Overcoming the Limitations of Cache Network Simulations

Roman Lutz
Advisors: Don Towsley, Ramesh Sitaraman, Mostafa Dehghan

***Abstract*—There are numerous experimental studies on caching in Information Centric Networks (ICN), but they all have major limitations including the use of only very few cache policies, often only one network cache strategy, arbitrary network topologies as well as unrealistic workloads. The goal of this work is to overcome these limitations and provide the first comprehensive experimental study as well as the means for other researchers to replicate or extend the results.**

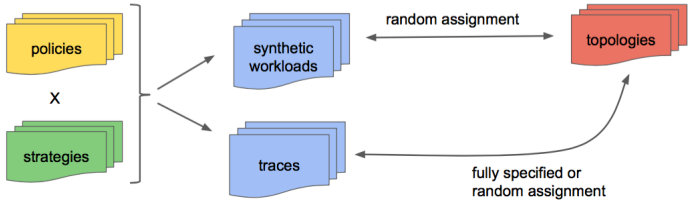


Fig. 1. An overview of our system.

I. Introduction

Communication in the Internet has evolved significantly from when it was invented with the design goals as described by Clark [9]. In addition to host-to-host message exchanges, techniques like multicast and optimizations for content delivery such as Content Delivery Networks (CDN) were added. The necessity for such workarounds arose from the dramatic increase in popularity and the need to support new applications. While industry is more focused on improving the existing Internet, researchers have been looking into alternatives for the TCP/IP architecture for a while now.

One of the explored directions has been Content-Centric Networks (CCN) [14]. The key idea is to make communication based on requests called interests that result in responses containing the requested content. The contents all have globally unique hierarchical names like, for example, `umass/cics/people/romanlutz/website`. Hosts can then simply send interests to the closest router without any knowledge of who has the content or where the content is stored. The CCN routers resolve interests based on their Forward Information Base (FIB) that essentially works as a routing table. As an optimization, routers can cache contents in order to possibly serve interests directly. These caching capabilities have sparked a renewed interest in caching in the research community.

Importantly, it makes sense to re-evaluate existing caching policies for a number of reasons. First of all, the distribution of frequencies for contents in CCNs is different from the distributions we can observe in memory access traces such as the ones used by Megiddo and Modha [25]. Secondly, it is important to distinguish between chunk level and object level caching. While CDNs and other future Internet architectures may cache whole objects, CCN operates on chunks that together make up objects. Depending on the intended application, it may be useful to simulate with either granularity. Finally, cache networks include multiple nodes and therefore caches which means we have to simulate the entire network if we want to find the best policies for the network as a whole. Moreover, we might employ more complex cache network strategies to decide when to use our caching policy at some nodes. All in all, there is a need for a comprehensive experimental evaluation to determine the most effective combinations of cache policies and network cache strategies for different workloads and topologies.

However, this is not a new realization. Previous works such as [16], [31], [34], [36], [37] have identified the same need, but all of them suffer from one or more of the following limitations:

- They do not use real network traces and instead only generate workloads according to a (Mandelbrot-)Zipf distribution. These are stationary and fail to capture changes in popularity.
- They assume very simple topologies such as a single path between receiver and content source or a tree.
- They use only very simple cache policies such as FIFO, Random, LRU.
- They use only very simple network cache strategies such as Leave Copy Everywhere (LCE).

Any of these limitations prevent us from drawing significant conclusions. The contributions of this work are therefore twofold:

- We present the results with respect to the overall cache hit ratio of simulations with a large variety of different configurations.
- We provide the enhanced simulator [23] as an open-source tool for other researchers to re-use and possibly extend our results.

Figure 1 shows an overview of the functionality. By pairing up caching policies with network cache strategies we get a number of test configurations. These are applied to every selected scenario which can be either based on synthetic workloads or traces. Finally, the traces can either be very specific and assign the requests to nodes in a topology or - in case the traces do not provide all this information - the

requests are randomly assigned to designated receiver nodes, which is the default for synthetic workloads. The simulator provides all these options in a concise manner and includes functionality to display, analyze and visualize the results.

The structure of this report is as follows: Section VIII summarizes related work on cache networks, modeling and simulations, cache policies, network cache strategies as well as network measurements.

II. Related Work

There have been a few attempts at simulating cache networks before which we summarize below:

Rossi and Rossini [34] conducted a study similar to ours on cache performance for Content-centric Networks. However, they did not use real traces, very simple cache policies and fewer options for the strategy layer. While they argue that anything but simple cache policies is infeasible in CCNs due to the line-speed constraint, we would at least like to see how much is lost in terms of cache performance compared to more complex policies and whether it is worth trying to build more efficient hardware that allows for complex policies.

