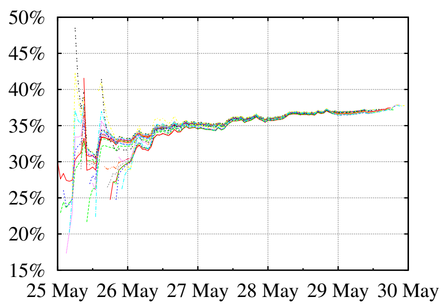
Observation 4.2. *The cutoff scale is hard to be measured as it changes on a daily basis as a function of many factors that cannot be rigorously quantified. However, we observe that for practical purposes aggregate web traffic would benefit for caching no more than a daily content catalog.*

In Fig.2(a),(b) we also observe that the cacheability stabilizes at about 47% while traffic reduction amounts to almost 37%. These values provide a first rough estimation of the opportunities to cache data currently available within the ISP network at relatively low user fan-out. While statistics just presented provide insights on the potential gains achievable by caching a daily content catalog, we now investigate temporal evolution of the catalog.

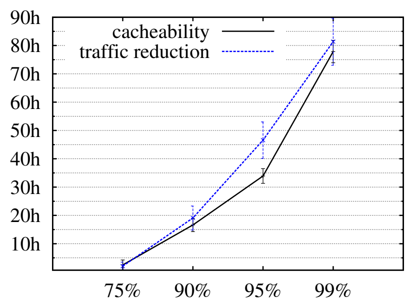
To this aim, we introduce a measure of auto similarity based on the *Jaccard coefficient*, that indicates the proportion of objects in



(a) Cacheability



(b) Traffic reduction



(c) Hours needed to reach some percentiles of the cacheability and traffic reduction plateaus, with standard deviations.

Figure 2: Cumulative cacheability and traffic reduction, starting from different hours and for various timespans.

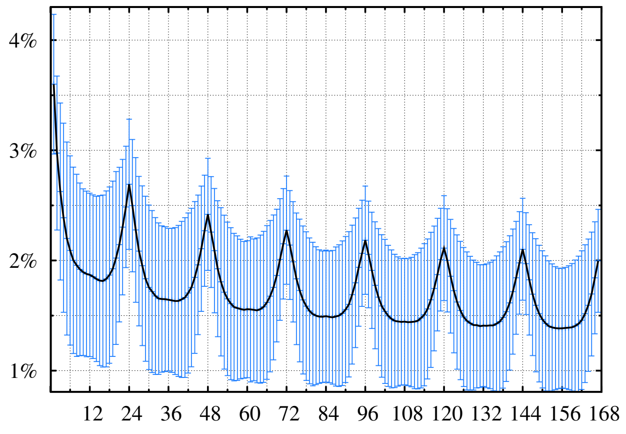


Figure 3: Jaccard auto correlation function $\mathcal{R}_{\Delta T}(k)$.

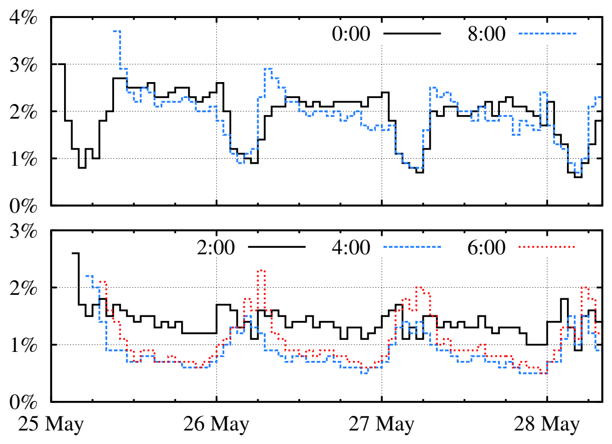


Figure 4: Jaccard coefficient, $J(C_{k_0\Delta T}, C_{(k_0+k)\Delta T})$, $k_0 = 0, 2, 4, 6, 8$.

common between two given sets: $J(A,B) = \frac{|A\cap B|}{|A\cup B|}$ (If $A = B = \emptyset$ then $J(A,B) \triangleq 1$. Clearly, $0 \leq J(A,B) \leq 1$. We then define the Jaccard auto correlation function as

$$\mathcal{R}_{\Delta T}(k) = \frac{1}{n} \sum_{i,j:|i-j|=k}^{n} J(C_{i\Delta T}, C_{j\Delta T}) \qquad (3)$$

being $C_{i\Delta T}$ the content catalog measured over the time window $i\Delta T$. Fig.3 shows $\mathcal{R}_{\Delta T}(k)$ for $k = \{0, \dots, 168\}$, ΔT = 1hour, during one working week in May 2014 (showing standard deviation as error bars). An interesting conclusion can be drawn.

