Cacheability of YouTube Videos in Cellular Networks

Fehmi Ben Abdesslem
SICS Swedish ICT
fehmi@sics.se

Anders Lindgren
SICS Swedish ICT
andersl@sics.se

ABSTRACT

Video traffic now represents a growing proportion of the traffic on cellular networks, causing capacity problems for operators and increased delays for users. Studies have shown that deploying caches at the network level reduces the delay for the end-user and the overall traffic volume for the telecom operator. In this paper, we analyse a large nation-wide dataset of real-life video requests sent by mobile users to a popular video streaming website. This analysis is the first to rely on such a large dataset, and sheds light on the optimal cacheability of video content with caches distributed in the cellular network, and how efficient some existing cache replacement algorithms are at reducing the number of requests sent to the video provider. We show that depending on the cache size and algorithm parameters, up to 20.33% of the requests can be served by a local cache.

1. INTRODUCTION

Over the past few years, traffic from wireless and mobile devices has grown at a rapid rate. It is expected to exceed traffic from wired devices by 2016, and mobile data traffic will increase 13-fold between 2012 and 2017 [6]. This increase in traffic is to a large extent driven by video content, which is expected to make up 69% of all consumer Internet traffic in 2017 (up from 57% in 2012). This is becoming an increasing problem for cellular network operators, as a large part of their available bandwidth is consumed by video traffic. In particular, the backhaul from the base station is often a bottle-neck in these systems. Therefore, the introduction of information-centric networking architectures [7, 12] and caches at base stations show great potential to improve performance and reduce load on the system. There are three main benefits to caching [8]:

- The servers delivering the content receive less requests, hence reducing the resources needed and the risks of overload.
- The network bandwidth usage is reduced, allowing more traffic to go through links and equipments.
- The end-users perceive a reduced delay and potentially a better quality of experience when watching a video.

AllThingsCellular'14, August 22, 2014, Chicago, IL, USA.

http://dx.doi.org/10.1145/2627585.2627588.

In this paper, we determine the cacheability (the maximum cache hit ratio) of YouTube content in mobile networks. When users are accessing videos from the cellular network, we analyse to what extent videos can be dynamically stored in local caches, so that future local requests can be answered directly by the local cache without involving the core network and the YouTube servers.

To understand the efficiency of such approach, we use a large dataset containing more than 75 million requests sent by more than 3 million users. To the best of our knowledge, this is the first study to look at such a large dataset to evaluate the potential of caching video content in cellular networks.

Caching is only useful is a substanstial amount of users are accessing the same content – therefore, in this paper we address the question *"Do most users watch the same videos of cute kittens?"* and is this commonality large enough to motivate the deployment of in-network caches? We first investigate the potential gain achievable by looking at the YouTube requests sent by users from different cells. We then investigate different caching algorithms with varying parameter settings to see how close they can come to providing an optimal caching efficiency.

2. RELATED WORKS

Early studies explored cacheability of web resources by defining static content as cacheable, and dynamic content as not cacheable. Cao and Irani [4] conducted a trace-driven study of web cacheability, analysing more than 24 million accesses to web proxies. They investigated the performance of different caching algorithms and showed high cacheability (up to 50% cacheable content). This was confirmed by Mills and Mikhailov [17], who showed that more than 40% of HTML resources and 60% of images do not change, exploring over 1.8 million requests from proxy cache log traces.

The internet in general and the web in particular have since evolved and the content available is very different from the content studied in these previous works. For instance, user generated content (UGC) websites are now a large part of user activities, with websites offering a large variety of contents generated by users, such as question-answer databases, digital video, blogging, podcasting, forums, reviews, social networking, social media, or mobile phone photography. The amount of static content (not dynamically generated) represents a much higher proportion of the traffic nowadays, and it is assumed that most of the content is cacheable. Therefore, the cacheability is now simply defined by recent studies as the fraction of requests that can be answered by a cache.