Rossini and Rossi [36] used both a 4x4-torus and the GEANT topology with YouTube-like file sizes and amounts as well as popularities following a Zipf distribution ($\alpha \in [0.5, 2.5]$). In their conclusion, however, they mention that a Mandelbrot-Zipf distribution might be a better fit. The only caching policies were LRU and a random policy. As an interesting extension, they proposed that multiple repositories per content may exist. Depending on the forwarding policy a node may ask all repositories or only the closest for the content. Additionally, their model includes locality restrictions which limit a content's validity to a certain area. They argue that a random policy can be used with minimal loss instead of LRU, where the loss ranged from about 2-5% in cache hit rate. From their observations they conclude a minimal influence of the topology as opposed to the importance of the strategy layer.

Rossini and Rossi [37] used a variety of real-world topologies to evaluate multi-path interest forwarding strategies. Their workload model is a Mandelbrot-Zipf distribution with $\alpha \in \{1, 1.15\}, q \in \{0, 5\}$. As before, they used only simple cache policies (LRU, FIFO, BIAS, UNIF) and network cache strategies which they call decision policies (LCE, LCD, FIX(P)), while again referring to the line-speed constraint of CCN.

Kim and Yeom [16] used a workload model with K classes of objects where the elements within a class have equal occurrence probabilities and the probabilities of the classes follow a Zipfian distribution with $\alpha \in [1, 2]$. They selected one arbitrary network topology with 14 nodes with caching capabilities. They varied the cache sizes from 32 to 1024 elements. Apart from the cache hit ratio they also investigate average response times, the average hop count and the maximum link load. The main part of their paper are two optimization algorithms that operate on a single-path and network-wide basis. The comparison shows that they outperform LRU.

Psaras, Clegg, Landa, Chai and Pavlou [31] both analytically modeled caching in CCNs using Markov Chains and simulated with ns-2. Their experiments are restricted to tree topologies. The results indicate that popular contents are most likely cached in leaf nodes and that sizing the caches appropriately is crucial for good results.

Internet measurements provide valuable data for simulations. Saleh and Hefeeda [41] measured traffic over nine months from more than 100 ASes and fitted Mandelbrot-Zipf models to the data. They found that the Mandelbrot-Zipf distribution captures the frequencies much better than just the Zipf distribution. The parameters with the best fit for each AS were (with a few exceptions) with $0.5 \leq \alpha \leq 0.8$ and $0 \leq q \leq 20$. Our own parameters were chosen with regards to these values.

III. Icarus Simulator

The icarus simulator [40] provides the foundation that our simulator is built on. We have enhanced it in multiple ways including new cache policies, workloads such as the deterministic trace-driven workload, new methods to select the cache size (absolute, trace-dependent), additional collectors for a variety of statistics, optimized I/O for large-scale results files, libraries for trace analytics as well as the visualization of results and trace properties. The enhanced simulator is available as an open-source tool [23].

IV. Experiments

The experiments were executed with our enhanced version of the Icarus simulator. Whenever synthetic workloads were used, we averaged the results over three experiments. The cache hit rates were measured as

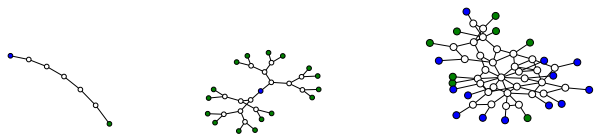
$$\sum_{i=1}^{n} h_i \Big/ \sum_{i=1}^{n} r_i$$

where r_i is the number of requests arriving at node i and h_i the number of hits at node i. Let us denote the number of checks resulting in hit or miss at all nodes combined as $r = \sum_{i=1}^{n} r_i$. Note that r may vary from scenario to scenario. For example, a cache hit in a node close to the receiver prevents the other caches on the path to the source from having to check the cache. The number of caches varies depending on the topology. The overall parameter selections are summarized in Table I.

Parameter	Value(s)
cache size	10^3 elements for every node
number of contents	10^5
number of requests	10^7
workload model	Mandelbrot-Zipf(q, α)
q	$0, 5, 50$
α	$0.5, 0.75, 1$
network cache strategy	LCE, LCD, CL4M, ProbCache+, RandomChoice
caching policy	LRU, 2-LRU, ARC, DSCA, 2-DSCA, DSCAAWS
topology	Path, Tree, GEANT

TABLE I
THE PARAMETERS FOR OUR SYNTHETIC EXPERIMENTS.

The trace-driven simulations use just under 20 million requests while the cache size, strategies and policies are the same as with the synthetic experiments. The traces were only used with the GEANT topology.



(a) Path (b) Tree (c) GEANT

Fig. 2. The topologies used for our experiments.

A. Topology

For the experiments we used three topologies: Path, Tree and GEANT (Figure 2). Path consists of one receiver, one source and five caches in between. Tree has a branching factor of 2 with a depth of 4. That means there is one source, 14 caches and 16 receivers. GEANT is similar to other real-world topologies and a popular topology to use for cache network experiments. The topology already defines which nodes are receivers, content sources and nodes with caching capabilities. In total, it includes 8 receivers, 13 sources, 19 caches as well as 13 routers without caches.

