Revisiting Cacheability in Times of User Generated Content

Bernhard Ager, Fabian Schneider, Juhoon Kim, Anja Feldmann
{bernhard|fabian|jkim|anja}@net.t-labs.tu-berlin.de
TU Berlin, Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10589 Berlin

***Abstract*—Today's Internet traffic is dominated by users' demand for exchanging content. In particular, multi-media content, i. e., photos, music, and video, as well as software downloads and updates contribute substantially to today's Internet traffic. One option for reducing network costs is to use caches—exploiting the observation that content popularity is consistent with Zipf's law. Yet, Web caching became unprofitable due to the increase in popularity of dynamic Web content. However, since at this point rich content is not very dynamic caching appears to be worthwhile again.**

We base our analysis on anonymized traces from a large European ISP connecting more than 20,000 residential DSL customers to the Internet, collected in 2009. We focus on the most prominent protocols in this environment—HTTP, BitTorrent (BT), eDonkey, and NNTP—and estimate the potential of caching for traffic reduction. On the one hand, our results show that the potential for caching most client/server-based applications like HTTP and NNTP is small. On the other hand P2P-based applications such as BitTorrent and certain HTTP based applications have high content duplication ratios.

I. Introduction

The Internet has evolved into a system where users can easily share content with their friends and/or other users via applications such as wikis, blogs, online social networks, P2P file-sharing applications, One-Click Hosters, or video portals, to name a few of the most well-known user generated content (UGC) services. In terms of volume, multi-media content including photos, music, and videos, as well as software downloads and updates, are major contributors and together responsible for most of the Internet traffic [11], [12], [16], [17]. Indeed, HTTP is again accounting for more than 50 % of the traffic [11], [12], [16], [17] and is hardly (mis-)used as transport protocol for other applications [12]. Among the causes for the increase of HTTP traffic are One-Click-Hosters such as rapidshare.com or uploaded.to and the increase of streaming content, e. g., offered by youtube.com.

In the early stages of the Internet Web caches were very popular. However, the efficiency of Web caches [7], [15] decreased drastically as the popularity of advanced features increased: dynamic/personalized Web pages (via cookies), AJAX-based applications, etc. Reexamining today's content we find that a large fraction, especially multi-media content, is static and therefore it might be rewarding to re-evaluate the caching potential. This is confirmed by recent caching efficiency studies for specific applications, e. g., Gnutella [8], Fasttrack [20], YouTube [22], BitTorrent [9], and Web [6]. Rather than focusing on a specific application, we in this paper study a *set of application protocols*. For these we investigate the potential of caches, traffic redirection for sharing content, as well as causes of non-cacheability.

In this paper we present observations based on passive packet-level monitoring of more than 20,000 residential DSL lines from a major European ISP. This unique vantage point coupled with our application protocol analysis capabilities enables a more comprehensive and detailed characterizations than previously possible. We focus on the cacheability of multiple applications that are predominantly used for sharing content in our environment[1]: *(i)* HTTP, *(ii)* BitTorrent, *(iii)* eDonkey, and *(iv)* NNTP. Note, that HTTP and NNTP are client/server protocols while BitTorrent and eDonkey are P2P protocols. It might seem odd to include NNTP, the protocol used by Usenet, but we surprisingly find that NNTP accounts for more than 2 % of the total traffic volume and seems to be used as an alternative for file-sharing [10]. In previous work, Karagiannis et al. [9] study the cacheability of BitTorrent before it became one of the dominant file-sharing applications based on data from a residential university enviroment. Erman et al. [6] only focus on the cacheability of Web traffic.

We find that the story for caching is ambivalent. For client/server-based applications, including NNTP and some Web domain classes, e. g., One Click Hosters, caching is ineffective. For other HTTP services, e. g., Software/Updates, we observe caching efficiencies up to 90 %. In addtion, for some domains, using opportunistic cache heuristics improves cacheability substantially. For P2P protocols, especially BitTorrent, there is substantial potential for caching if the cache actively participates in the protocol. Moreover, traffic localization via mechanisms, such as those proposed by, e. g., Aggarwal et al. [1], Xie et al. [21], Choffnes and Bustamante [3], and currently under discussion within the IETF ALTO [18] working group, are promising[2]. Traffic localization in effect uses local peers as cache.