Ager *et al.* [2] focus on UGC websites in general to study their cacheability. Their study relies on a large dataset collected by a European ISP, and mainly containing HTTP requests sent by 20,000 households during a couple of weeks. However, the study does not focus on YouTube cacheability in particular. Zink *et al.* [18] re-

lied on a dataset containing HTTP headers collected in a campus network, between clients in the campus and YouTube servers. The data covers 8 days and around 5,000 unique clients, but contrary to the dataset we used, the data was not generated by mobile devices. Other more recent studies such as in [5] explore datasets of UGC video requests, but not from mobile devices and without looking at the cacheability.

A more recent study by Arvidsson *et al.* analyses the demand patterns for YouTube of 35,000 mobile devices over several weeks [3]. The authors study the cacheability of YouTube content on a traditional network, with cache proxies at the access points, whereas we study the cacheability in cellular networks. Guillemin *et al.* [11] also study the cacheability of YouTube videos but with a single cache for each of the three cities studied. Our study considers different cache replacement algorithms, covers a nationwide network, and considers one cache per cell instead of one cache per city.

Cacheability in cellular networks has been studied by Erman *et al.* [9], who have explored HTTP requests collected by a large wireless carrier from several million users and over a period of 36 hours. However, the authors studied the cacheability of HTTP traffic in general and not of YouTube content. Ramanan *et al.* [16] also studied a large dataset collected from a live cellular network over 24 hours. While the authors did study video content in general and YouTube in particular, they did not analyse the cacheability of YouTube video content. They studied instead the cacheability of all YouTube content together, including videos, but also images and HTML pages.

Our work is the first study based on a nation-wide dataset that investigates the cacheability of UGC content in cellular networks.

3. DATASET AND METHODOLOGY

The analysis performed in this paper is based on a dataset of cellular network traffic traces from a large European operator that were collected on a national scale over a period of around 41 days between December 2011 and January 2012. The traces contain the URL of around 30 billion HTTP requests together with timestamps and an anonymised identifier of the user sending it, and the network cell the user was connected to when sending the request.

For the sake of this study, we focus on requests that are sent to YouTube and disregard all other traffic. We extract the video identifiers from the request URLs to identify the different videos requested by users, along with the anonymised cell identifier used. There are roughly 3 million YouTube users in the dataset, and they account for roughly 30% of the total users appearing in the dataset. YouTube users have sent around 75 million video requests for 10 million different videos. A video request from a user is identified by a set of HTTP requests sent to retrieve chunks of that video by that user. Hence, a video request typically triggers several HTTP requests.

Since we are looking at the potential for caching video content, we only study cells that received at least 1,000 requests for YouTube videos, as we are not interested in isolated cells attracting less traffic, such as in rural areas. We consider that the cost of deploying a cache system in such cells is not worth the benefits.

The data available in our dataset does not allow us to accurately determine the exact size of each video. Therefore, for our analysis, we count the cache size and the achievable cacheability in terms of number of videos. Note that a requested video might only be downloaded partially, as the user might decide to abort the download and stop playing the video. The caching systems studied hereafter can be optimised by only caching partial contents. This is out of scope of this study.

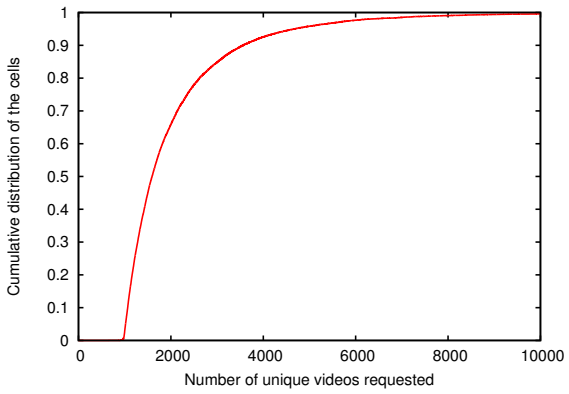


Figure 1: Cumulative distribution of the number of unique videos requested for each cell.