B. Content Placement

For every content there is exactly one content source. The assignment of contents to sources is random and can be weighted with pre-defined weights for sources, but we chose uniform assignment for the sake of simplicity.

C. Strategy Layer

While Icarus offers a number of network cache strategies we selected only five of them:

- Leave Copy Everywhere (LCE) uses the cache policy at every node to decide whether a content shall be cached.
- Leave Copy Down (LCD) [19] moves the the content one hop closer to the receiver while also keeping it at the original location (serving cache or server).
- Cache Less For More (CL4M) [7] uses the concept of betweenness centrality

$$C_B(v) = \sum_{i \neq v \neq j \in V} \frac{\sigma_{i,j}(v)}{\sigma_{i,j}}$$

  where $\sigma_{i,j}$ is the number of content paths between nodes i and j. The node v with the largest $C_B(v)$ on the way to the receiver determines whether or not to cache the object with its caching policy.
- ProbCache+ [30] tries to estimate the caching capabilities of the path weighted by the distance to the receiver. Nodes with a distance x (in hops) from the content source use their cache policy with a probability of

$$\text{ProbCache+}(x) = \frac{\sum_{i=1}^{c-(x-1)} N_i}{T_{tw} N_x} \times \left(\frac{x}{c}\right)^c$$

  c is the total length of the path from receiver to source, N_i is the cache size of the i-th node from the receiver and T_{tw} is the target time window, an adjustable parameter. The first part is the estimated path caching capability, the second part is the cache weight that increases the closer a cache is to the receiver.
- RandomChoice randomly decides to use the cache policy of one node on the path from serving cache/server to receiver.

D. Caching Policy

For each of our experiments all caches were assigned the same policy. We denote the cache size by n and selected the following policies:

- Least Recently Used (LRU) keeps the n last requested contents in the cache.
- 2-LRU uses two levels of LRU caches, the first one containing only the objects' metadata. The first time an element is observed its metadata is stored in the first level. If it is requested again while its metadata is saved, the actual object is cached on the second level. This way contents with only a single occurrence are not cached.
- Adaptive Replacement Cache (ARC) [25] is similar to 2-LRU, but uses both a real cache and a shadow cache with metadata on both levels and adaptively sizes the caches according to the incoming requests.
- Data Stream Caching Algorithm (DSCA) [33] splits the cache into two parts: an LRU part and a cache with the top-k elements. DSCA uses a pre-defined window size, and after every window the top-k elements are updated based on a Space Saving [26] table. Space Saving provides guarantees on the top-k elements, but the number of actually guaranteed contents varies. LRU is simply used for the remaining slots.
- 2-DSCA combines ideas from 2-LRU and DSCA. Elements are not automatically cached in the LRU list, but only upon the second occurrence, which is when they are already in the Space Saving table. This is a solid approximation of 2-LRU, while the top-k portion remains the same as in DSCA.
- DSCA with adaptive window size (DSCAAWS) does not use a fixed window size, but instead uses a hypothesis test to determine whether the current window should be extended or not. As before, whenever a window ends, the top-k are updated.

Specifically the last two policies are subject to current research. The simulator offers other options as well, but they have shown to perform worse in prior experiments or are currently in an experimental state.

E. Workload

We call the model or trace that describes the stream of generated requests the workload.

1) Synthetic Workload: The original version of Icarus allowed for a Zipf distribution only. Since other works suggested a generalization, the Mandelbrot-Zipf distribution, we added it to the simulator. With parameters α and q, the elements are ranked $1, \ldots, N$. The probability for a request to be for the

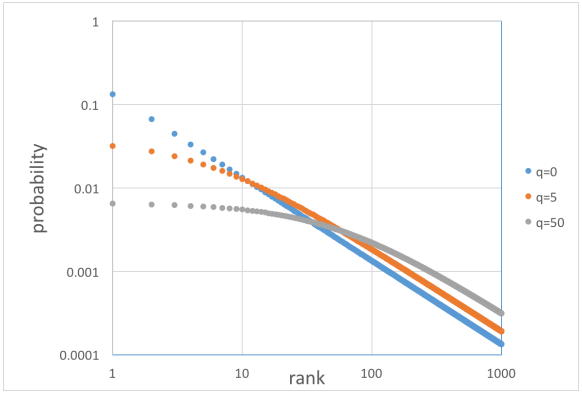


Fig. 3. The probabilities of the Mandelbrot-Zipf distribution for various values of q and $\alpha = 1$.

k-th ranked element is then

$$\frac{1}{(k+q)^{\alpha} \sum_{i=1}^{N} \frac{1}{(i+q)^{\alpha}}}$$

For a normal Zipf distribution, q is set to 0. Figure 3 illustrates the effect of the parameter q introduced in the Mandelbrot-Zipf distribution. In a log-log plot, the probabilities of the Zipf distribution are a straight line. With $q > 0$ the probabilities of the higher ranked elements decrease.