II. Data Sets

We base our study on multiple sets of anonymized packet-level observations of residential DSL connections collected at

[1] In previous work Maier et al. [12] found that at our vantage point HTTP is responsible for more than 50 % of the traffic while P2P (BitTorrent and eDonkey) contribute less than 15 %. Even assuming all unclassified traffic to be P2P in total it only accounts for less than 30 %.

[2] The key idea is that ISPs and P2P users collaborate to locate close by peers.

TABLE I
OVERVIEW OF ANONYMIZED PACKET TRACES AND SUMMARIES.

Name	Start date	Dur	Size	Application Volume HTTP	BitTorrent	eDonkey	NNTP
APR09	Wed 01 Apr'09 2am	24 h	>4 TB	58 %	9 %	3 %	2 %
AUG09	Fri 21 Aug'09 2am	48 h	>11 TB	63 %	9 %	3 %	2 %
HTTP-14d	Wed 09 Sep'09 3am	14 d	> 200 GB	corresponds to > 40 TB HTTP			
NNTP-15d	Wed 05 Aug'09 3am	15 d	> 2 GB	corresponds to > 2 TB NNTP			
BitTorrent-14d	Sat 20 Jun'09 3am	14 d	> 80 GB	corresponds to > 5 TB BitTorrent			

aggregation points within a large European ISP. Our monitor, using Endace monitoring cards, allows us to observe the traffic of more than 20,000 DSL lines to the Internet. The data anonymization, classification, as well as application protocol specific header extraction is performed immediately on the secured measurement infrastructure using the Bro NIDS [13] with dynamic protocol detection (DPD) [5]. For HTTP we extend the standard protocol analyzer to compute MD5 hashes across the HTTP bodies. In addition, we developed DPD-based analyzers for NNTP, eDonkey, and BitTorrent [4], including the Azureus Messaging Protocol [2] and the LibTorrent Extension Protocol.

We use an anonymized 24 h packet trace collected in April 2009 (APR09) and an anonymized 48 h trace collected in August 2009 (AUG09). These are the same data sets as analyzed by Maier et al. [12]. While we typically do not experience any packet loss, there are several multi-second periods (less than 5 minutes overall per packet trace) with no packets due to OS/file-system interactions.

For studying long term effects of cacheability we used Bro's online analysis capabilities to collect several anonymized protocol specific trace summaries (BitTorrent-14d, NNTP-15d, HTTP-14d) which span at least 2 weeks. Due to the amount of traffic at our vantage point and the resource intensive analysis we gather the online trace summaries one at a time. Table I summarizes characteristics of the traces, including their start, duration, size, and application mix.

With regards to the application mix, see Table I, Maier et al. [12] find that HTTP, NNTP, BitTorrent, and eDonkey each contribute a significant amount of traffic. Moreover, their total traffic adds up to more than 72 % of the overall traffic at our vantage point. Similar protocol distributions have been observed at different times and at other locations of the same ISP.

Surprisingly, NNTP accounts for more than 2 % of the traffic. However, a detailed investigation [10] shows that 99 % of the current traffic is bound to/from fee-based NNTP servers. These fee-based offers are competing with One-Click-Hosters, as the client/server-based alternative for file-sharing.

III. TERMINOLOGY AND APPROACH

Before delving into the details of the application specific cacheability analysis we introduce our terminology. Caching refers to saving a copy of a reply to a request on a server—the *cache*—with the intention to satisfy subsequent requests for the same content from the cache instead of the origin. For us all requests to a content item except the first one are in principle *cacheable*. We, in this paper, explore the potential for cacheability rather than focusing on specific caching heuristics, thus we are never limited by disk space.