The resulting sub-set is composed of around 40 million requests sent by 2 million users to approximately 20,000 cells and targeting 6.6 million unique videos. The median number of unique videos requested for each cell is 1,593, with an average of 2,309. Figure 1 shows the distribution of the cells per number of unique videos requested.

4. CACHING ALGORITHMS

We consider a scenario where each cell of the network maintains a cache to answer requests of the users in the cell.

Initially, when the cache is not full, every newly requested content is stored so that the next queries for that content can be directly answered by the cache. When the cache is full, and the requested content is not in the cache memory, the replacement strategy dictates which content, if any, must be deleted from the cache and replaced by the new content.

Caching algorithms found in the litterature follow different strategies to replace a content when the cache is full [15]. While different classifications can be used for the replacement strategies ([13, 14]), we use the one proposed by Podlipnig and Böszörmeny [15]: frequency-based, recency-based, randomised, recency/frequency-based, and function-based.

Our dataset is not suitable to evaluate function-based strategies, relying for example on object size. Instead, we focus on typical algorithms implementing frequency- and recency-based strategies, and compare with a randomised strategy.

4.1 Least Frequently Used (LFU)

The Least Frequently Used (LFU) cache replacement policy is a frequency-based strategy where the least frequently accessed content (accessed the least number of times) in the cache is replaced when the cache is full. In this paper we consider two main versions of this algorithm:

In-cache LFU only keeps a counter for videos that are already in the cache, which means that if some content is evicted from the cache and later another access for that content happens, the previous history for that content is lost. When a piece of content which is not in the cache is requested, it will always be added to the cache and replace some other content if the cache is full.

Perfect LFU keeps track of the frequency of all videos ever accessed. Perfect LFU is more complex with higher overhead than in-cache LFU, as the cache must keep a counter for every content requested. This in turns will give it a possibility to remember the history of content that is not currently in the cache. The perfect

LFU strategy only replaces an existing content in the cache when the new requested content has a higher frequency than the content in cache with the minimum frequency.

In their basic version, both of these LFU variants count the frequency of the requests since their first request. However, the video popularity is dynamic and changes over time. If a video was popular for a period of time and then loses its popularity, the video will be kept in the cache even if there are no more requests for this video. To alleviate this, we use a version of LFU with a *time window*, which only remembers requests from the last time window in the past, removing older data. We use several time windows from one hour to ten days, and in also consider an infinite time window, which is equivalent to the basic version of LFU.

4.2 Least Recently Used (LRU)

Recency-based strategies use recency as a main factor when deciding which content to evict from the cache. The most common such strategy is Least Recently Used (LRU) that has been applied successfully in many different areas.

In our evaluations, we keep track of the most recent request for all content items currently in the cache. When a new item (not currently in the cache) is requested, it is always added to the cache, and the item in the cache that has not been requested for the longest time is evicted.

4.3 Random

Instead of relying on frequency or recency, a random replacement strategy removes randomly one content from the cache and replaces it with the newly requested content. This has shown to be a valid alternative to more complex replacement strategies [10]. Note that frequent contents will still persist in entering the cache, and if frequent contents are few compared to all the other contents requested, they are less likely to be evicted when cache size is large. This makes the random strategy a valid strategy for some contexts.

5. RESULTS

The main metric commonly used to evaluate the cache algorithms is the hit ratio. Other metrics include bandwidth consumption and latency reduction, for instance. However, our dataset does not allow us to determine accurately the size of the content being requested, nor to analyse the latency experienced by the users during the viewing sessions.

The hit ratio for a given cache proxy can be simply defined as the proportion of requests received by this proxy and answered with a content already in cache. If the content requested is not in cache, then it is sent all the way up to the original servers (here, YouTube servers). The higher the hit ratio, the more efficient is the caching algorithm.