The α parameter indicates how much the performance gain through caching can be. The higher α, the more skewed is the distribution and the easier it is to identify the most frequently occurring elements. The closer α gets to 0, the more it approximates a uniform distribution, thus reducing the overall gain of caching.

Parameter values for the Mandelbrot-Zipf distribution were chosen similar to Rossi and Rossini's experiments [34] as well as Saleh and Hefeeda's measurement results [41].

In terms of the request generation itself, we went with a total of 10,000,000 requests for 100,000 unique objects. The measurements do not include a warmup phase where hits and misses are not counted that is equal to the number of contents.

2) Trace-driven Workload: The simulator offers two ways of using traces depending on how much information the traces provide. If the topology of the trace is known it can be used to assign the requests to the specified receivers. Otherwise, the simulator has an option of randomly assigning requests to receivers.

The trace we used is from the Brazilian ISP and media network Globo. The requests were recorded over a whole month and include both full files as well as chunks of files depending on the requested media type. Due to the sheer size of the trace (198 million requests), we used only a small part (20 million) for our simulations.

V. Results

The heatmaps in the Appendix illustrate which combinations of caching policies and network cache strategies performed well together. We discuss some of the observations in the following.

A. Synthetic workloads

Figure 5 shows the results of various Mandelbrot-Zipf workloads on a path topology with 7 nodes. In all but one case, the combination of ARC and ProbCache+ had the highest cache hit rate, sometimes by multiple percentage points. 2-DSCA with a window size 64 times larger than the number of monitored elements (2000) combined with LCD had a minimally higher rate for $q = 50$ and $\alpha = 1$. Overall, ProbCache+ and LCD deliver solid results with any policy. Other consistently top-ranked combinations are 2-LRU with RandomChoice, ARC and RandomChoice, 2-DSCA and LCD.

The results for a tree topology with branching factor 2 and height of 4 are presented in Figure 6. ARC and ProbCache+ outperform every other combination in all scenarios. Other pairs worth mentioning are 2-DSCA with either LCD or CL4M.

Figure 7 shows the results for GEANT which are very different from the previous two topologies. Overall, there are only a few combinations that actually performed well on all synthetic workloads, namely ARC, 2-LRU and 2DSCA with RandomChoice and LCD. ARC and ProbCache+ also provided good results and as before, ARC is by far the best policy for ProbCache+. The CL4M strategy consistently performs badly with every policy which was not the case with the tree and path topologies.

Overall, RandomChoice and LCE only do well with policies that have some degree of scan-resistance, that is ARC, 2-LRU and 2-DSCA. CL4M works best with DSCA or 2-DSCA, but never shows promising results. ProbCache+ and LCD seem to work well regardless of which policy is used.

B. Trace-driven workloads

Figure 4 shows the results from the experiments with a part of the Globo trace. Overall, the relative differences between the rates are lower compared to the synthetic workloads. The two main takeaways are that DSCAAWS performs badly regardless of the strategy and the same applies to ProbCache+ no matter which policy it is used with. LCD achieves remarkable results with every policy except DSCAAWS and delivers the overall best results together with ARC.

C. Overall observations

Contrary to the suggestions of previous studies we find that the topology has a substantial influence on the results. A good example is CL4M with synthetic workloads which had good results on path and tree topologies, but performed consistently badly with on GEANT.

As expected, different workloads make a huge difference. When comparing the Mandelbrot-Zipf workload with the Globo trace on the same topology (GEANT), CL4M went from the overall worst strategy to a solid overall cache hit rate. LCE showed similarly extreme changes the opposite way. Only LCD and RandomChoice performed very well on both workloads.

However, RandomChoice's success depends heavily on the caching policy. The results of the path and especially the tree topologies illustrate the vast difference.

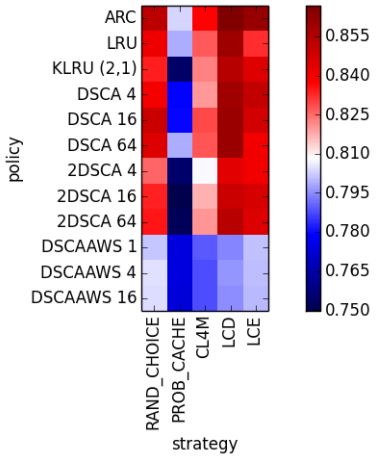


Fig. 4. The cache hit rates for the trace-driven workload based on the Globo trace. The underlying topology used was GEANT.

We tried to identify a robust strategy - policy combination that consistently does well, or at least does not have very negative outliers. The candidates we came up with based on the heatmaps are: RandomChoice or LCD with ARC, 2-LRU or 2-DSCA; ARC with LCE; ARC with ProbCache+.

Finally, one of the stated goals of this work is to estimate the gain from using more complex policies instead of LRU. For this we determined the difference between the best LRU result and the overall best result for every workload and topology. In absolute percentage points LRU performs between 3 and 8 points worse with synthetic workloads. Since the cache hit ratios vary a lot, it makes sense to consider the relative percentage difference. There, the best LRU combinations come up 5.5% to 31% short of the best values. In case of the Globo trace the gain is less than 1%.