OBSERVATION 4.3. *The catalog is weakly auto-correlated, as $\mathcal{R}_{\Delta T}(k)$ falls from 100% to less than 5% and it completely regenerates asymptotically, as $\mathcal{R}_{\Delta T}(k) \to 0$ when $k \to \infty$. A periodic component with period of about 24 hours is also present as a result of users' daily routine.*

Finally, we show that catalog show a night/day effect in Fig.4, that reports $J(C_{k_0\Delta T}, C_{(k_0+k)\Delta T})$, with ΔT = 1hour and $k_0 < k \leq 72$, for multiple $k_0 = \{0, 2, 4, 6, 8\}$. The figure shows that the catalog has different properties during off peak hours ($k_0 = \{0, 2, 4\}$) than peak hours ($k_0 = \{6, 8\}$). Off-peak hours are characterized by content items that unlikely appears again in the future, while on-peak content items show periodic components of 24 hours.

4.2 Content popularity estimation

Hereafter, we present a model of content popularity estimated over 24 hours. According to the observations reported above, the timescale of interest turns out to be approximately defined by removing the night period, i.e. the off peak phase. This may be a complex task as the off peak phase changes on a daily basis. Nevertheless, we observe that the off peak phase has statistically weak impact on the overall distribution as it carries samples in the tail of the popularity distribution at low rate, so that 24 hours can be used as timescale. We test the empirical popularity against various models and find the discrete Weibull to be the best fit with shape around 0.24 (long tailed). In Fig.5, we report the empirical popularity distribution with corresponding 95% confidence bands (see [20] for similar analysis). We also plot, in red, the model fit to the available sample with 95% confidence bands. By means of extensive tests on this model we assess accuracy over all 24 hours samples on our data set. It follows that:

- for the *tail* of the distribution, a simple discrete Weibull passes a χ^2 goodness of fit test [12], with p-values exceeding 5% significance level, while
- the good model for the *entire distribution* turns out to be trimodal with three components: a discrete Weibull for the *head* of the distribution, a Zipf for the *waist* and, a Weibull

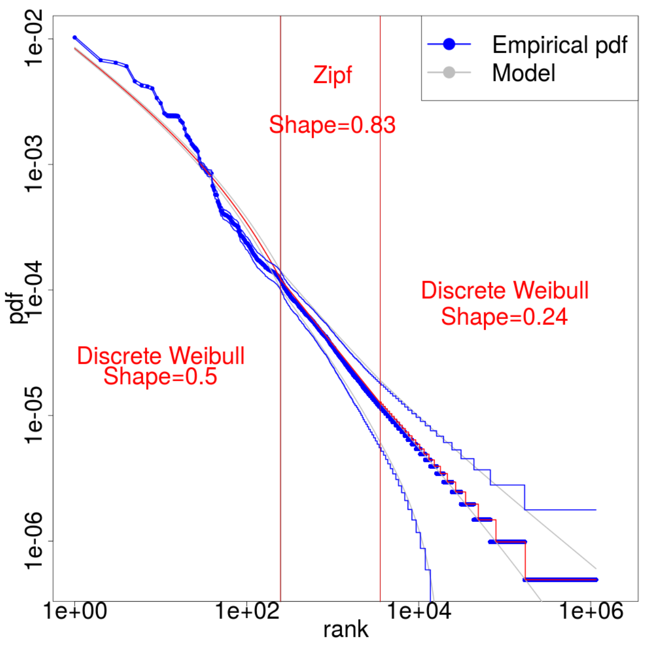


Figure 5: Empirical web content popularity and model fitting with corresponding confidence bands.

again for the *tail*, i.e. $f(k) =$

$$\begin{cases} \phi_1 \dfrac{\beta_1}{\lambda_1}\left(\dfrac{k}{\lambda_1}\right)^{\beta_1-1} e^{-(k/\lambda_1)^{\beta_1}} & k < k_1 \\ \dfrac{\phi_2}{k^{\alpha_2}} & k \in [k_1, k_2] \\ \phi_3 \dfrac{\beta_3}{\lambda_3}\left(\dfrac{k}{\lambda_3}\right)^{\beta_3-1} e^{-(k/\lambda_3)^{\beta_3}} & k > k_2 \end{cases}$$