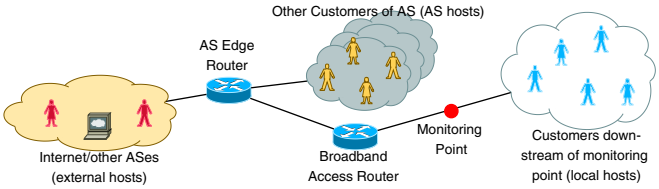


Fig. 1. Vantage point and sets of hosts: *local*, *AS*, and *external*

We use the following two cacheability metrics: number of cacheable requests and cacheable volume. If k_i denotes the total number of downloads for item i of size s_i the cacheable volume of n items is computed as *cacheability* according to Equation (1). The cacheable requests are calculated using the same equation with all sizes equal to 1.

$$cacheability = \frac{\sum_{i=1..n}(k_i - 1) \cdot s_i}{\sum_{i=1..n} k_i \cdot s_i} \quad (1)$$

To estimate the impact of caching it is important to consider which mechanisms are available to redirect requests to the cache and where the cache is located. If P2P content is available at multiple different locations one can use mechanisms [1], [3], [21] currently under discussion within the IETF ALTO [18] group. In addition, it may be beneficial or even necessary to setup dedicated caches within the network. With regards to network location, see Figure 1, we distinguish between *(i)* hosts that are downstream from the monitor, *local* hosts, *(ii)* hosts that are within the ISP's autonomous system (AS) yet not local, *AS* hosts, and *(iii) external* hosts.

A. Cacheability: Peer-to-Peer

In P2P the unit for caching is either the complete object or the transfer unit if the P2P protocol splits the file into chunks, e. g., for more efficient transfers. The former captures the number of users interested in an object. The latter corresponds to the observed traffic. Since users may not download complete objects while online these metrics can differ significantly. In BitTorrent the complete objects are the torrents and the transfer units are the (fixed size) blocks. In eDonkey the objects are the files and transfer units are blocks.

Given that a BitTorrent retrieval can last multiple days it is likely that a any trace includes some partial downloads. Regarding a cacheability analysis the lack of knowledge about transfers that occurred before the start of the observation

period is problematic as these could have primed the cache. Fortunately, for BitTorrent we can leverage the *bitfield* and *have messages* to infer which transfers occurred prior to the trace start using a similar methodology as Karagiannis et al. [9].

Given that P2P clients are also P2P servers we can in addition estimate a lower bound for the cacheability for AS hosts and external hosts. Thus, we can study the potential of P2P cacheability with regards to *(i)* the number of peers interested in an object, referred to as *peers*, *(ii)* transferred blocks, called *blocks*, and *(iii)* transferred blocks given a primed cache, called *primed*, for all three classes of hosts.

B. Cacheability: Client-Server

For HTTP and NNTP we note the following differences: *(i)* Each expression of interest for an object corresponds to a full or partial download of the whole object; *(ii)* we cannot infer information for clients outside of our local network. Moreover, cache control in HTTP 1.1 offers a lot of options via explicit cache-control headers. To identify HTTP objects we use both—the size and the MD5 sum of the HTTP body (*object ID*).

With regards to cacheability the units of interest for NNTP are articles. NNTP articles are comparable to objects in HTTP or emails in IMAP. Commands for accessing articles are `ARTICLE`, `HEAD`, and `BODY`, which request to download the whole article, only the headers, and only the body, respectively. Although NNTP includes a large number of commands the above ones are the only ones not used for navigating, controlling, and listing. Accordingly, the other commands contribute only a tiny fraction (less than 2‰) to the overall NNTP volume. NNTP differs from HTTP as article IDs are unique and articles cannot be modified except by the administrator. To identify objects we thus use the article ID.

We study a range of different cacheability scenarios ranging from *ideal* to *realistic*, see Table II. The *ideal* scenario is used to derive an upper bound on the cacheability using Equation (1). This is the only scenario applicable to NNTP. However, for HTTP this scenario is practically infeasible as one would have to reliable predict that two requests (different URLs) are for the same content. We observe that CDNs are often performing load balancing and thus serve the same content under URLs that only differ in the host name. Moreover, a realistic cache is typically unable to cache objects if two clients use different query methods or if the return code differs. The *domain* scenario thus adds these three criteria to the object ID, and therefore gives us insight on the impact of load balancing over different servers. To account for the observation that identical object IDs can be hosted by the same providers at different URLs (host name and/or path), e. g., to help with load balancing or to accommodate GET parameters we use the scenario *complete*.

In the fourth scenario, *full*, we simulate an actual HTTP cache which respects the cache control headers: `Pragma`, `Cache-Control`, `Expires`, `Last-Modified`, `Etag`, `Authorization`, and the HTTP methods as specified in RFC 2616. However, unlike a real cache, we are still using the object ID. We use this scenario only to evaluate the negative impact of the object ID on cacheability.

