On the Effects of Caching in Access Aggregation Networks

John Ardelius, Björn Grönvall
Swedish Institute of Computer Science, SICS
{john,bg}@sics.se

Lars Westberg, Åke Arvidsson
Ericsson Research
{Lars.Westberg,Ake.Arvidsson}@ericsson.com

ABSTRACT

All forecasts of Internet traffic point at a substantial growth over the next few years. From a network operator perspective, efficient in-network caching of data is and will be a key component in trying to cope with and profit from this increasing demand. One problem, however, is to evaluate the performance of different caching policies as the number of available data items as well as the distribution networks grows very large.

In this work, we develop an analytical model of an aggregation access network receiving a continuous flow of requests from external clients. We provide exact analytical solutions for cache hit rates, data availability and more. This enables us to provide guidelines and rules of thumb for operators and Information-Centric Network designers.

Finally, we apply our analytical results to a real VoD trace from a network operator and show that substantial bandwidth savings can be expected when using in-network caching in a realistic setting.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Information networks, Performance evaluation

Keywords

Caching, Access aggregation network. Scalability

1. INTRODUCTION

Internet traffic forecasts point at a substantial growth over the next few years and a major part of this traffic is expected to be related to video [3, 8].

From an operators' perspective, a problem is to manage this new traffic at a reasonable cost. A possible solution to this problem is the introduction of in-network caching as has been proposed by the Information-Centric Networking (ICN) community. However, it shall be remembered that in-network caching has benefits not only to operators but also consumers and content providers may benefit from it.

While the amount of video related traffic is increasing, so is the number of distinct available items. Recent studies show, *e.g.* , that there are on the order hundred million different items available in BitTorrent networks [5]. Performance analyses of systems handling such large amounts of data are inherently difficult and mostly lacking.

The purpose of this paper is to introduce a tool or analytical model of in-network caching for truly large scale content catalogues. The tool is then used to gain insights, provide guidelines and, make predictions on the effects of in-network caching that are of importance to the designers of future ICN architectures.

Our aim is to provide a simple tool for assessing the load on caches and links at various levels in the network as well as estimating the availability of data. We remark that an in-network caching architecture must also account for the complex trade-offs between different kinds of hardware solutions and their associated costs. Our tool can provide valuable insights in this decision process.

The main contribution of this work is an analytical model for the performance of caches in truly large scale hierarchical access aggregation networks. The model is applicable to any cache eviction policy and we study two benchmark policies which yield estimates of:

- The hit rate at any level in the cache hierarchy as well as for the access network as a whole for networks containing of the order 10^8 distinct items.
- The specific content held at caches at various levels in the cache hierarchy.
- The overhead costs due to redundant caching.
- The performance gains from varying cache sizes.

Finally, based on traffic data from the network operator Orange, the tool is used in a realistic setting to show substantial bandwidths savings when in-network caching is applied to Video-on-Demand traffic (VoD).

2. BACKGROUND AND SCOPE

Access aggregation networks and cellular back-haul networks typically form tree-like topologies. In fixed networks traffic ingress and egress tends to be concentrated to the leaves and the root of the tree. In contrast to this, cellular base stations are often co-sited with nodes in the tree and traffic may enter or leave at any node in these networks.

ICN'12, August 17, 2012, Helsinki, Finland.

It is hard to give a specific number of child nodes per parent node in the tree as this number depends on the underlying transmission technology and the dimensioning margins for the specific traffic mix, but a typical range is 3–10 and we note that our model is flexible in this respect.

Because of space limitations, the discussion will be limited to access aggregation networks in the fixed domain leaving cellular networks for future work.

In general, we cannot predict topologies or usage patterns of future networks. Nevertheless, we will make the assumption that in-network caching can be deployed at any node in our tree shaped networks.

3. RELATED WORK

The efficiency of caches has previously been subject to benchmark studies in the context of disk accesses [9].

Analytical studies of caches also exist, in [4] the hit rate performance of LRU is studied using an approximate model for a single cache. Our approach is somewhat different in that we are studying the exact solution and in the context of a hierarchy of caches. Other work such as [11] analyze worst (and best) case behavior of caches whereas we are interested in the average case which will dominate for truly large systems.

The performance of LRU caches has also been studied by Che *et al.* in [2] and in the context of hierarchical caching in [6], where a very accurate hit rate approximation per content is given. However, solving the approximation equation for millions of items is neither straight forward nor feasible in practice.