with parameters $\lambda_1, \beta_1, \alpha_2, \lambda_3, \beta_3, \phi_1, \phi_2, \phi_3 \in \mathbb{R}^+$; $k_1, k_2 \in \mathbb{N}$. The parameters have been estimated by using standard maximum likelihood (ML) applied to the piecewise function $f(k)$. The set of parameters of each piece of $f(k)$ is estimated independently to the others in order to fix the shapes exponents $\beta_1, \alpha_2, \beta_3$ and the scale factors λ_1, λ_3. An ML estimator is not available for the entire distribution and we therefore use the method of moments to determine ϕ_1, ϕ_2, ϕ_3. The procedure can be iterated to obtain a better estimation by running ML first and MM afterwards. In our samples, after few iterations (e.g. four) the parameters stabilize to stationary values that we have reported for one day in Fig.5 where $\beta_1 = 0.5$, $\alpha_2 = 0.83$, $\beta_3 = 0.24$.

Interesting conclusions can be drawn from our popularity characterization. In literature, the majority of analytical models assume a Zipf distribution with shape $\alpha < 1$ for the entire distribution. Remark that, if the same Zipf law characterizing the waist is prolonged all over the support of the distribution, a finite support, corresponding to a content catalog estimation, must be imposed. Indeed, the miss probability of a cache (LRU or LFU) under Zipf requests with $\alpha < 1$ is a function of the ratio between the cache and catalog sizes, e.g. for LFU it is $1 - (x/N)^{1-\alpha}$ ([22]).

Cache performance would then depend on the ratio between cache and content catalog size, while it is a function of cache size only under the more precise Weibull tail fit that we made. More, the cardinality of the catalog, N, is estimated with unbounded confidence intervals (see Fig.5), whereas all Weibull's parameters can be estimated with arbitrary low error, by increasing the size of the sample. As a consequence, an overestimation of the catalog size by a given factor under the all-Zipf model would lead to memory over-sizing of the same factor for a given target miss ratio.

Conversely, the miss ratio under Weibull requests ([23]), e.g. of an LRU cache, can be estimated with arbitrary precision by increasing the size of the sample to estimate the popularity law. Hence, we derive that

OBSERVATION 4.4. *Accurate content popularity characterization is fundamental to drive a correct cache system dimensioning. Approximate models based on (i) all-Zipf assumption, (ii) possibly fit over long time scales, coupled to (iii) IRM model assumptions, may lead to excessively conservative results, and ultimately to misguided system design choices.*

5. TRACE DRIVEN SIMULATIONS

In the following, we present some realistic simulations driven by our dataset. The goal is to evaluate the amount of memory required within the back-haul to absorb cacheable traffic and to assess the accuracy of the model introduced in previous section for the dimensioning of a micro CDN system.

5.1 Single LRU cache at edge router

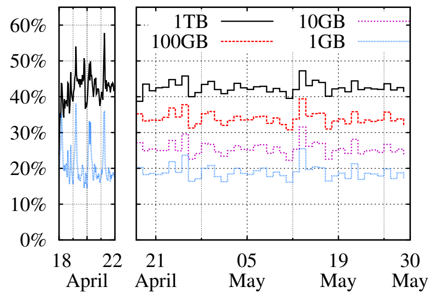
The first set of simulations is based on an LRU cache installed at the probe and driven by real users' requests. We simulate a LRU cache with different sizes and measure (i) the average hit ratio, (ii) the potential traffic savings over two timescales (1 hour and 1 day). The LRU cache simulates a transparent cache: a web object is stored in chunks, so that in case of a cache miss, only the missing chunk is requested. A following request for a bigger part of an object partially present in cache generates another miss, but only for the missing part of the object.

We consider sizes of 1GB, 10GB, 100GB and 1TB, and never explicitly flush cache content. Performance metrics are reported in Fig.6(a),(d) and compared with two additional systems obtained by shuffling all the requests on a hourly and daily basis, reported in Fig.6(b),(e) and Fig.6(c),(f) respectively. Request shuffling is useful to remove time correlation, which has huge impact on cache performance as already discussed in Sec.4 (i.e., shuffling produces a workload closer to that of IRM model).

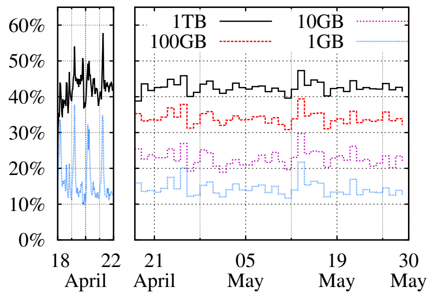
From the simulations we see that if the cache is big enough (1TB), time correlations has not impact on performance. For medium or small caches instead, time correlation affects significantly performance as hit ratios and saved traffic, twice as much in presence of temporal locality of requests.

