LAC: Introducing Latency-Aware Caching in Information-Centric Networks

Giovanna Carofiglio, Leonce Mekinda, Luca Muscariello

LAC: Introducing Latency-Aware Caching in Information-Centric Networks

Giovanna Carofiglio*, Leonce Mekinda†, Luca Muscariello†
* Cisco Systems, † Orange Labs Networks,
gcarofig@cisco.com, firstname.lastname@orange.com

***Abstract*—Latency-minimization is recognized as one of the pillars of 5G network architecture design. Information-Centric Networking (ICN) appears a promising candidate technology for building an agile communication model that reduces latency through in-network caching. However, no proposal has developed so far latency-aware cache management mechanisms for ICN. In the paper, we investigate the role of latency awareness on data delivery performance in ICN and introduce LAC, a new simple, yet very effective, Latency-Aware Cache management policy. The designed mechanism leverages in a distributed fashion local latency observations to decide whether to store an object in a network cache. The farther the object, latency-wise, the more favorable the caching decision. By means of simulations, show that LAC outperforms state of the art proposals and results in a reduction of the content mean delivery time and standard deviation by up to 50%, along with a very fast convergence to these figures.**

I. Introduction

Latency minimization, or building for virtual zero latency as commonly referred to, is one of the pillars of 5G network architecture design and is currently fostering important research work in this space. Inserting cache memories across the communication data path between different processing elements has been already demonstrated to be a reliable way of improving performance by caching - especially popular - content at network edge and so, reducing retrieval latency.

Besides other advantageous architectural choices, the introduction of in-network caching as a native building block of the network design makes Information Centric Networking (ICN) [5] a promising 5G network technology. In a nutshell, every ICN router potentially manages a cache of previously requested objects in order to improve object delivery by reducing retrieval path length for frequently requested content. In fact, if content is locally available in the cache, the router sends it back directly to the requester, otherwise it forwards the request (or Interest) for the object to the next hop according to name-based routing criteria. When the requested object comes back, it is stored in the local cache before sending it back to the requester. Given cache size limitations, a replacement policy is put in place to evict previously stored objects for accommodating the newly available ones. To this aim, various classical cache replacement policies, not specifically ICN-based exist: to cite a few, Least-Recently-Used (LRU), Least-Frequently-Used (LFU), First-In-First-Out (FIFO) and Random (RND) [3]. Within the panoply of cache management policies proposed in the literature, very few exploit object retrieval latency to orchestrate cache decisions. Some requires transport protocol modifications [13] or involve additional computational complexity [15] without significant caching performance increase.

Clearly, the constraints imposed by ICN in terms of high speed packet processing exclude every complex cache management policy. Therefore, we focus in this paper on a simple, hence feasible, cache management policy leveraging not only the objects replacement, but the cache insertion criterion, that we define based on monitored object latency.

The cache management mechanism we propose in this paper, LAC, lies upon the following principle: every time an object is received from the network, it is stored into the cache with a probability proportional to its recently observed retrieval latency. As such, it is an add-on laying on top of any cache replacement policy and feeding it at a regulated pace. In this way, LAC implicitly prioritizes long-to-retrieve objects, instead of caching every object regardless. The underlying tradeoff such caching mechanism tackles is between a limited cache size and delivery time minimization. As caching intrinsically aims to relieve the fallouts of network distance or traffic congestion, it must be aware of both delay factors to efficiently handle the cache size / delivery time tradeoff. Data retrieval latency is a simple, locally measurable and consistent metric for revealing either haul distance or traffic congestion.

II. Related Work

In the context of ICN research, previous work have considered the enhancement of cache mechanisms with the aim of reducing caching redundancy over a delivery path. We can distinguish two categories of related work: those leveraging content placement (e.g. [17], [8]) as opposed to those proposing caching mechanisms based on selective insertion/replacement in cache (e.g. [13], [1], [4], [9]). The first class of approaches has a limited applicability to controlled environments like a CDN (Content Delivery Network), where topology and content catalog are known a priori. Both [17] and [8] deal with video streaming in ICN and orchestrate caching and scheduling of requests to caches in order to create a cluster of caches with a certain number of guaranteed replicas ([8]). Unlike these approaches, our work belongs to the second class of caching solutions and aims at defining a decentralized caching solution that automatically adapts to changes in content popularity, network variations etc. by leveraging content insertion/replacement operations in cache. We share the same objective as [1], where authors propose