The cacheability of a content in a particular context is defined as the maximum hit ratio reachable by a caching algorithm, assuming an infinite cache size. We use the cacheability definition proposed by Ager *et al.* [2] and used in other works [1, 16], where k_i denotes the total number of requests for video i, and n is the number of unique videos requested:

$$cacheability = \frac{\sum_{i=1}^{n}(k_i - 1)}{\sum_{i=1}^{n} k_i}$$

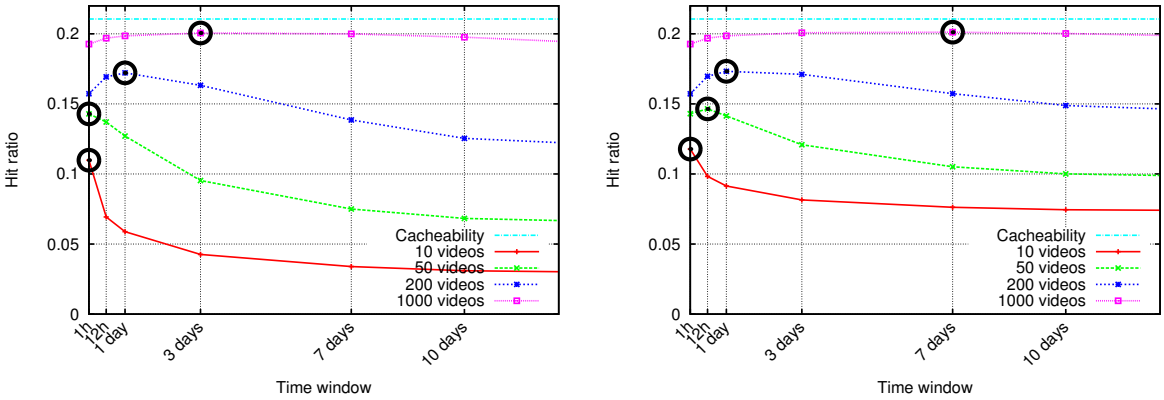
First we consider the overall cacheability for YouTube videos in our dataset. If a single central cache for the whole cellular network was deployed, the cacheability of our dataset would be 83.42%. Note that for YouTube videos this value is much higher than the cacheability observed in previous studies for all HTTP content in general (around 30% [9]) and video content in general (around 50% [16]). However, while it is certainly of interest for the operator to reduce the outgoing and incoming traffic for its network for economical reasons, the main bottleneck is not located between the operator and the content provider (in this case the YouTube servers). Instead, the bottleneck is mainly in the cellular access network, where bandwidth out to base stations can be scarce if usage is heavy. Therefore, in this paper we focus on studying the effects of distributed caches further out in the network. Instead of a central cache, we set up local caches co-located with the base station at each cell, thus allowing us to reduce the traffic in the network and the latency for the end-user. However when a cached video is requested a second time from a different cell, it will have to be cached again in that new cell, reducing the total cacheability. In this section, we consider that each cell has its own local cache, assuming that users sending requests to the same cell share roughly the same location. Hence, the cacheability observed for a local cache at each cell is always lower than the one for a central cache proxy for all the network. In our dataset, the local cacheability is 21.06%.

5.1 LFU algorithms

Table 1 shows the hit ratios for the perfect LFU algorithm, for different cache sizes and different time windows. As expected, the performance increases with the cache size. However, choosing an infinite time window reduces the cache ratio, as it does not take into account the evolution of video popularity. For instance, if a video is very popular for one week, its frequency will keep it in the cache, even though it is not popular anymore in the next weeks. By choosing a limited time windows, the algorithm only takes into account the frequency of the video in that time window, and hence will remove it from the cache if it is no longer popular.