VI. Future Work

We see a number of ways to extend the existing work:

- Different workloads models may result in totally different results. It would be interesting to compare the existing Mandelbrot-Zipf model with others, especially if they incorporate changes in content popularity. There is also the possibility that chunks of objects follow a different distribution than the whole objects. The Mandelbrot-Zipf distribution may not be adequate anymore then.
- In order to validate the results of our traces we would like to use other traces as well.
- It is already possible to use fully specified traces that map requests explicitly to receivers, but we do not currently have such rich traces.
- We would like to investigate the possibility of using different cache policies at different nodes. For example, it may be beneficial to use some policies closer to the user than further away and viceversa.
- We are already capturing the individual cache hit rates of all nodes, but do not use the data for the evaluation. This might give us an indication on whether different policies in a network make sense.
- Overall it would be great to have a larger number of cache policies. Interesting algorithms include Clock with Adaptive Replacement (CAR) [3] and LRFU [20].
- The same applies to the network cache strategies. Interesting new strategies could be Least Benefit [52] or Move Copy Down (MCD). Hashrouting [39] is another promising option, but the tradeoff between path stretch and cache hit rate needs to be accounted for.
- In addition to the cache hit rates, other metrics could be used such as the average response times, the average hop count and the maximum link load. However, it is important to use a realistic configuration for link delays and request generation. The traces often do not provide times.
- Since other papers often mention the line speed requirement it is necessary to provide a study on the average computing time a policy takes. For instance, our simulations indicate that LRU, 2-LRU and ARC are significantly faster (factor $5-30$) than the DSCA variants. This could prohibit DSCA from ever getting used in cache networks and should be factored into the evaluation.

VII. Conclusion

Previous studies on caching in cache networks don't provide comprehensive evaluations. By extending the Icarus simulator we provide the first tool capable of extensive experiments with synthetic workloads and traces that purely aims at capturing cache hit rates.

Based on the first experiments with the simulator we already deliver a few interesting insights. Firstly, we reject the idea that the topology has barely any influence on the results. Secondly, we are able to confirm the large impact of the workload by comparing synthetic and real-world workloads. Thirdly, we provide a number of combinations of caching policy and network cache strategy that have proven to consistently perform well. Lastly, we evaluated whether the added complexity of policies such as DSCA and ARC brings the desired increase in cache hit rate compared to LRU. While there is definitely a significant improvement with synthetic workloads, we are unable to confirm the same with the trace. More traces should allow for a better estimate.

We are looking forward to use this work ourselves and help others use it as the foundation for future work.

VIII. References

[1] Somaya Arianfar, Pekka Nikander, and Jörg Ott. On content-centric router design and implications. In *Proceedings of the Re-Architecting the Internet Workshop*, page 5. ACM, 2010.

[2] Somaya Arianfar, Pekka Nikander, and Jörg Ott. Packet-level caching for information-centric networking. In *ACM SIGCOMM, ReArch Workshop*, 2010.

[3] Sorav Bansal and Dharmendra S Modha. Car: Clock with adaptive replacement. In *FAST*, volume 4, pages 187–200, 2004.

[4] Mudashiru Busari and Carey Williamson. Simulation evaluation of a heterogeneous web proxy caching hierarchy. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, pages 379–388. IEEE, 2001.

[5] Igor Carvalho, Airton Ishimori, A Abel, et al. A multicriteria caching decision for information centric networks. In *2016 International Conference on Information Networking (ICOIN)*, pages 129–134. IEEE, 2016.

[6] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[7] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache less for more in information-centric networks (extended version). *Computer Communications*, 36(7):758–770, 2013.

[8] Hoon-gyu Choi, Jungmin Yoo, Taejoong Chung, Nakjung Choi, Taekyoung Kwon, and Yanghee Choi. Corc: coordinated routing and caching for named data networking. In *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems*, pages 161–172. ACM, 2014.

[9] David Clark. The design philosophy of the darpa internet protocols. *ACM SIGCOMM Computer Communication Review*, 18(4):106–114, 1988.

[10] Jie Dai, Zhan Hu, Bo Li, Jiangchuan Liu, and Baochun Li. Collaborative hierarchical caching with dynamic request routing for massive content distribution. In *INFOCOM, 2012 Proceedings IEEE*, pages 2444–2452. IEEE, 2012.

[11] Michele Garetto, Emilio Leonardi, and Stefano Traverso. Efficient analysis of caching strategies under dynamic content popularity. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2263–2271. IEEE, 2015.

[12] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM, 2007.

[13] Mohamed Hefeeda and Osama Saleh. Traffic modeling and proportional partial caching for peer-to-peer systems. *Networking, IEEE/ACM Transactions on*, 16(6):1447–1460, 2008.