a congestion-aware caching mechanism for ICN based on estimation of local congestion, of popularity and of position with regard to the bottleneck. The congestion estimate in this work does not allow to differentiate content items in terms of latency like in our work. A similar consideration holds for other related approaches: the ProbCache work in [13], which utilizes the same cache probability for every content item at a given node and the cooperative caching mechanism in [4]- [9] exploiting overall popularity and distance-to-server. Clearly, the rationale is the same, but the distance-to-server metric does not reflects the differences in terms of latency, distance to bottleneck on a per-flow basis that our approach takes into account. Beyond ICN, caching literature is vast [12] and our review here does not attempt to be exhaustive, while rather to position our contribution with regard to closest classical caching approaches. Starobinski *et al.* [14] and later Jelenković *et al.* [6] describe a cache management mechanisms to optimize the storage of variable size documents. In their work, the whole Move-To-Front rule is *symmetric*, i.e. applied in both hit and miss events (as for LRU, LFU etc.) while our approach, instead, may be denoted as *asymmetric*, since it restricts the stochastic decision of MTF to cache miss events, leaving object replacement subject to deterministic LRU.

III. Problem formulation and design choices

The problem of improving end-user delivery performance can be formulated as the minimization of the overall average delivery time $\mathbb{E}[\mathrm{T}]$ for all users in the network and over all requested objects.

$$\begin{cases} \min \sum_{u\in\mathcal{U}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}_{u,k}} q_{k,u} w_{k,r,u} \mathbb{E}[\mathrm{T}_{k,r,u}] & (1) \\ \sum_{k} q_{k,u} = 1, \qquad \forall u & (2) \\ \sum_{r} w_{k,r,u} = 1, \qquad \forall k,u & (3) \\ 0 \leqslant q_{k,u} \leqslant 1 \qquad \forall k,u & (4) \\ 0 \leqslant w_{k,r,u} \leqslant 1 \qquad \forall k,r,u & (5) \end{cases}$$

where $q_{k,u}$ is the normalized request rate of object k from user u (namely, the popularity function at user u), and $w_{k,r,u}$ is the probability to download object k from route r and $\mathbb{E}[\mathrm{T}_{k,r,u}]$ is the average latency to retrieve object k on route r. The set of routes available at user u is identified by $\mathcal{R}_{u,k}$.

In this paper we look for a distributed algorithm that tries to minimize this objective by obtaining $w_{k,r,u}$ without any coordination among the nodes and no signaling. The optimal objective expressed in Eq.(1) can be heuristically generalized to every node n in the network by substituting $q_{k,u}$ with the local residual popularity $q_k(n)$ at node n and $\mathbb{E}[\mathrm{T}_{k,r,u}]$ with the local virtual residual round trip time for object k on route r, denoted by $\mathbb{E}[\mathrm{VRTT}_{k,r}(n)]$. Hence we set the probability to store an object k at a given node n, proportional to the popularity and latency locally observed at node n. It is left to future work to prove that this distributed heuristic is actually optimal. The intuition behind Eq.(1) is that user u downloads an object k from a remote path r inversely proportional to its popularity and retrieval latency. A globally optimal strategy performed in each node would heuristically prefer to locally cache popular content and with high retrieval latency.

In this paper we design a heuristic based on the aforementioned criterion. Thus, our general formulation of the probability to cache a requested object k on node n at time i,

$$p_{k,r,i}(n) \propto \min\left(\frac{(\mathrm{VRTT}_{k,r,i}(n))^{\beta}}{\left(f\left((\mathrm{VRTT}_{t}(n))_{t<i}\right)\right)^{\gamma}}, 1 \right) \qquad (6)$$

$(\mathrm{VRTT}_t(n))_{t<i}$ denotes all priorly encountered object retrieval latencies. Object retrieval latency includes processing, queuing, transmission and propagation delays. f is a real-valued function that might be, for example, a mean, the median or the maximum of the encountered latencies. It embodies the cache state in a single metric for confrontation with the new object to store. β and γ are intensity parameters. They aim at stressing the rejection of easy-to-retrieve objects. $\propto$ means "is proportional to". The object retrieval latency and the probability of caching it are, hereby, made proportional. Note that the caching decision may cumulatively depend on another fixed or dynamic factors (such as the outcome of another random experiment).