We now assess memory requirements, showing statistics that are gathered in an online fashion by our probe. We introduce an additional metric, the *virtual cache size*, defined as the sum of the total observed sizes of the cacheable objects. Such additional metric, not accounted for by models, allows to quantify the effect of incomplete downloads (e.g. as an effect of user impatience). Since many objects, especially the largest ones, are not always entirely requested, we define the size of an object as the largest size observed on-wire for the given object.

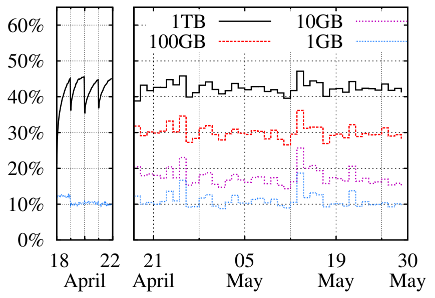
Fig.7 plots cacheability (top row), traffic reduction (middle row) and virtual cache size (bottom row), over more than one month in 2014, over hourly and daily timescales. Different system configurations are arranged by columns: namely, we either assume that a single cache is installed within at the OLT level (left column),



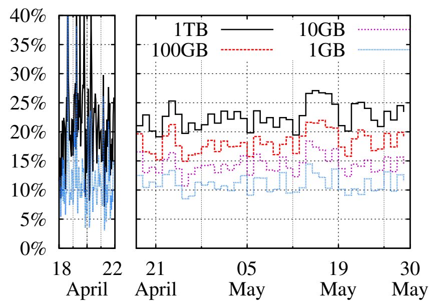
(a) Hit ratio.



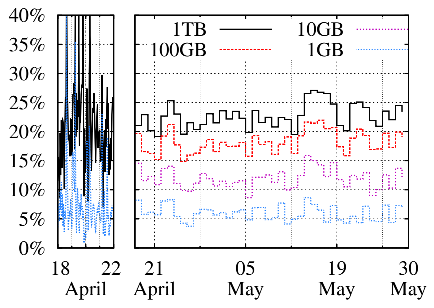
(b) Hit ratio, shuffling hourly.



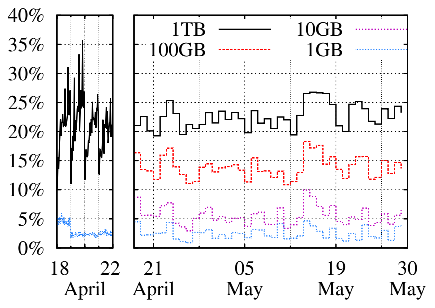
(c) Hit ratio, shuffling daily.



(d) Saved traffic.



(e) Saved traffic, shuffling hourly.



(f) Saved traffic, shuffling daily.

Figure 6: LRU cache simulations: behavior with one hour averages (left side) and the general view of the whole dataset with daily averages (right side).

or that caches are deployed only within ONT in the home of each user (middle column), or that caches are deployed at both ONT and OLT levels (right column). The OLT-caching scenario in the left column of Fig.7(a),(d),(g) is striking: with a little more than 100GB of memory, 35% of average traffic can be saved for a user fan-out equal to 2048. In the ONT-caching scenario, only duplicated requests coming from the same users are filtered by the cache: in this case, an average memory size of about 100MB (per user, totalizing 200GB given the user fan-out) reduces user traffic by 25%, corresponding to a same level of load reduction in the GPON access. Finally, the last column of figures, Fig.7(c),(f),(i), shows that employing caches at both OLT and ONT level, the ISP network would benefit from 25% load reduction in the GPON and 35% on back-haul links, while improving the latency for all users.

6. μCDN DESIGN CHOICES

The above results confirms the potential caching benefits arising from the deployment of limited additional memories. In this section, we comment on implications of the above findings in terms of the architectural design that is best suited to actually leverage this caching potential.

6.1 CCN as uniform CDN substrate

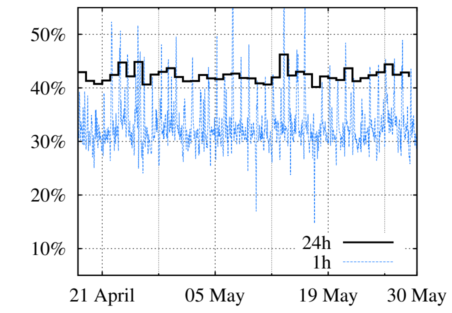
Transparent caching is today a very useful technique to reduce redundant traffic while, at the same time, providing higher levels of QoS to the users. All the different existing solutions cited [3, 30, 5, 27], are based on a common set of principles. They leverage DPI (Deep Packet Inspection) to identify and reconstruct content on-line. Once the content is identified, it is split into identifiable data chunks, corresponding to proper networking datagrams, that are cached and routed across the cache infrastructure. Most of these technologies are proprietary and their design and implementation details may only be obtained partially through non disclosure agreements. The challenge of integrating and inter-operate various CDN technologies into one inter-operable infrastructure has led to the creation of the standardization working CDNi at IETF [35]). However, such standardization activity struggles with attracting the main CDN players, so raising concerns about the success of the integration process. By embedding a service agnostic content caching into the network architecture, Information Centric approaches appear to be more promising candidates to provide a common underlying network substrate for the deployment of next generation CDN systems.