[14] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[15] Wu Jinlong. Analysis of the performance of cachereplacement policies for avideo-on-demand system. 2013.

[16] Yusung Kim and Ikjun Yeom. Performance analysis of in-network caching for content-centric networking. *Computer Networks*, 57(13):2465–2482, 2013.

[17] Jim Kurose. Information-centric networking: The evolution from circuits to packets to content. *Computer Networks*, 66:112–120, 2014.

[18] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. The lcd interconnection of lru caches and its analysis. *Performance Evaluation*, 63(7):609–634, 2006.

[19] Nikolaos Laoutaris, Sofia Syntila, and Ioannis Stavrakakis. Meta algorithms for hierarchical web caches. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 445–452. IEEE, 2004.

[20] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H Noh, Sang Lyul Min, Yookun Cho, and Chong Sang Kim. Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE transactions on Computers*, (12):1352–1361, 2001.

[21] Junghwan Lee, Kyubo Lim, and Chuck Yoo. Cache replacement strategies for scalable video streaming in ccn. In *Communications (APCC), 2013 19th Asia-Pacific Conference on*, pages 184–189. IEEE, 2013.

[22] Zhe Li and Gwendal Simon. Cooperative caching in a content centric network for video stream delivery. *Journal of network and systems management*, 23(3):445–473, 2015.

[23] Roman Lutz. Icarus: enhanced version. https://github.com/romanlutz/icarus, 2016.

[24] Michele Mangili, Fabio Martignon, and Antonio Capone. Performance analysis of content-centric and content-delivery networks with evolving object popularity. *Computer Networks*, 2015.

[25] Nimrod Megiddo and Dharmendra S Modha. Arc: A self-tuning, low overhead replacement cache. In *FAST*, volume 3, pages 115–130, 2003.

[26] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Database Theory-ICDT 2005*, pages 398–412. Springer, 2005.

[27] Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, and Periklis Chatzimisios. Information-centric networking and multimedia services: present and future challenges. *Transactions on Emerging Telecommunications Technologies*, 25(4):392–406, 2014.

[28] Stefan Podlipnig and Laszlo Böszörmenyi. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398, 2003.

[29] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM, 2012.

[30] Ioannis Psaras, Wei Koong Chai, and George Pavlou. In-network cache management and resource allocation for information-centric networks. *Parallel and Distributed Systems, IEEE Transactions on*, 25(11):2920–2931, 2014.

[31] Ioannis Psaras, Richard G Clegg, Raul Landa, Wei Koong Chai, and George Pavlou. Modelling and evaluation of ccn-caching trees. In *NETWORKING 2011*, pages 78–91. Springer, 2011.

[32] Haiyang Qian, Ravishankar Ravindran, Guo-Qiang Wang, and Deep Medhi. Probability-based adaptive forwarding strategy in named data networking. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 1094–1101. IEEE, 2013.

[33] Antonio A Rocha, Mostafa Dehghan, Theodoros Salonidis, Ting He, and Don Towsley. Dsca: A data stream caching algorithm. In *ACM SIGCOMM CoNEXT CCDWN Workshop*, 2015.

[34] Dario Rossi and Giuseppe Rossini. Caching performance of content centric networks under multi-path routing (and more). *Relatório técnico, Telecom ParisTech*, 2011.

[35] Dario Rossi, Giuseppe Rossini, et al. On sizing ccn content stores by exploiting topological information. In *INFOCOM Workshops*, pages 280–285, 2012.

[36] Giuseppe Rossini and Dario Rossi. A dive into the caching performance of content centric networking. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on*, pages 105–109. IEEE, 2012.

[37] Giuseppe Rossini and Dario Rossi. Evaluating ccn multi-path interest forwarding strategies. *Computer Communications*, 36(7):771–778, 2013.

[38] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Understanding sharded caching systems.

[39] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Hash-routing schemes for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 27–32. ACM, 2013.

[40] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Icarus: a caching simulator for information centric networking (icn). In *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, pages 66–75. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[41] Osama Saleh and Mohamed Hefeeda. Modeling and caching of peer-to-peer traffic. In *Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on*, pages 249–258. IEEE, 2006.

[42] Vasilis Sourlas, Paris Flegkas, Lazaros Gkatzikis, and Leandros Tassiulas. Autonomic cache management in information-centric networks. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 121–129. IEEE, 2012.

[43] Vasilis Sourlas, Lazaros Gkatzikis, Paris Flegkas, and Leandros Tassiulas. Distributed cache management in information-centric networks. *Network and Service Management, IEEE Transactions on*, 10(3):286–299, 2013.

[44] Vasilis Sourlas, Leandros Tassiulas, Ioannis Psaras, and George Pavlou. Information resilience through user-assisted caching in disruptive content-centric networks. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9. IEEE, 2015.