As highlighted by Figure 2, for both versions of LFU the choice of the time windows does affect the hit ratio, and the right trade off depends on the cache size. A short time window of only one hour is better when the cache size is small, whereas longer time windows are more and more efficient when the cache size is bigger. Figure 3 shows the view count over time for two examples of videos from our dataset, and we can see that videos have different popularity patterns over time. Videos like Video A experience bursts of high popularity for short periods of time, while videos like Video B have a lower, but more constant over time, popularity rate. For small cache sizes, it is important to be able to cache mostly type A videos during their popular periods, as they are targeted by a large number of requests and will increase the hit ratio. Due to the small cache size, it is beneficial to rapidly evict such videos once their popularity is over and cache some newly popular video instead. However, those videos are relatively rare, and overall there are more videos like Video B. Thus, when the size of the cache is larger, keeping a record of videos that are popular over a longer time scale, like Video B, allows the cache to keep such videos in the cache as well, which further increases the hit ratio over time. Therefore, longer time windows are more beneficial for large cache sizes, while small caches needs to be dynamic and focus on content that is popular on shorter time scales, making a shorter time window preferential.

The hit ratios for the in-cache LFU algorithm are shown in Table 2. We observe that the values are always equal or greater than the hit ratios obtained by perfect LFU. As for the perfect LFU algorithm, the hit ratio is higher when the cache size is bigger, and the best time window for each cache size is longer when the cache size is bigger. This result shows that for the videos requested in our dataset, it is more efficient to only keep track of the frequency of videos already in cache. In the perfect LFU algorithm, a new video must receive enough views to exceed that of videos already in the



(a) Perfect LFU.

(b) In-cache LFU.

Figure 2: Hit ratios obtained with the LFU algorithms for different cache sizes. The best value for each cache size (circled) is obtained with shorter time windows for small cache sizes, and longer time windows for larger cache sizes.

Table 1: Overall hit ratios with the perfect LFU caching algorithm (track all videos).

Cache size / Time window	1 hour	12 hours	24 hours	3 days	7 days	10 days	infinite
10 videos	10.98%	6.93%	5.88%	4.26%	3.39%	3.10%	2.24%
50 videos	14.29%	13.71%	12.71%	9.53%	7.50%	6.83%	4.90%
200 videos	15.72%	16.93%	17.21%	16.33%	13.86%	12.54%	8.68%
1000 videos	19.27%	19.69%	19.85%	20.06%	20.00%	19.76%	16.04%
infinite	21.06%						

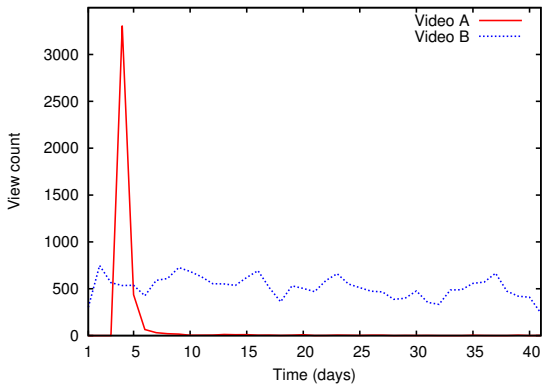


Figure 3: View count of two videos with different popularity patterns. Video A will have a high request frequency during short time windows, while Video B has higher frequency over a longer time window duration. Thus, different caching choices may be made depending on the time window.

cache before an old video is evicted. Thus, there is a risk that new videos that experience a sudden popularity burst may miss many potential cache hits, while in-cache LFU always adds a new video to its cache and evicts something else. In the perfect LFU algorithm, a once-popular videos are more likely to stay in the cache "too" long, and if evicted are also more likely than new videos to be added to the cache again. This will decrease the hit ratio, because that once-popular video will compete with videos that are currently becoming popular.

Table 3: Overall hit ratio with LRU and Random.

Cache size	LRU	Random
10 videos	12.44%	11.41%
50 videos	15.60%	14.67%
200 videos	18.02%	17.20%
1000 videos	20.33%	19.99%
infinite	21.06%	

5.2 LRU and Random algorithms

Removing the least recent video from the cache appears to provide an even better hit ratio than both LFU variants, for every cache size, as shown in Table 3. This result shows that despite being more complex than the LRU algorithm, both LFU algorithms do not perform as well as the LRU algorithm in terms of hit ratio. The same table shows that if the time window is not properly chosen, removing randomly any video from the cache can sometimes give better

Table 2: Overall hit ratio with the in-cache LFU caching algorithm (track only videos in the cache).