TABLE II
CRITERIA FOR IDENTIFYING CACHEABLE OBJECTS IN HTTP.

scenario	object ID	HTTP method	return code	2nd level domain	host	path	cache control
ideal	✔						
domain	✔	✔	✔	✔			
complete	✔	✔	✔		✔	✔	
full	✔	✔	✔		✔	✔	✔
realistic		✔	✔		✔	✔	✔

In the fifth and last scenario, *realistic*, the cache behaves like an ideal caching proxy server with unlimited disk space. The cache always performs the best possible caching option. E. g., if an object is stale it is not purged from the cache. Rather the next query for the same object causes a conditional request to the HTTP server, thus allowing the refresh of a cached object without actually downloading it. If an object is already partially cached when a request occurs, only the missing parts are fetched from the server if the object has not changed in the meantime. If no expiration time is set we use a heuristic motivated by RFC 2616: $t_{expiry} = t_{now} + min(0.1 \cdot (t_{now} - t_{Last-Modified}), 1\,\text{day})$. Table II summarizes the five HTTP scenarios. Note, only the scenarios *full* and *realistic* are using timeouts.

IV. RESULTS

Our analysis shows that the potential cacheability differs substantially among the application protocols.

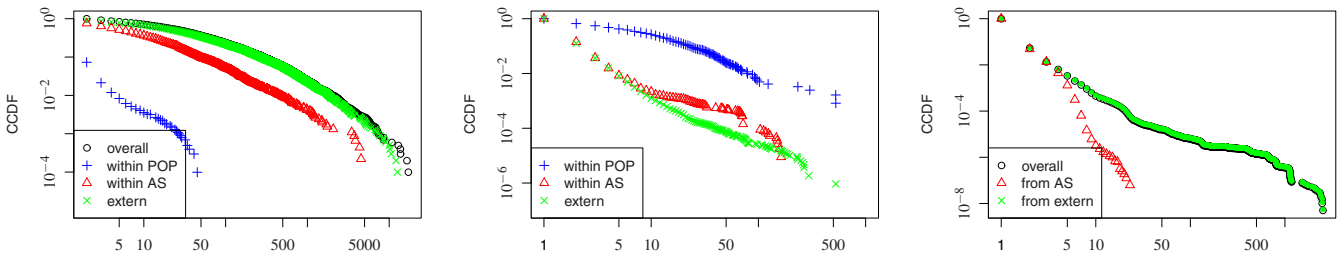
A. Cacheability of Peer-to-Peer

Even though P2P is no longer dominating residential traffic, P2P still contributes a significant amount of traffic. In fact, BitTorrent and eDonkey are the second and third most voluminous protocols after HTTP, with more than 10 % of the total traffic volume.

In Figure 2 we show the complementary cumulative distribution functions of the distribution we use to calculate cacheability of BitTorrent for BitTorrent-14d. The first indicator for estimating the potential of caching is the number of peers per torrent swarm, scenario *peers*. Figure 2 (a) shows this distribution for local, AS, external, and all hosts. We see that many torrents have a sizable number of peers within the AS.

If all peers within the AS would do a complete download 97 % of the bytes are downloadable from peers within the AS—corresponding to an AS-wide caching efficiency of 97 %. This is a lower bound as there may be other peers within the AS. When we consider only local hosts caching efficiency drops substantially to 27 %. Nevertheless, this is still promising for caching.

When focusing on scenario *primed* we find that 85 % of the download volume are in principle cacheable. However, this cacheability result relies on the availability of hosts that are torrent seeders (learned via the bitfield and have messages). We find that some hosts are seeders for a large number of torrents,



(a) Peers per torrent (b) Torrents per peer (c) Downloads per block

Fig. 2. Complementary cumulative distribution functions (CCDFs) of BitTorrent analysis for BitTorrent-14d (note the distinct scales).