A performance analysis of a two-layer hierarchy using the Che approximation is given in [7] for moderately sized (10^4 items) systems.

Caches operating under continuous data flux have been studied in the context of automatic control in [13], again limited to single caches.

The performance of web caches has been well studied experimentally and using simulations. Few analytical studies treating explicit eviction policies exist, however. A nice presentation of some general web caching techniques can be found in [12].

4. THE NETWORK MODEL

We model an access network as a graph of interconnected nodes (routers). Each node may have any number of clients (hosts) attached which request and receive content. Clients can only be connected to one node at the time. Nodes can either serve requests immediately or relay them to other nodes in the network.

To simplify the analysis we approximate the connection topology of the nodes by a d-regular tree. Leaf nodes at the bottom resemble access routers far out in the network whereas the root is a member of the backbone or core network providing connectivity to the Internet.

Due to the tree structure of the network, a node is connected to one node higher up in the tree and d nodes below except for the leaf nodes at the very bottom which only are connected upwards. The *level* of a node indicates its distance plus one in hops from the leaf level hence the root will have the highest level equal to the depth of the tree and the leafs will be on level one. Each node also has a storage capacity or *cache size* C which determines the number of cache items it can store locally.

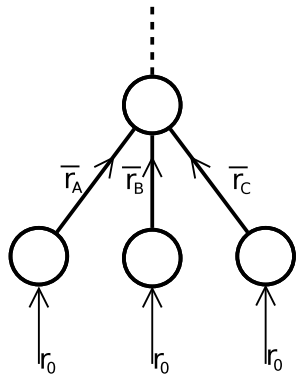

Figure 1: Each leaf node receives a vector of external requests r_0 and relays the requests not served by its cache $\bar{r}_k$ to its parent node.

Further we assume a fixed, large catalogue of N items which can be requested by any node in the network. Content requests can originate either from clients connected to the node, or from other nodes in the network which are unable to service the request from their local caches. In more detail, a node serves all requests for which the content resides in its cache, and relays all other requests to its parent node. This is depicted in Fig. 1.

4.1 Continuous content flow

Rather than examining the behavior of the system at each individual request, we are interested in the overall *average* performance of the system as function of a few key parameters. Given a probability distribution for the content request rates, an eviction policy for the local caches and a particular network topology, we would like to determine the bandwidth usage and content availability for the entire network.

In order to do this, we study the system in the limit where the number of requests per time unit is so large that we may approximate the arrival process of discrete requests by a continuous *flow* which is specified as a rate of requests per unit time. This approach is similar to fluid analysis of queuing systems [10]. Our analysis then operates on distributions of these flows rates rather than on discrete request events. We thus assume that there exists a period of time where the distribution of content requests is approximately static. We further assume that the exchange rate of availble items is such that it is approximately static as well.

In the continuous limit we define a *request vector* $\mathbf{r}$ as a vector of average request rates for each available item,

$$\mathbf{r} \equiv \{\mu_0, \mu_1, \ldots, \mu_N\}.$$

We further assume that two requests for a particular piece of content are independent which motivates us to model the continuous process of requests for item i as a Poisson process with rate λ_i. This is indeed a simplification, but it will provide us with a lower performance bound since correlation among requests can be exploited by a clever caching strategy.

The requests received by a particular node k can then be split up in two parts; one which contains requests from local clients, $\mathbf{r}_0$, and another one which contains requests relayed

from other nodes in the network, $\bar{\mathbf{r}}_k$,

$$\mathbf{r}_k = \mathbf{r}_0 + \bar{\mathbf{r}}_k. \quad (1)$$

As suggested in various studies, *e.g.* [1], the aggregate request intensities from external clients can be fairly well approximated by a power law distribution. Here we assume that intensities follow a Zipf distribution with parameter α which determines the shape of the power law. Typical values for the popularity of Internet content lies in the range $\alpha = 0.6 - 0.9$ [1, 5].

$$\lambda_i(i, \alpha, N) \propto \frac{1/i^\alpha}{\sum_{n=1}^{N}(1/n^\alpha)} \equiv p(i) \quad (2)$$

The item indexes are ordered by decreasing popularity where item 1 is the most popular.

The first term in equation (1) represents requests issued by clients *locally* at the node itself and is obtained as

$$\mathbf{r_0} \equiv \{\lambda_0, \lambda_1, \ldots, \lambda_N\}$$

where the rates are given by the Zipf distribution (2).