6.2 CCN communication model

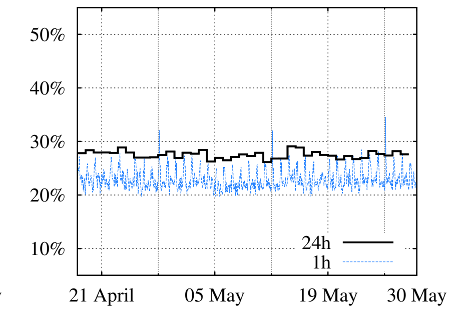
Going one step further in terms of architectural details, the key characteristics of the communication model of today CDNs are put in perspective with evolutionary aspects of CCN in [10]. The main CDN components are (i) DNS and content based routing, (ii) HTTP and chunk-based data transfer (iii) in-network caching. Note that a commercial CDN also includes a large set of analytics to log service quality and monitor the CDN performance.

If we replace the data plane of a CDN system with CCN, the main innovation introduced is to overcome the current misuse of HTTP as a content-centric transport protocol and to integrate a chunk-based connectionless receiver driven transport coping with in-network caching.

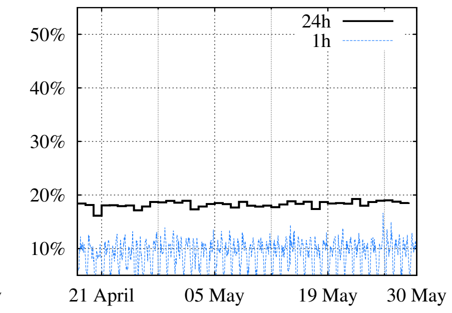
Let us describe CCN communication model more in detail ([21]). The founding principle of CCN is to enrich network layer primitives with content awareness so that routing, forwarding, caching and data transfer functions are performed on content names rather than on location identifiers (IP addresses). Content items are divided into a sequence of chunks uniquely identified by a name and a permanent copy is stored in one or more repositories. The nam-



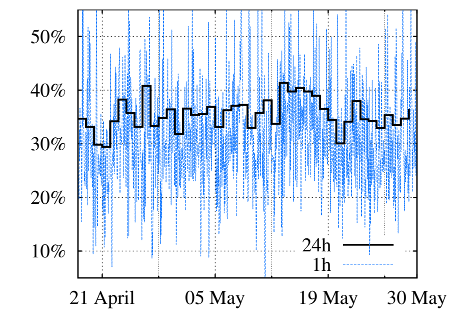
(a) Cacheability, cache at OLT only.



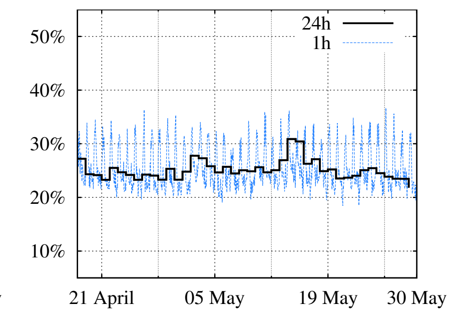
(b) Average Cacheability, cache at ONT only.



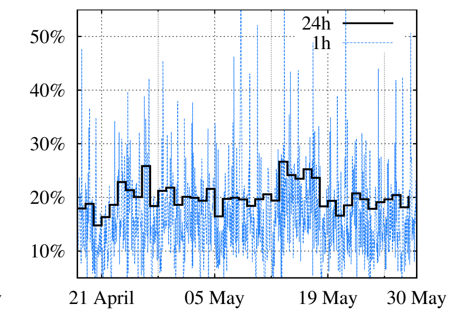
(c) Cacheability at OLT, with ONT cache present.