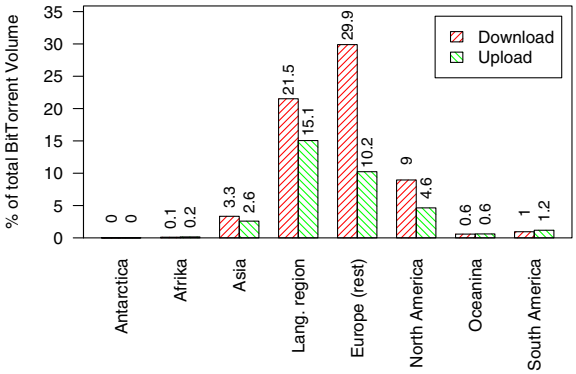


Fig. 3. Normalized BitTorrent traffic volume

see Figure 2 (b), and that these are online for substantial time periods. This is consistent with the results of Stutzbach and Rejaie [19]. However, if we only consider actual downloads, scenario *block*, the cacheability drops to roughly 9 %, see Figure 2 (c), contrary to the previous results. Part of the reason is that we may not have observed a flashcrowd effect during the 14 day observation period. Another reason is that clients often are seeders for many torrents but are only actively involved in a small number of downloads. For data from 2004 from a residential complex at a university Karagiannis et al [9] found that the cacheability for actual downloads is between 6 and 11.8 % which is consistent with our results. However, they saw a substantially lower overall cache efficiency with less than 18.5 %.

We also note that almost all (> 89 %) of the chunks are downloaded form and uploaded to external hosts even though the content is available locally, see Figure 2 (c). We are able to identify such content by inspecting the bitfield messages of the peers. This shows the potential of both P2P neighbor selection strategies [1], [3], [21] as well as caches and confirms the results of Plissonneau et al. [14] for eDonkey and Karagiannis et al. [9].

To check if our observations are biased by the language of the content we examine the geolocation of the BitTorrent traffic with the help of the ISP. Figure 3 shows the amount of BitTorrent traffic downloaded from or uploaded to different

TABLE III
CACHEABILITY OF NNTP ARTICLES

Cacheability	APR09	AUG09	NNTP-15d
by number of requests	2.0 %	2.6 %	7.0 %
by volume	2.1 %	2.7 %	6.9 %

locations normalized by the *overall* BitTorrent traffic volume. We distinguish between continents and within Europe we separate the local language region of the vantage point from the rest of Europe. Only one third of the content is being downloaded from the same language region. This indicates that while there many be some bias due to language it is not the dominating effect with regards to cacheability. The overall download volume is roughly twice that of the upload volume. This effect may be due to the asymmetry of the underlying DSL lines. In addition, Europe seems to be preferable as the upload to download ratio is better.

For eDonkey the cacheability results for peers per file and transferred blocks are not quite as good. For a 24h trace from September 2008 we find that the cacheability per file is 71 % within the AS. However, when considering blocks the cacheability again drops substantially to roughly 4 %.

B. Cacheability of NNTP

Our caching analysis of NNTP shows that the majority of all articles is requested exactly once, see Table III. We find that less than 7 % of all articles are downloaded multiple times. With respect to volume we observe similar results—the cacheability is again less than 7 %. While we cannot identify a temporal trend cacheability increases with the length of the observation period, i. e., NNTP-15d, and thus also the number of different DSL lines using the NNTP protocol. However, we note that a small number of lines, less than 2 %, are using NNTP. However, NNTP users are usually among the top volume users.

C. Cacheability of HTTP

Given the number of different HTTP scenarios we first discuss some general observations. Next, we explore if cacheability differs by Web service, how it scales with population size, and what might be possible cache optimizations. In the following we report results only for HTTP-14d as the results are consistent across all traces.

TABLE IV
EFFECTIVE CACHE CONTROL HEADERS

Cache control header	Frequency
`Cache-control`	57.2 %
`Pragma`	0.5 %
`Expires`	1.7 %
`Etag`	22.8 %
`Last-Modified`	6.8 %
none	12.0 %