The second part of equation (1) represents the flows relayed from other nodes n,

$$\bar{\mathbf{r}}_k \equiv \sum_{n \in M_k} \bar{\mathbf{r}}_{n \to k}. \quad (3)$$

where M_k is the set of nodes which relay requests to node k.

Further, we simplify the analysis by assuming that the network is well balanced (which is only approximately true in practice). The assumption lets us treat each node on the same level interchangeably,

$$\mathbf{r}_i = \mathbf{r}_j; \forall\{i, j : \text{level}(i) = \text{level}(j)\}. \quad (4)$$

This means, due to symmetry of the tree, that the *average* request pattern seen by a node at level x is the same for all nodes at this level.

The probability that a node k holds an item in its cache is given by the vector

$$\mathbf{c}_k \equiv \{c_k^0, c_k^1, \ldots, c_k^N\}; c_k^i \in [0, 1].$$

5. EVICTION POLICIES

There are two basic cache eviction algorithms known as LFU (Least Frequently Used) and LRU (Least Recently Used). Many modern eviction algorithms are variations of these two and use combinations of *frequency* and *recency* in their implementation.

The LFU eviction policy is to remove the item with the smallest number of accesses. Unfortunately this means that the algorithm can not be implemented in constant time hence it is therefore mostly of theoretical importance when applied to large caches.

The other, much simpler policy is LRU. It evicts the item that has not been accessed in the longest time. LRU can easily be implemented in constant time using a double linked list.

Our main concern in this paper is to provide benchmark points for our analytic model. For this reason we provide analytic expressions for the performance of both LFU and LRU under continuous content flow.

Thus, we study the system in steady-state and do not consider temporal variations. This will lead to conservative performance estimates, since variations in principle are exploitable by clever eviction policies.

5.1 Least Frequently Used (LFU)

In our model, when the cache eviction policy is based on frequency information, the node will look at its incoming request vector and compare the frequency of request for the items held in the local cache with requests for other items. The best option for the node, regarding its request vector $\mathbf{r}$ is then to cache the items with maximal average request rate. Let $F_C(\mathbf{r})$ be a function that returns the C largest values from $\mathbf{r}$. The cache probability vector for node k, $\mathbf{c}_k$ will then be given by.

$$\mathbf{c}_k^{\text{LFU}} = \mathbf{1}_{i \in F_C(\mathbf{r}_k)}$$

Where $\mathbf{1}$ is the indicator function. The intuitive interpretation of this LFU model is that if average request rates are static and known, the best policy in steady state is to cache items with high average request rates. This is not however an optimal strategy since temporal variation in requests can motivate eviction of items which are popular in the long run. This LFU model will thus provide us with an average case estimate rather than an upper or lower performance bound.

5.2 Least Recently Used (LRU)

In order to calculate the probabilities for a node k to have an item a in its cache while using the LRU policy we can do this by calculating its complement, namely the probability that the item will be evicted. The eviction probability is determined by the probability the cache receives a sequence of requests **not** containing a while at least $K-1$ other items are requested. We denote this probability by R(a).

Consider a sequence of requests of length s. A *configuration* of s is a vector $K = \{k_1, k_2, \cdot, k_N\}$ with the specific number of requests, k_j, issued for item j in this sequence. $\sum_j k_j = s$. If we denote the set of items different from a by $\bar{a}$, the probability for a specific configuration K of the s requests among the items $\bar{a}$ is given by the multinomial distribution

$$P(\bar{a}, K, s) = \frac{s!}{\prod_{i \in \bar{a}} k_j!} \prod_{i \in \bar{a}} (p(i))^{k_j} \quad (5)$$

Summing over all s we get

$$\sum_{i=C}^{\infty} P(\bar{a}, K, i) = \frac{(\sum_{j \in \bar{a}} p(j))^C}{1 - (\sum_{j \in \bar{a}} p(j))} \quad (6)$$

This is the probability that item a will not appear in the sequence but we also need to make sure that at least $C-1$ other items are requested. This will then cause item a to be evicted.