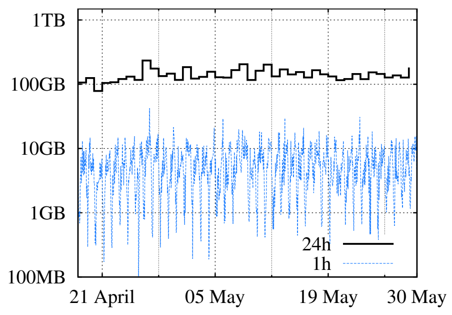
(d) Traffic reduction, cache at OLT only.



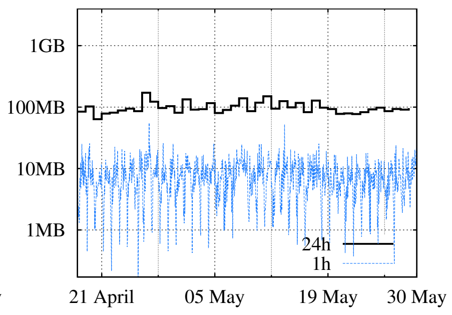
(e) Average traffic reduction, cache at ONT only.



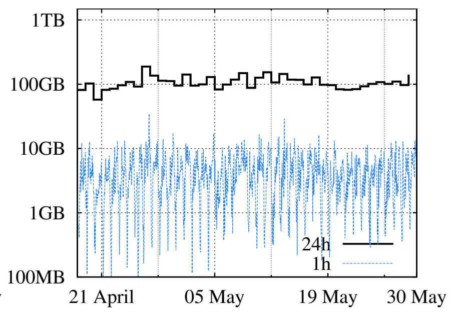
(f) Traffic reduction at OLT, with ONT cache present.



(g) Virtual cache size, cache at OLT only.



(h) Average virtual cache size, cache at ONT only.



(i) Virtual cache size at OLT, with ONT cache present.

Figure 7: Time evolution of hourly and daily statistics; by rows cacheability, traffic reduction and virtual cache size are reported respectively in the three cases: (i) cache at OLT only, (ii) cache at ONT only and (iii) cache at OLT and ONT.

ing convention is not specified, only a hierarchical structure like the one already adopted by HTTP is required for entries aggregation in name-based routing tables.

Naming data chunks allows the network to directly interpret and treat content according to its semantic with no need for DPI (Deep Packet Inspection) as for transparent caching. The content delivery process is then driven by three basic communication mechanisms:

- *Name based request routing*. CCN forwarding engine is based on a name-based routing table storing one or more potential next hops towards a set of content items and on a Pending Interest Table (PIT) keeping track of ongoing requests in order to route data packets back to the user along the reverse request path (breadcrumb routing).

- *Receiver-based connectionless transport*. Differently from current sender-based transport model, CCN data transfer is triggered and controlled by user requests (Interests) in a pull-based fashion. Rate and congestion control is performed at the end user by means of a connectionless, yet stateful transport protocol with the following characteristics: (i) no connection instantiation for the support for user/content mobility; (ii) support for retrieval from multiple sources, a priori unknown at the user (e.g. intermediate caches); (iii) support for multipath communication (to improve user performance and traffic load balancing).

-*In-network caching*. The content-awareness provided by names to network nodes enables a different use of routers' buffers, not only to absorb input/output rate unbalance but also for temporary in-network caching and processing (i.e. data transcoding) of in-transit Data packets. Even without additional storage capabilities in routers, the information access by name of CCN allows new uses of in-network wire speed storage. Indeed, network nodes temporarily store content items/chunks in order to serve future requests for the same content (reuse) and to recover from potential packet losses (repair). Upon reception of a chunk request, a CCN node first checks if the requested chunk is present in the local cache. If it is the case, the content is returned back to the user. Otherwise, the request is forwarded to the next hop by the ICN request routing.

CCN data plane can be easily implemented in hardware in CPEs (ONT or home gateways) and router line cards. The necessary processing of content requests and replies (that many proprietary products already do for transparent caching applications) results feasible at high speed and simplified w.r.t. transparent caching solution (as it does not involve complex DPI operations). For the adoption of CCN communication model, we believe that a required step may be the standardization of the main building blocks: packet formats, processing and API specifications.