[45] Kalika Suksomboon, Saran Tarnoi, Yusheng Ji, Michihiro Koibuchi, Kenji Fukuda, Shigeto Abe, Nakamura Motonori, Masaki Aoki, Shigeo Urushidani, and Shigeru Yamada. Popcache: cache more or less based on content popularity for information-centric networking. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 236–243. IEEE, 2013.

[46] Wenting Tang, Yun Fu, Ludmila Cherkasova, and Amin Vahdat. Modeling and generating realistic streaming media server workloads. *Computer Networks*, 51(1):336–356, 2007.

[47] Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review*, 43(5):5–12, 2013.

[48] Athanasios V Vasilakos, Zhe Li, Gwendal Simon, and Wei You. Information centric network: Research challenges and opportunities. *Journal of Network and Computer Applications*, 52:1–10, 2015.

[49] Jason Min Wang, Jun Zhang, and Brahim Bensaou. Intra-as cooperative caching for content-centric networks. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 61–66. ACM, 2013.

[50] Sen Wang, Jun Bi, Zhaogeng Li, Xu Yang, and Jianping Wu. Caching in information-centric networking. In *AsiaFI Future Internet Technology Workshop*, 2011.

[51] Sen Wang, Jun Bi, and Jianping Wu. Collaborative caching based on hash-routing for information-centric networking. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 535–536. ACM, 2013.

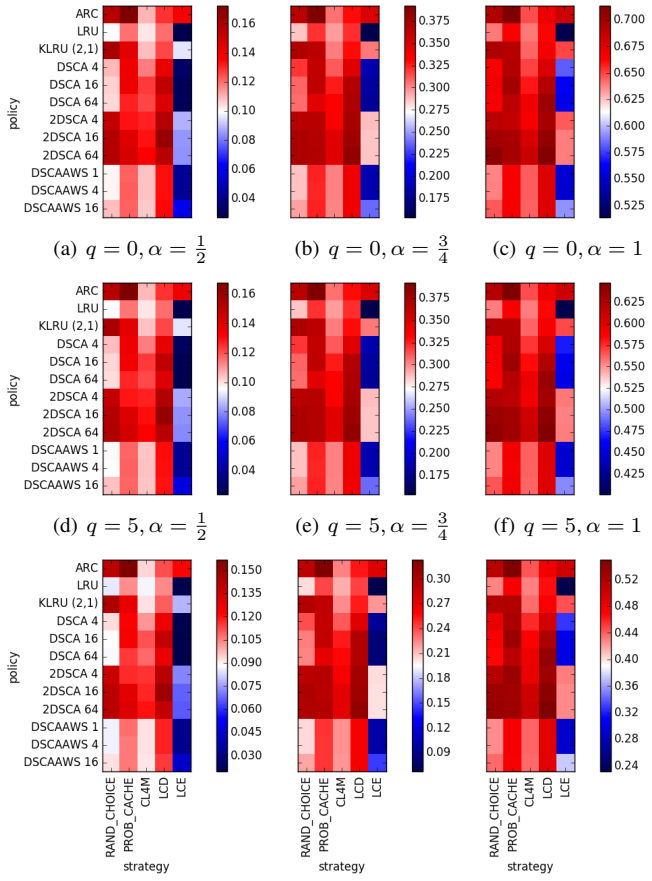
[52] Sen Wang, Jun Bi, Jianping Wu, Zhaogeng Li, Wei Zhang, and Xu Yang. Could in-network caching benefit information-centric networking? In *Proceedings of the 7th Asian Internet Engineering Conference*, pages 112–115. ACM, 2011.

[53] Yonggong Wang, Zhenyu Li, Gareth Tyson, Steve Uhlig, and Gaogang Xie. Optimal cache allocation for content-centric networking. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–10. IEEE, 2013.

[54] Yonggong Wang, Zhenyu Li, Gareth Tyson, Steve Uhlig, and Gaogang Xie. Design and evaluation of the optimal cache allocation for content-centric networking. *Computers, IEEE Transactions on*, 65(1):95–107, 2016.

[55] Guoqiang Zhang, Yang Li, and Tao Lin. Caching in information centric networking: a survey. *Computer Networks*, 57(16):3128–3141, 2013.