Let $Q_j(a)$ denote the set of all unique sequences of length s where only j unique items are requested. The probability for such a subset is again given by equation (6). We then need to subtract all such sets where less than $C-1$ unique items are requested since these sets will for sure not cause a to be evicted. However, in each $Q_j(a)$ we will over count a number of subsets of shorter length $j-1, j-2, \cdots$. Therefore, the correct probability is given by the inclusion-exclusion equation below. Taken together the eviction probability for an LRU cache is given by:

$$R(a) = p(a)(\frac{(1-p(a))^C}{p(a)} - Q(a)) = (1-p(a))^C - p(a)Q(a) \tag{7}$$

where

$$Q(a) = \sum_{i=1}^{C-1} Q_i(a) + \sum_{j=1}^{i-1}(-1)^{j+1}\binom{C-(N-1)}{N-1-j}Q_j(a) \tag{8}$$

The probability that a given item resides in cache is then given by:

$$\mathbf{c}_k^{\mathrm{LRU}} = \frac{1-R(k)}{\sum_j 1-R(j)} \tag{9}$$

Equation (9) can, for moderate number of items and cache size, easily be verified by simulation.

For very large numbers of available contents the exact solution is intractable due to the fact that we need to calculate each term in eq (8). However, since each $Q(a)$ only contains at most $N-1$ terms, each less than 1 we will when $C >> 1$ be able to approximate the cache probability by

$$c(k) \simeq 1 - (1-p(a))^C \tag{10}$$

6. BOTTOM-UP REQUEST ROUTING

In the following, content will not be allowed to flow up the tree. Due to the symmetry property of the request vectors (Eq. (4)), we denote the requests from a node at level j, which is same for all nodes at this level, by $\mathbf{r}^j$. Further, due to the regular degree of the tree, the residual flow equation (3) simplifies to

$$\bar{\mathbf{r}}^j = d\bar{\mathbf{r}}^{j-1} \tag{11}$$

The nodes at level 1 (the leafs) will in the bottom-up case only see requests from external client $\mathbf{r}^1 \equiv \mathbf{r}_0$ since no relay traffic reaches the leafs.

The residual flow out of level j is the flow not absorbed by the cache at that level and is given by

$$\bar{\mathbf{r}}^j = \mathbf{r}^j \cdot (1 - \mathbf{c}^j) \tag{12}$$

Using equation (4) again, the request vector for nodes at higher levels becomes

$$\mathbf{r}^j = \mathbf{r}_0 + d\bar{\mathbf{r}}^{j-1} = \mathbf{r}_0 + d(\mathbf{r}^{j-1} \cdot (1 - \mathbf{c}^{j-1})) \tag{13}$$

Solving this recursion relation using equation (11) and (12) gives

$$\bar{\mathbf{r}}^j = \mathbf{r}_0 + \sum_{i=1}^{j} d^{i-1} \prod_{n=j+1-i}^{j} (1 - \mathbf{c}^n) \tag{14}$$

and the residual flows

$$\bar{\mathbf{r}}^j = \mathbf{r}_0 d^{i-1} \prod_{n=1}^{j} (1 - \mathbf{c}^n). \tag{15}$$

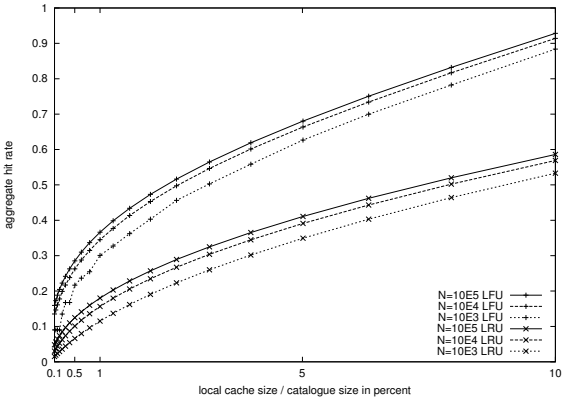

Figure 2: Aggregate hit rate as function of cache size for LFU and LRU.

7. RESULTS AND OBSERVATIONS

Next we solve the equations for a tree of degree $d = 10$ and a three levels for various catalog sizes N and cache sizes C. Requests will be Zipf distributed using $\alpha = 0.7$.

In Figure 2 we plot the aggregate hit rate for the whole cache hierarchy for three different catalogue sizes $N = 10^3, 10^4$ and 10^5. In each setup we vary the cache size from 0.1 to 10 percent of the catalogue size. We note that the hit rate of LFU is almost twice that of LRU and that the benefit of increasing cache size grows faster. In both cases we note a diminishing return of cache size to hit rate as the catalog grows large. Cache size has to grow super linear in order to achieve a linear gain in hit rate. This is also consistent with observations made by others e.g [1].