Cache size / Time window	1 hour	12 hours	24 hours	3 days	7 days	10 days	infinite
10 videos	11.78%	9.83%	9.14%	8.15%	7.62%	7.45%	6.95%
50 videos	14.30%	14.65%	14.14%	12.09%	10.52%	10.01%	8.61%
200 videos	15.72%	16.97%	17.33%	17.11%	15.73%	14.89%	11.94%
1000 videos	19.27%	19.69%	19.85%	20.08%	20.12%	20.03%	18.34%
infinite	21.06%						

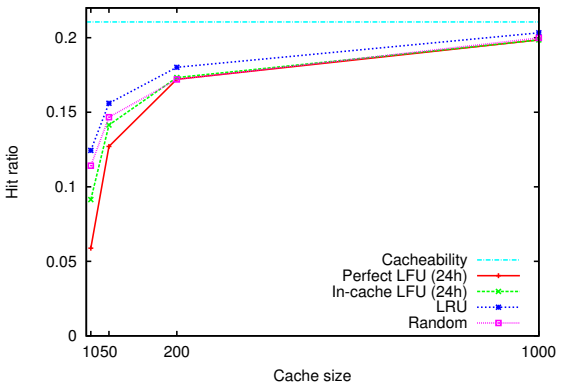


Figure 4: Hit ratios of different replacement strategies.

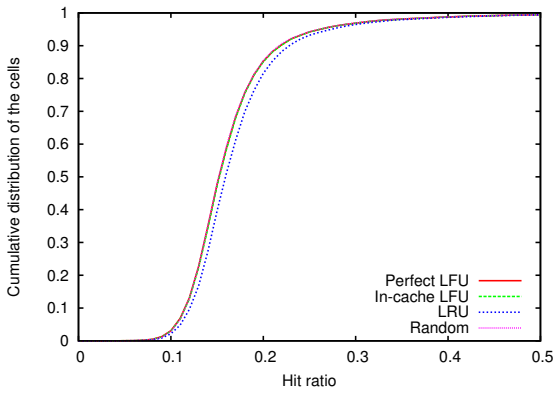


Figure 5: Cumulative distribution of the cells according to their hit ratio for the different cache strategies and cache size of 200 videos. Results for other cache sizes are similar and all show a same hit ratio distribution over the cells.

results than using an LFU algorithm. To obtain the hit ratios for the Random algorithm, we ran three simulations and show in Table 3 the average value. For each of these values, the standard deviation is very low ($< 10^{-4}$), which confirms that the values were not obtained by chance.

The same results can be seen in Figure 4 which compares the hit ratios obtained with different replacement strategies and shows clearly the better performance of LRU.

It is not only important to consider the overall cache hit ratio, but we must also understand the hit ratio for individual cells in the network. If cells have similar hit ratio, a more even performance can be expected, regardless of the location of the user. On the other hand, if there is large variance in the hit ratio between cells, cache deployment can be optimised to only be deployed in high impact locations. Figure 5 shows an example of of the distribution of the cells according to their hit ratio for the different replacement policies. We note that the hit ratios for the cells are rather homogeneous – a large majority of the cells fall in a fairly narrow range of hit ratio values. While the absolute numbers vary for different cache policies, the overall shape of the curves are the same. Most cells have similar characteristics and would experience similar performance improvements from the deployment of caches.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we explored the cacheability of YouTube content accessed from a cellular network. We rely on a large dataset covering a nationwide network and millions of users over a time period of 41 days. To the best of our knowledge, it is the first study on YouTube video cacheability for cellular networks on such a large scale. In particular, we study the local cacheability of videos, where each cell maintains a local cache. We found that in our dataset 21.06% of the requests could be answered by a local cache of infinite size. In the more realistic case where the cache size is limited, we evaluated different classic cache replacement strategies. We found out that we do not benefit from using more complex algorithms such as LFU: replacing the least recent video in the cache (LRU) is enough to obtain even better results.