IV. Performance Evaluation

We implement and test LAC by means of simulations carried out with the packet-level NDN simulator CCNPL-Sim (http://systemx.enst.fr/ccnpl-sim). LAC, our latency-aware LRU, is tested against two other fully distributed caching management mechanisms: LRU+Leave-Copy-Probabilistically (LCP) and LRU [7] [16]. By fully distributed, we mean mechanisms that do not require the exchange of any specific signaling between caches. Similarly to [1] [11], the content requests are assumed to follow a Poisson process with a 1 object/s rate. Objects are requested over a catalog of 20,000 Zipf-ranked objects with skewness $\alpha = 1.7$. Previous works have also modeled content popularity distribution as Zipf-like [2] and infer identical skewness [10]. We evaluated LAC with function f set to the mean latency of all ever-cached objects.

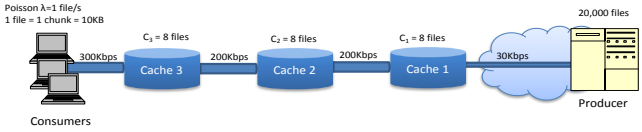
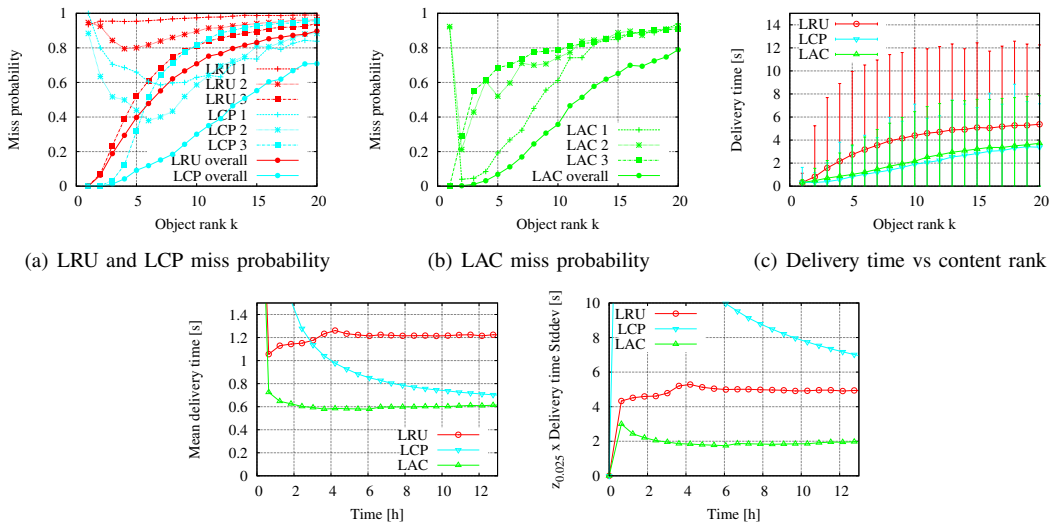


Fig. 1: Simulated line topology.

1) Line topology network: First, we consider the setting in Fig. 1, with three caching nodes in-line between the users and the publishing server. The four links from the consumers up to the publisher have capacities equal to 300Kbps, 200Kbps, 200Kbps and 30Kbps respectively. Cache sizes are equal to 80KBytes. Each object has an average size of 10KBytes, that we also take as fixed packet size. LCP is parametrized with the probability $p = 0.1$ and corresponds to the lowest



(a) LRU and LCP miss probability

(b) LAC miss probability

(c) Delivery time vs content rank

(d) Evolution of the mean delivery time

(e) Evolution of the delivery time standard deviation

Fig. 2: Line topology simulation: LAC decreases LRU delivery time by 50% and outperforms LCP on convergence.

mean latency-aware caching decision probability, Cache 3's. We configure LAC with $\beta = \gamma = 5$ to stress the rejection of quickly delivered objects. Related results are reported in Fig. 2. The resulting link load ρ on downlinks from the repository to the users is respectively : (0.5, 0.01, 0.03, 0.27) under LRU, (0.27, 0.02, 0.02, 0.27) under LCP and (0.22, 0.04, 0.06, 0.27) under LAC.