Figure 3 shows the hit rate of individual levels for a very deep tree network of 10 levels and a cache size of 0.1% ($C/N = 0.001$). We note that the first level (leaf) cache is the most important one. Caches at higher levels can still contribute to the aggregate cache effect, but only to a much smaller extent (this is not additive). Thus, when designing future in-network caching architectures, one should consider using either larger caches at higher levels or creating groups of collaborating creates to form larger virtual caches (paying the internal communication costs but increasing content availability). This holds for the whole parameter range but the relative performance of level 1 decreases with catalog size. This is intuitive since, for a very large N, each level in the cache will store very popular items and will be of similar importance to performance.

Next we examine what items will likely be cached by the two eviction policies. A rank plot of popular items and the probability of finding them cached is given in Figure 4. Here we see the difference between LRU and LFU clearly. LFU caches *only* the most popular items (head of the distribution) whereas LRU will cache all items with a non zero probability. An interesting effect of this is that if caches did collaborate, LFU would still have to request many items from outside of the tree whereas LRU could almost surely find the item somewhere locally. Thus, collaborative caching can also be used to improve availability in the case of network problems or failures of external links.

7.1 A realistic example

Next we will apply the analytic model on a fictitious ac-

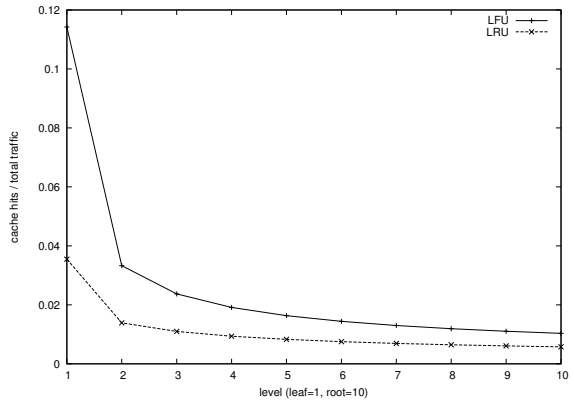

Figure 3: Hit rate for various levels in the cache hierarchy. The leaf level is denoted 1.

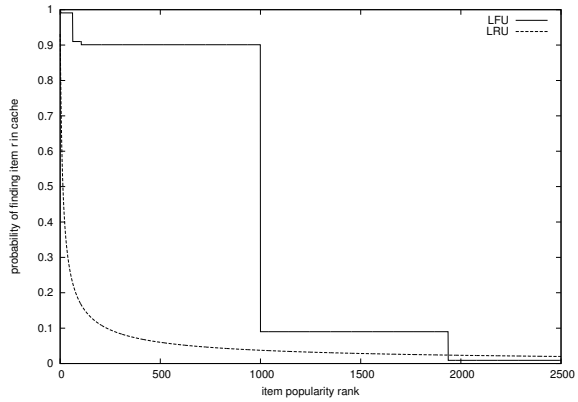

Figure 4: Comparison between which items are cached by LRU and LFU by rank.

cess aggregation network. The network will consist of three levels of routers each with a fan-out (degree) of 10. The topmost (root) router is directly connected to the operators' metropolitan network and the remaining routers form a tree below the root. A possible interpretation of this sample network is one hundred Digital Subscriber Line Access Multiplexer (DSLAM) pools that in turn are connected to ten Broadband Remote Access Servers (BRAS) that are conneced to the root router.

Each router is equipped with 360 Gbytes of flash memory to be used for in-network caching. We pick 360 Gbytes because there is a standard PCI-Express card that currently costs 700 EUR with this amount of memory. The card provides 540 Mbytes/s of sustained read capacity. For simplicity all routers are equipped with the same amount of cache memory.

Duration	8 days		
Requests	29,707	5,199 clients	5.7 reqs/client
Catalog	3,518 items	2,473 GB	703 MB/item

The average content size is 703 Mbytes which with a 360 Gbyte cache corresponds to a cache capacity of 512 items. In this example we want to calculate lower bounds on cache hit rates and will consequently use the LRU eviction policy, which in our model does not benefit from correlations in the requests stream.

With these parameters as input to the analytic model we expect a hit rate of 45% at the routers closest to the terminals. At the next level hit rate will be 26% and at the root 22%. Taken together this represents an aggregate cache hit rate of 68%, i.e only 32% of the VoD traffic is expected to be visible outside the tree.

The results (hit rates) of the calculations are summarized for three different xache sizes below.