TABLE V
CACHEABILITY OF TOP 15 DOMAINS (BY BYTES)

type of service	UGC	fraction	*complete*	*full*	*realistic*
OCH1	✔	12.6 %	2.6 %	0.0 %	1.5 %
OCH2	✔	1.3 %	6.0 %	1.1 %	2.0 %
OCH3	✔	1.0 %	7.1 %	0.0 %	0.2 %
OCH4	✔	0.9 %	2.6 %	0.0 %	0.1 %
Video1	✔	10.8 %	13.0 %	0.0 %	3.8 %
Video2	✔	2.2 %	43.4 %	0.1 %	5.2 %
Video3	✔	1.4 %	7.0 %	0.0 %	0.2 %
Video4	✔	1.4 %	14.7 %	1.5 %	3.6 %
Video5	✔	1.1 %	11.7 %	2.5 %	9.0 %
Software1		2.8 %	63.0 %	68.6 %	64.8 %
Software2		1.8 %	22.0 %	2.9 %	87.4 %
Software3		0.9 %	12.9 %	3.3 %	54.7 %
Software4		0.8 %	56.9 %	42.7 %	65.4 %
CDN1	?	1.5 %	34.8 %	12.8 %	25.4 %
Search	?	1.0 %	56.0 %	5.3 %	32.7 %
overall		100.0 %	21.0 %	9.5 %	21.7 %

General Observations: In principle, there is substantial potential for caching HTTP (see Table VI). In the *ideal* scenario 71 % of the requests are cacheable and 28 % of the bytes. This is consistent with previous results [7] which also observe a significantly higher request hit rate than byte hit rate.

Disabling caching across different second level domains, scenario *domain*, does not decrease the caching efficiency by much. It is still 71 % for requests and 27 % for bytes. Disabling caching across different URLs for the same object, scenario *complete*, causes the efficiency to drop to 57 % and 21 %. This shows that identical objects are usually not hosted by different providers, while for each provider it is quite common to host the same object on different hosts or with different paths.

Including cache control headers, the scenario *full*, reduces cache efficiency drastically to 16 % and 9.5 %. However, in the *realistic* scenario overall cacheability increases to 47 % and 22 %. The omission of the object ID is responsible for this increase in cacheability: *(i)* the cache may be allowed to serve an object that has changed on the server; *(ii)* aborted downloads and partial requests lead to different object IDs and are thus only cacheable in the *realistic* scenario.

The results of Erman et al. [6] for the *ideal* scenario are even more promising. They found a cacheability of 92 % for requests and 68 % for bytes. Their final results after considering cache-control also show a substantial drop but again indicate a better cache hit rate with 32 % of the bytes. One reason for the more promising results are that they assume that the size of the download is equal to the `Content-Length` header. However, we find that many downloads are interrupted prematurely. In particular large downloads are often aborted.

TABLE VI
OVERALL HTTP CACHEABILITY

	ideal	domain	complete	full	realistic
Bytes	28 %	27 %	21 %	9.5 %	22 %
Requests	71 %	71 %	57 %	16 %	47 %

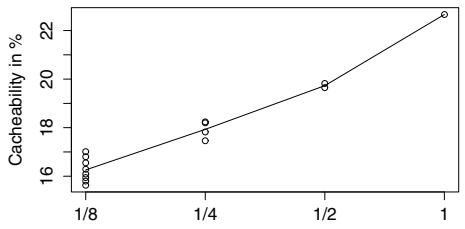


Fig. 4. Cacheability dependent on fraction of population

We find that this can lead to measurement error of over 40 % in download volume. Moreover, Erman et al. only consider the `Cache-Control` header. However, one third of the replies with cache control are controlled by a different header, e. g., `Expires`, as listed in Table IV.

Individual Sites: Table V shows the cacheability for the top 15 domains (by bytes) classified according to the Web service that they offer for the scenarios *complete*, *full*, and *realistic*. In column UGC we mark if a domain is dominated by user generated content. When comparing scenarios *complete* and *full* we see that respecting cache control headers often has a devastating effect on cacheability for some domains. When comparing scenarios *full* and *realistic* we see that the negative impact of the object ID also differs across sites. The site Software1 differs: realistic cacheability is lower than predicted by the *full* scenario. This can occur if an intermediate HTTP request invalidates the cache before allowing access to a resource, e. g., when delivering a login page instead of the requested object upon missing authentication cookies.