Clearly, the expensive traffic to the publisher decreases significantly with LAC, while very little increase can be observed on the other links. The tremendous gain in delivery time (50% of LRU's) can be appreciated in both its first and second moments. Such a delivery time standard deviation decrease plays a central role in stabilizing customers quality of experience.

2) Tree topology network: The next results are those achieved in the ICN setting in Fig. 3, spanning a binary tree topology whose seven caching nodes are spread over three network levels, between the users and the repository (publishing server). In this configuration, cache sizes are 8MBytes. Object size is taken equal to 1 MB. Downlink capacities from the users up to the repository are 30Mbps-capable, except the last one toward the repository, which is 9Mbps. Each packet has an average size of 10KBytes, making every object equal to 100 packets in size. Caches are equipped for LAC decision, with $\beta = \gamma = 3$. Cache 4 is on the first layer (the closest to the consumers), Cache 8 on the second layer and Cache 10 on the third (the farthest to the users). LCP's $p = 0.03$. That corresponds to LAC's mean latency-aware caching decision probability. We report the related charts in Fig. 4.

The observed link load ρ on downlinks from the repository to the users is respectively: (0.7, 0.31, 0.18, 0.6) under LRU, 0.7, 0.07, 0.33, 0.6) under LCP and (0.7, 0.12, 0.23, 0.6) under LAC. Again, our LAC mechanisms allows to lower maximum and average link load over the network. So, even though LAC reduces by half LRU's load between layer 3 and layer 2 caches,

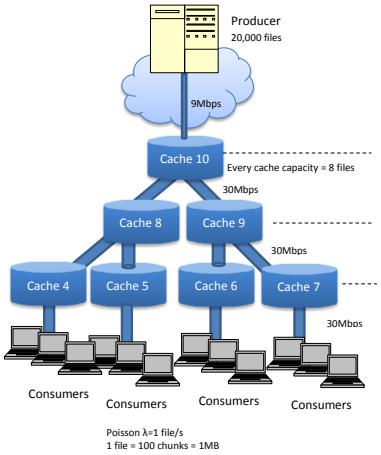


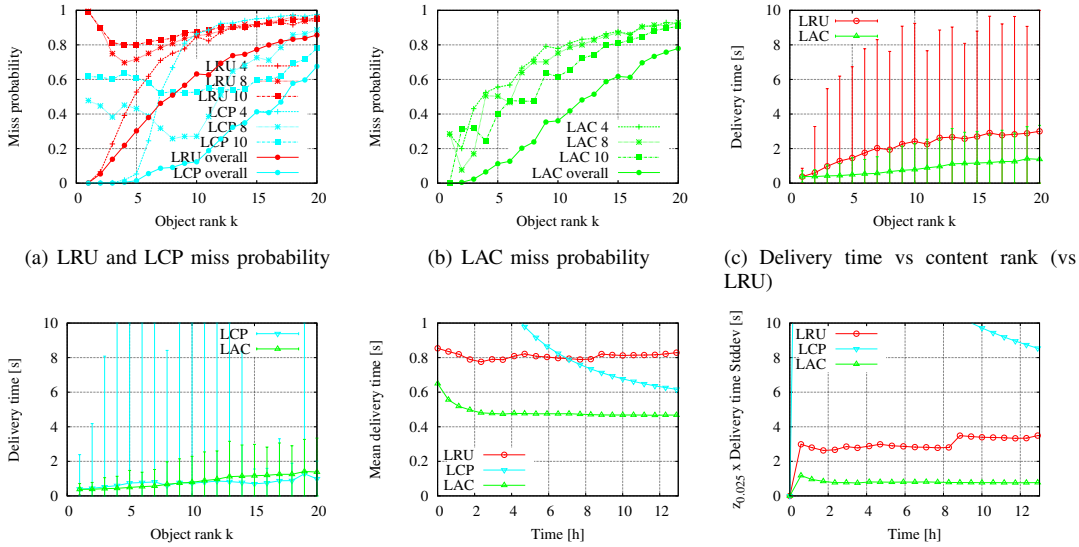
Fig. 3: Simulated tree topology

it still rely on caching delegation, which implies some inter-cache traffic.