Level	256GB	360GB	512GB
Root router	0.184	0.218	0.260
Middle (BRAS)	0.223	0.264	0.315
Leaf (DSLAM)	0.376	0.448	0.523
Total	0.604	0.683	0.758

An important question to consider is if flash memory will provide enough capacity to satisfy all read traffic.

To this end, we first note that from the trace data we can deduce an average VoD rate of 30 Mbyte/s of which 45% or 13.5 Mbyte/s should be read from our cache. In our experience, peak VoD load can be as much as 5 times the average load. In this case, there is plenty of spare capacity to satisfy the demand.

Similarly, at the root level we must read data from the cache at an average rate of 265 Mbyte/s. Although this value may be possible, the same does not apply to peak loads as much as 5 times the average load, in which case we need to use flash memory with higher read performance. Such memory exists and considering that this router should be a high end one, the additional cost can likely be motivated.

In June 2011 Cisco reported [3] *Internet video is now 40 percent of consumer Internet traffic, and will reach 62 percent by the end of 2015, not including the amount of video exchanged through P2P file sharing. The sum of all forms of video (TV, video on demand [VoD], Internet, and P2P) will continue to be approximately 90 percent of global consumer traffic by 2015.*

If we assume Internet video accesses to be similar in character to our VoD data, it should be within reach to save some 27% of all consumer Internet traffic and this should extend to 42% savings by the end of 2015. Also, remember that our model is conservative and does not benefit from request correlations. Bandwidth savings should thus be expected to be larger than this.

8. CONCLUSION AND FUTURE WORK

In this paper we have developed an analytical model of an access aggregation network with in-network caches and we have given exact analytical expressions for the performance of LRU and LFU caches. Using these results we were able to derive probabilities that given items reside in particular caches, determine cache efficiencies in terms of hit rates and network loads. The model enabled us to study systems scaling to millions of data items and thousands of nodes, something that is often impossible using simulations.

In the context of Zipf distributed accesses we have observed that LRU is better at caching items across the catalog making it a better candidate for collaborative caching and improvements in availability. On the other hand LFU can achieve better cache hit rates.

For practical purposes, one needs to strike a balance between frequency and recency in the eviction policies to meet

expectations of availability and cache hit rate. The analytical model can aid in striking this balance.

Finally, in a realistic example we showed that it should be possible to save 68% of VoD traffic using inexpensive caches. As of today this corresponds to approximately 27% of all consumer Internet traffic [3].

Future directions of this work will be to analyze other eviction policies using the same basic, model and to extend it to study the impact of intra network cache collaboration. We intend to extend the analysis to more realistic request patterns containing temporal correlations among requests as well as time varying popularity distributions.

9. REFERENCES

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134. IEEE, 1999.

[2] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *Selected Areas in Communications, IEEE Journal on*, 20(7):1305–1314, 2002.

[3] Cisco visual networking index: Forecast and methodology, 2010-2015, June 2011.

[4] A. Dan and D. Towsley. An approximate analysis of the lru and fifo buffer replacement schemes. *ACM SIGMETRICS Performance Evaluation Review*, 18(1):143–152, 1990.

[5] G. Dán and N. Carlsson. Power-law revisited: large scale measurement study of p2p content popularity. In *Proceedings of the 9th international conference on Peer-to-peer systems*, pages 12–12. USENIX Association, 2010.

[6] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for lru cache performance. *Arxiv preprint arXiv:1202.3974*, 2012.

[7] C. Fricker, P. Robert, J. Roberts, and N. Sbihi. Impact of traffic mix on caching performance in a content-centric network. *Arxiv preprint arXiv:1202.0108*, 2012.

[8] G. Haßlinger and F. Hartleb. Content delivery and caching from a network provider's perspective. *Computer Networks*, 55(18):3991–4006, 2011.

[9] R. Karedla, J.S. Love, and B.G. Wherry. Caching strategies to improve disk system performance. *Computer*, 27(3):38–46, 1994.

[10] L. Kleinrock. *Queueing Systems: Volume 2: Computer Applications*, volume 82. John Wiley & Sons, 1976.

[11] J. Reineke and D. Grund. Relative competitiveness of cache replacement policies. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):431–432, 2008.

[12] J. Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, 1999.

[13] D.A.B. Weikle, S.A. McKee, and W.A. Wulf. Caches as filters: A new approach to cache analysis. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1998. Proceedings. Sixth International Symposium on*, pages 2–12. IEEE, 1998.