There are significant differences among the Web hosters. Sites offering software appear to ensure good cacheability (e. g., > 50 %) and can take advantage of caching (e. g., $>$ 60 % for the *realistic* scenario). Some sites hosting videos have substantial potential for caching (> 40 %) but do not take advantage of it. CDNs also have potential but realize it only partially (34.8 vs. 12.8 %). One click hoster (OCH) have hardly any caching potential (< 8 %) and do not even take advantage of the little potential. The caching hit rate for the *realistic* scenario is less than 2 %. We observe that sites dominated by user generated content exhibit considerably lower cacheability than other sites, e. g., software hosters.

Population Size: Next, we explore the impact of the population size on cacheability. We randomly subdivide our population into smaller sub-populations and recompute the cacheability for the *realistic* scenario. We observe that cacheability appears to increase with population size. When doubling the population size the increase in cacheability ranges from 1.6 % to 2.9 %, cf. Figure 4. We presume that the caching potential further increases with an increase in population. However,

TABLE VII
OPPORTUNISTIC CACHING FOR TOP 15 DOMAINS (SCENARIO *realistic*). DOMAINS WITH NO IMPROVEMENT ARE NOT SHOWN.

		improvement (percentage points)				
type	baseline	10 s	10 min	1 h	1 d	∞
OCH1	1.5 %	8.1 %	16.6 %	17.0 %	17.8 %	18.4 %
OCH2	2.0 %	0.0 %	0.1 %	0.1 %	0.1 %	0.1 %
Video4	3.6 %	0.2 %	1.0 %	1.2 %	1.3 %	1.3 %
Video5	9.0 %	0.0 %	0.1 %	0.1 %	0.3 %	0.4 %
Software3	54.6 %	0.0 %	0.0 %	0.0 %	0.0 %	0.1 %
CDN1	25.4 %	0.1 %	3.0 %	9.5 %	19.5 %	23.7 %
Search	32.7 %	0.0 %	0.3 %	0.5 %	0.7 %	0.7 %
overall	21.7 %	1.1 %	2.6 %	2.9 %	3.5 %	4.0 %

there may be saturation effects. Also note that the variability of cacheability increases with decreasing population size.

Cache Optimizations: Next we explore why the potential for HTTP caching is not used. For this purpose we allow violations of the strict caching semantics for two high-volume sites. For Video1 we study the impact of personalization and load balancing over servers with different host names. We start at a baseline of 3.8 %. Removing personalization, i. e., parameters, from URLs yields an increased cacheability of 20.1 %. Unifying host names increases cacheability to 24.6 %. Thus, we conclude that personalization can be a major cause for non-cacheability of objects.

Some objects do not include any information regarding their cacheability. Thus in principle they cannot be cached. This may be either intentional, or by negligence of the operator. We now explore if opportunistic caching, meaning setting artificial expiry times and thereby violating strict cache semantics, can help for such objects. More specifically, we examine expiry times of 10 s, 10 min, 1 h, 1 d, and infinite.

The overall effect of opportunistic expiries is only small: 2.6 % increase for a ten minutes timeout and 4.0 % for infinite caching. However, OCH1 and CDN1 show a large increase in cacheability. For OCH1 even ten minutes is sufficient to gain most of the benefits. Further investigation shows that misconfigured download accelerators are responsible. Such accelerators download large objects across multiple parallel connections. While this is not per se harmful, these accelerators issue partial requests for overlapping regions. As soon as the desired data is fetched, the accelerator closes the connection. However it takes time to cancel the transaction, and therefore additional data is downloaded. We observe clients that open up to 300 parallel connections resulting in an increase of the download volume by a factor of three. With some cache tuning such extra downloads can be eliminated with a small opportunistic timeout.

V. SUMMARY

Our analysis of 20,000 residential broadband DSL lines of a large European ISP shows that contrary to recent work caching is not necessarily beneficial. For NNTP and some Web domain classes, including sites dominated by user generated content, we hardly find any potential. However, some Web service provider can take advantage of caches, e. g., software download providers or CDNs have substantial potential even if unused due to cache control, personalization, and load balancing.

Caching for P2P protocols is in principle very promising, especially when combined with P2P neighbor selection strategies. However, taking advantage of the potential is non-trivial as the cacheability for simple chunk downloads drops to 9 %. Therefore, we plan to explore how to take advantage of the potential caching rates of more than 95 % for BitTorrent in future work.

REFERENCES

[1] AGGARWAL, V., FELDMANN, A., AND SCHEIDELER, C. Can ISPs and P2P users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev. 37*, 3 (2007).