6.3 Transparent web caching and encryption

As mentioned in Sec.3, the usage of encryption instead of plain text in the Internet is growing and is also considered a desirable feature to deploy increasingly as much as possible. HTTP 2.0 draft [6] currently under discussion in the HTTPbis IETF working group specifies encryption by default by using TLS 1.2 [13]. TLS provides communications security for client/server applications and encrypts everything included in the TCP byte stream. The encryption service provided by TLS is not compatible with proxy or transparent caching which is however a very important service successfully deployed to reduce bandwidth consumption in many network locations. Caching non encrypted web traffic in proxies or transparent appliances is today implemented and optimized by using HTTP almost as a transport layer. An HTTP datagram has also been proposed in [31] to effectively implement almost all the functionalities CCN provides, with the exception of data encryption and security in general. It is clear that encrypted web traffic, using TLS, can be cached only in the end points, i.e. the client web browser, or application, and in content provider appliances. Of course this latter end point can be distributed in a CDN which manages the encryption on behalf of the content provider. Therefore caching encrypted web traffic cannot be implemented as a transparent network primitive because it would always require the sender to delegate encryption to a third-party, e.g. a CDN. A minimum level of cooperation is required to guarantee inter-networking of communications primitives which are based on delegation. Datagram based packet-switched networks make use of delegation for data forwarding and routing but other services like name resolution, as provided by the DNS, do require delegation as well. We believe that in-network data caching is an additional transparent network primitive that cannot be implemented without guaranteeing the required level of security and interoperability that TLS cannot provide.

TLS tunnels can be bypassed in principle, at a certain extent, but not in a fully transparent way to the end users. A web proxy between the content provider and the user can terminate a TLS tunnel by issuing on the fly a fake certificate to the user and create a new TLS tunnel between the proxy and the user. In this way all traffic would not be encrypted anymore at the proxy. Caching would then be possible again. Such system is however a workaround and does not constitute a solution for many reasons: (i) the user is supposed to accept a certificate signed by a non trusted authority, (ii) the client adds the proxy as a trusted certification authority which can then issue trusted keys; (iii) the client accepts self certified certificates. This technique has the drawbacks to expose the user to higher level of vulnerability to external attackers. To the best of our knowledge, CCN is the only network architecture addressing data security by design instead of tunnel security, which is a feature that HTTP does not provide yet.

6.4 Technical feasibility

Previous sections have shown that significant benefits in terms of traffic reduction can be achieved at the cost of very limited additional memory. Indeed, a single ONT installed in a user's home would only need to host approximately 100MB of additional memory. Such memory, currently not available in the optical device, is available in the home gateway, that are equipped with enough CPU resources to easily manage 100MB of RAM at 1Gbps. Upstream to the OLT, 100GB memory in a router line card would be enough to provide the early shown gains and it seems feasible in current hardware. Hence, the deployment of a μCDN technology in the home would only require to implement the content-centric forwarding engine in the home gateway firmware – which again seems feasible due to the current development effort on several CCN/NDN prototypes.

7. DISCUSSION AND CONCLUSION

In this paper, we provide evidence of the potential gains associated to the deployment of micro CDN technologies in ISP networks. Our analysis is grounded on a large dataset collected via the on-line monitoring of links between the access and the back-haul network of Orange France. Leveraging one month and a half of continuous traffic monitoring, we are able to support our design by an accurate characterization of content dynamics.

The large data set we have used allowed fine grained statistical analysis of content popularity dynamics, whose value goes beyond the primal objective of this paper. Indeed, the analysis demonstrates the inadequacy of traditional models, like simplistic Zipfs workloads, and their failure for prediction and dimensioning. The gains are striking: with a negligible amount of memory of 100MB on CPEs, the load in the access network (GPON) can be reduced by 25%, while embedding a 100GB of dynamic memory in edge IP router line cards can also reduce back-haul links load of about 35%.

The significant potential gains we have measured in today traffic does no apply to encrypted web application (15% in our dataset) which cannot be transparently cached. Caching TLS encrypted traffic is however going to be a significant issue for ISPs as it would require some form of interaction with the content provider, or the CDN on its behalf. Transparent caching of encrypted traffic might also be achieved by TLS tunnels interception which implies some form of increased vulnerability at the user's client. We suggest instead to use CCN as the best fit technology to address all the drawbacks of currently available workarounds and providing caching as a transparent network primitive. We additionally assess technical feasibility in nowadays hardware, and argue that such a small amount of additional memory can be supported at high speed on current technologies. We finally identify steps for micro-CDN implementation, outlining arguments to support a CCN-based deployment as basic building block. Our analysis suggests that CCN-based solutions are close enough to be deployed in real ISP networks – its realization is part of our ongoing work.

Acknowledgments

This research work has been partially funded by the Technological Research Institute SystemX, within the project "Network Architectures" hosted at LINCS and has been partially funded by the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane"). We thank Giovanna Carofiglio for the fruitful discussions and the anonymous reviewers who helped improving the manuscript.

8. REFERENCES

[1] H. Abrahamsson and M. Nordmark. Program popularity and viewer behaviour in a large tv-on-demand system. In *ACM IMC*, 2012.