Using a local cache size of only 50 videos is enough to achieve 15% hit ratio, which is already fairly close to the cacheability of 21.06%, and with a cache size of 1000 videos, we can achieve up to 20.33% cache hit ratio. Increasing the hit ratio while maintaining a limited cache size is important, but considering that even a random replacement strategy gives very good results, as it reduces the network traffic generated by YouTube video distribution with up to 19.99%, we do not believe it meaningful at this moment to look at more advanced cache eviction algorithms to reach the cacheability of 21.06%. Instead we will in future work investigate ways to increase that cacheability. In particular, we are considering to share cached data among mobile devices, and to cache videos at multiple levels in the network by aggregating several cells for a neighbourhood or city into one cache.

7. ACKNOWLEDGMENT

This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 246016.

This work was partially funded by the Future Networking Solutions action line of the European Institute of Innovation and Technology ICT Labs, and by the FP7 Marie Curie IRSES project MobileCloud under grant agreement No. 612212.

8. REFERENCES

[1] H. Abrahamsson and M. Nordmark. Program popularity and viewer behaviour in a large TV-on-demand system. In *Proceedings of 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*. IEEE, 2010.

[3] A. Arvidsson, M. Du, A. Aurelius, and M. Kihl. Analysis of user demand patterns and locality for youtube traffic. In *Teletraffic Congress (ITC), 2013 25th International*, pages 1–9, 2013.

[4] P. Cao and S. Irani. Cost-aware www proxy caching algorithms. In *Usenix symposium on internet technologies and systems*, volume 12, 1997.

[5] L. Carlinet, T. Huynh, B. Kauffmann, F. Mathieu, L. Noirie, and S. Tixeuil. Four months in daily motion: Dissecting user video requests. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, Aug 2012.

[6] Cisco Systems, Inc. Cisco Visual Networking Index: Forecast and Methodology, 2012-2017, May 2013.

[7] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl. Network of information (netinf) - an information-centric networking architecture. *Comput. Commun.*, 36(7), Apr. 2013.

[8] B. D. Davison. A web caching primer. *Internet Computing, IEEE*, 5(4):38–45, 2001.

[9] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, S. Sen, and O. Spatscheck. To cache or not to cache: The 3g case. *Internet Computing, IEEE*, 15(2):27–34, 2011.

[10] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy. Performance evaluation of the random replacement policy for networks of caches. In *Proceedings of SIGMETRICS*, 2012.

[11] F. Guillemin, B. Kauffmann, S. Moteau, and A. Simonian. Experimental analysis of caching efficiency for youtube traffic in an isp network. In *Teletraffic Congress (ITC), 2013 25th International*, pages 1–9, Sept 2013.

[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of CoNEXT*, 2009.

[13] S. Jin and A. Bestavros. Greedydual*: Web caching algorithms exploiting the two sources of temporal locality in web request streams. *Computer Communications*, 24(2):174–183, 2001.

[14] B. Krishnamurthy and J. Rexford. *Web protocols and practice: HTTP/1.1, Networking protocols, caching, and traffic measurement*, volume 108. Addison-Wesley Reading, 2001.

[15] S. Podlipnig and L. Böszörmenyi. A survey of web cache replacement strategies. *ACM Comput. Surv.*, 35(4):374–398, Dec. 2003.

[16] B. Ramanan, L. Drabeck, M. Haner, N. Nithi, T. Klein, and C. Sawkar. Cacheability analysis of http traffic in an operational lte network. In *Wireless Telecommunications Symposium (WTS)*, 2013.

[17] C. E. Wills and M. Mikhailov. Examining the cacheability of user-requested web resources. In *Proc. 4th International Web Caching Workshop*, 1999.

[18] 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.