IX. Appendix

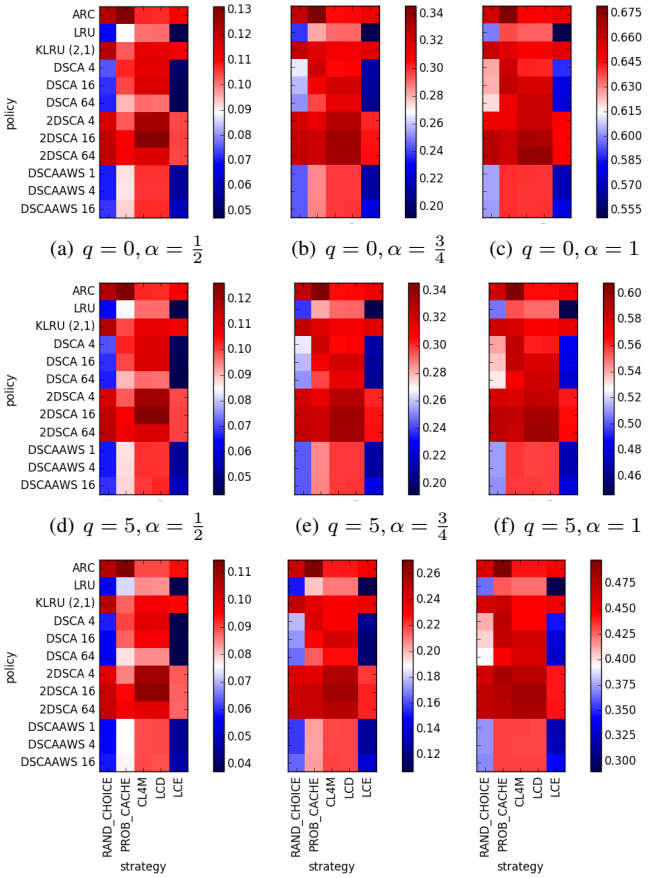


(a) $q = 0, \alpha = \frac{1}{2}$ (b) $q = 0, \alpha = \frac{3}{4}$ (c) $q = 0, \alpha = 1$

(d) $q = 5, \alpha = \frac{1}{2}$ (e) $q = 5, \alpha = \frac{3}{4}$ (f) $q = 5, \alpha = 1$

(g) $q = 50, \alpha = \frac{1}{2}$ (h) $q = 50, \alpha = \frac{3}{4}$ (i) $q = 50, \alpha = 1$

Fig. 5. The cache hit rates for a Path topology with 7 nodes and Mandelbrot-Zipf workload with a variety of values for q and α.

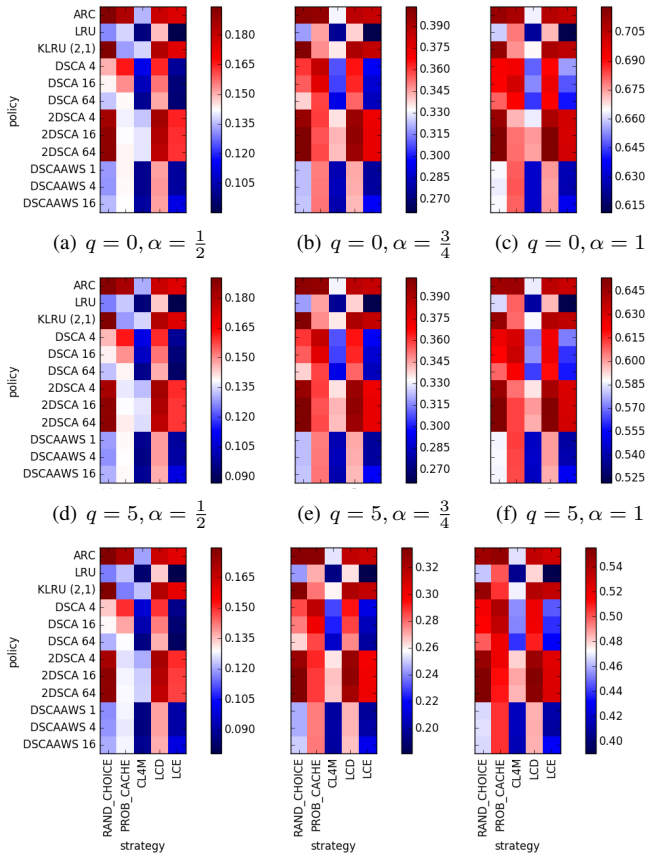


(a) $q = 0, \alpha = \frac{1}{2}$ (b) $q = 0, \alpha = \frac{3}{4}$ (c) $q = 0, \alpha = 1$

(d) $q = 5, \alpha = \frac{1}{2}$ (e) $q = 5, \alpha = \frac{3}{4}$ (f) $q = 5, \alpha = 1$

(g) $q = 50, \alpha = \frac{1}{2}$ (h) $q = 50, \alpha = \frac{3}{4}$ (i) $q = 50, \alpha = 1$

Fig. 6. The cache hit rates for a Tree topology with branching factor 2, height 4 and Mandelbrot-Zipf workload with a variety of values for q and α.



(a) $q = 0, \alpha = \frac{1}{2}$ (b) $q = 0, \alpha = \frac{3}{4}$ (c) $q = 0, \alpha = 1$

(d) $q = 5, \alpha = \frac{1}{2}$ (e) $q = 5, \alpha = \frac{3}{4}$ (f) $q = 5, \alpha = 1$

(g) $q = 50, \alpha = \frac{1}{2}$ (h) $q = 50, \alpha = \frac{3}{4}$ (i) $q = 50, \alpha = 1$

Fig. 7. The cache hit rates for the GEANT topology and Mandelbrot-Zipf workload with a variety of values for q and α.