[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] Akamai Aura Lumen. Licensed suite of operator cdn solutions. `http://www.akamai.com/html/solutions/aura_lumen_cdn.html`.

[4] M. S. Allen, B. Y. Zhao, and R. Wolski. Deploying Video-on-Demand Services on Cable Networks. In *Proc. of the ICDCS*, Washington, DC, USA, 2007.

[5] Altobridge. Data-at-the-edge ®. `http://www.altobridge.com/data-at-the-edge%E2%84%A2/architecture/`.

[6] M. Belshe, R. Peon, and M. Thomson. Hypertext transfer protocol version 2, 2014.

[7] G. Carofiglio, M. Gallo, and L. Muscariello. Bandwidth and Storage Sharing Performance in Information Centric Networking. *Elsevier Science, Computer Networks Journal, Vol.57, Issue 17*, 2013.

[8] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang. Optimal Multipath Congestion Control and Request Forwarding in Information-Centric Networks. In *Proc. of IEEE ICNP*, 2013.

[9] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino. Modeling Data Transfer in Content-Centric Networking. In *Proc. of ITC23*, 2011.

[10] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, and M. Varvello. From Content Delivery Today to Information-Centric Networking. *Elsevier Science, Commputer Networks Journal*, 2013.

[11] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems. *IEEE/ACM Transactions on Networking*, 2009.

[12] R. B. D'Agostino and M. A. Stephens, editors. *Goodness-of-fit Techniques*. Marcel Dekker, Inc., New York, NY, USA, 1986.

[13] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176.

[14] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less Pain, Most of the Gain: Incrementally Deployable ICN. In *Proc. of ACM SIGCOMM*, 2013.

[15] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Rfc 2616, hypertext transfer protocol – http/1.1, 1999.

[16] A. Finamore, M. Mellia, Z. Gilani, K. Papagiannaki, V. Erramilli, and Y. Grunenberger. Is There a Case for Mobile Phone Content Pre-staging? In *Proc. of ACM CoNEXT*, 2013.

[17] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy. Performance Evaluation of the Random Replacement Policy for Networks of Caches. *Elsevier Science, Performance Evaluation Journal*, 2014.

[18] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric Networking: Seeing the Forest for the Trees. In *Proc. of ACM HotNets-X*, 2011.

[19] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube Traffic Characterization: a View From the Edge. In *Proc. of the ACM SIGCOMM IMC*, 2007.

[20] W. Gong, Y. Liu, V. Misra, and D. Towsley. On the Tails of Web File Size Distributions. In *Proc. of Allerton Conference on Communication, Control, and Computing*, 2001.

[21] V. Jacobson, D. Smetters, J. Thornton, and al. Networking Named Content. In *Proc. of ACM CoNEXT*, 2009.

[22] P. Jelenkovic, X. Kang, and A. Radovanovic. Near Optimality of the Discrete Persistent Access Caching Algorithm. In *Proc. of International Conference on Analysis of Algorithms (DMTCS)*, 2005.

[23] P. R. Jelenković. Asymptotic approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities. *The Annals of Applied Probability*, 9(2):430–464, 1999.

[24] Jet-Stream. Technology overview. `http://www.jet-stream.com/technology-overview/`.

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

[26] Netflix. The case against isp tolls, april 24, 2014. `http://blog.netflix.com/2014/04/the-case-against-isp-tolls.html`.

[27] Netflix. Open connect content delivery network. `https://www.netflix.com/openconnect`.

[28] F. Olmos, B. Kauffmann, A. Simonian, and Y. Carlinet. Catalog dynamics: Impact of content publishing and perishing on the performance of a lru cache. In *Proc. of ITC26*, 2014.

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

[30] PeerApp. Transparent caching in dsl operator networks. `http://www.peerapp.com/Solutions/dsl.aspx`.

[31] L. Popa, A. Ghodsi, and I. Stoica. HTTP as the narrow waist of the future Internet. In *ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets'X)*, 2010.

[32] 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.

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

[34] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review*, 43(5):5–12, 2013.

[35] R. van Brandenburg, O. van Deventer, F. L. Faucheur, and K. Leung. Models for HTTP-Adaptive-Streaming-Aware Content Distribution Network Interconnection (CDNI). RFC 6983 (Informational), July 2013.

[36] 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.

[37] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proc. of the ACM SIGOPS/EuroSys*, New York, NY, USA, 2006.

[38] M. Zink, K. Suh, Y. Gu, and J. Kurose. Characteristics of YouTube network traffic at a campus network - Measurements, models, and implications. *Comput. Netw.*, 53(4):501–514, Mar. 2009.