[2] Azureus messaging protocol. http://www.azureuswiki.com/index.php/Azureus_messaging_protocol, Apr 2009.

[3] CHOFFNES, D. R., AND BUSTAMANTE, F. E. Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *Proc. ACM SIGCOMM* (2008), pp. 363–374.

[4] COHEN, B. The BitTorrent Protocol Specification. http://bittorrent.org/beps/bep_0003.html, 2008.

[5] DREGER, H., FELDMANN, A., MAI, M., PAXSON, V., AND SOMMER, R. Dynamic application-layer protocol analysis for network intrusion detection. In *Proc. Usenix Security Symp.* (2006).

[6] ERMAN, J., GERBER, A., HAJIAGHAYI, M. T., PEI, D., AND SPATSCHECK, O. Network-aware forward caching. In *Proc. World Wide Web Conference* (2009).

[7] FELDMANN, A., CACERES, R., DOUGLIS, F., GLASS, G., AND RABINOVICH, M. Performance of web proxy caching in heterogeneous bandwidth environments. In *Proc. IEEE INFOCOM* (1999).

[8] HEFEEDA, M., AND SALEH, O. Traffic Modeling and Proportional Partial Caching of Peer-to-Peer Systems. *IEEE/ACM Trans. Networking 16*, 6 (2008).

[9] KARAGIANNIS, T., RODRIGUEZ, P., AND PAPAGIANNAKI, D. Should Internet Service Providers fear peer-assisted content distribution? In *Proc. ACM Internet Measurement Conference* (2005).

[10] KIM, J., SCHNEIDER, F., AGER, B., AND FELDMANN, A. Today's usenet usage: Characterizing NNTP traffic. In *Proceedings of the 13th IEEE Global Internet Symposium* (March 2010).

[11] LABOVITZ, C., MCPHERSON, D., AND IEKEL-JOHNSON, S. NANOG 47: 2009 internet observatory report. http://www.nanog.org/meetings/nanog47/abstracts.php?pt=MTQ1MyZuYW5vZzQ3&nm=nanog47.

[12] MAIER, G., FELDMANN, A., PAXSON, V., AND ALLMAN, M. On dominant characteristics of residential broadband internet traffic. In *Proc. ACM Internet Measurement Conference* (Nov 2009).

[13] PAXSON, V. Bro: A system for detecting network intruders in real-time. *Computer Networks 31*, 23–24 (1999).

[14] PLISSONNEAU, L., COSTEUX, J.-L., AND BROWN, P. Detailed analysis of eDonkey transfers on ADSL. In *Next Generation Internet Design and Engineering, 2006. NGI '06. 2006 2nd Conference on* (2006).

[15] RABINOVICH, M., AND SPATSCHECK, O. *Web Caching and Replication*. Addison-Wesley Professional, 2001.

[16] SANDVINE INC. 2009 global broadband phenomena. http://www.sandvine.com/news/global_broadband_trends.asp, 2009.

[17] SCHULZE, H., AND MOCHALSKI, K. Internet study 2008/2009. http://www.ipoque.com/resources/internet-studies/ (need to register), 2009.

[18] SEEDORF, J., AND BURGER, E. W. Application-layer traffic optimization (ALTO) problem statement. RFC 5693, Oct 2009.

[19] STUTZBACH, D., AND REJAIE, R. Understanding churn in peer-to-peer networks. In *Proc. ACM Internet Measurement Conference* (2006).

[20] WIERZBICKI, A., LEIBOWITZ, N., RIPEANU, M., AND WOŹNIAK, R. Cache Replacement Policies Revisited: The Case of P2P Traffic. In *Cluster Computing and the Grid* (2004), pp. 182–189.

[21] XIE, H., YANG, Y. R., KRISHNAMURTHY, A., LIU, Y. G., AND SILBERSCHATZ, A. P4P: Provider portal for applications. In *Proc. ACM SIGCOMM* (2008), pp. 351–362.

[22] ZINK, M., KYOUNGWON, S., YU, G., AND KUROSE, J. Watch Global, Cache Local: YouTube Network Traffic at a Campus Network. In *Proc. SPIE* (2008), vol. 6818.