Finally, we observe as a general rule that implementing LAC decreases the overall cache miss probability i.e. the probability that all solicited caches fail to serve the requested object. It also decreases and stabilizes the overall object delivery time. Indeed, the mean delivery time and the 95% confidence interval around the average, both decrease by up to 50%. Note also that this overall improvement is not achieved to the detriment of the convergence speed, unlike LCP. The latter, indeed, exhibits tremendously slow convergence and extremely high delivery time standard deviation.

V. Conclusion and Future Work

In the paper, we showed the benefits of leveraging latency for caching decisions in ICN and proposed LAC, a latency-aware cache management policy that bases cache insertion



(a) LRU and LCP miss probability

(b) LAC miss probability

(c) Delivery time vs content rank (vs LRU)

(d) Delivery time vs content rank (vs LCP)

(e) Evolution of the mean delivery time

(f) Evolution of the delivery time standard deviation

Fig. 4: Tree topology simulation: LAC decreases LRU delivery time by 30% and outperforms LCP on convergence.

decisions on measurements of residual latency over time on a per-object basis. While keeping the same low complexity as standard LRU with probabilistic cache insertion, it provides a finer-grained differentiation of content in terms of expected residual latency. Two main advantages have been demonstrated: (i) superior performance in terms of realized delivery time at the end-user plus maximum and average link load reduction, when compared to classical LRU and probabilistic caching approaches; (ii) faster convergence with regard to probabilistic caching approaches along with reduced standard deviation.

We leave for future work a thorough characterization of LAC dynamics, especially in a network of caches, where the coupling with hop-by-hop forwarding may be addressed through a joint optimization. The sensitivity to variations in network conditions and routing will also be investigated to highlight the benefit in terms of self-adaptiveness of a measurement-based approach with regard to classical latency-insensitive approaches.

REFERENCES

[1] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda. Congestion-aware caching and search in information-centric networks. In *Proceedings of the 1st International Conference on Information-centric Networking*, ICN '14, pages 37–46, New York, NY, USA, 2014. ACM.

[2] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: Incrementally deployable icn. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 147–158, New York, NY, USA, 2013. ACM.

[3] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy. Performance evaluation of the random replacement policy for networks of caches. *CoRR*, abs/1202.4880, 2012.

[4] A. Ioannou and S. Weber. Towards on-path caching alternatives in information-centric networks. In *Local Computer Networks, 2014 IEEE International Conference on*, 2014.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[6] P. R. Jelenković and A. Radovanović. Optimizing lru caching for variable document sizes. *Comb. Probab. Comput.*, 13(4-5):627–643, July 2004.

[7] N. Laoutaris, S. Syntila, and I. Stavrakakis. Meta algorithms for hierarchical web caches. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 445–452, 2004.

[8] Z. Ming, X. Mingwei, and D. Wang. Time-shifted tv in content centric networks: The case for cooperative in-network caching. In *Proceedings of the IEEE International Conference on Communications, 2011*, ICC'11, 2011.

[9] Z. Ming, X. Mingwei, and D. Wang. Age-based cooperative caching in information-centric networks. In *Proceedings of the IEEE Nomen 2012*, Nomen'12, 2012.

[10] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti. Characterizing web-based video sharing workloads. *ACM Trans. Web*, 5(2):8:1–8:27, May 2011.

[11] S. Oueslati, J. Roberts, and N. Sbihi. Flow-aware traffic control for a content-centric network. In *INFOCOM, 2012 Proceedings IEEE*, pages 2417–2425, March 2012.

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

[13] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pages 55–60, New York, NY, USA, 2012. ACM.

[14] D. Starobinski and D. Tse. Probabilistic methods for web caching. *Perform. Eval.*, 46(2-3):125–137, Oct. 2001.

[15] J. Tong, G. Wang, and X. Liu. Latency-aware strategy for static list caching in flash-based web search engines. In *Proceedings of the 22Nd ACM International Conference on Information and Knowledge Management*, CIKM '13, pages 1209–1212, New York, NY, USA, 2013. ACM.

[16] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie. Optimal cache allocation for content-centric networking. In *IEEE Intl. Conference on Network Protocols*, 2013.

[17] Y.-T. Y. Yu, F. Bronzino, R. Fan, C. Westphal, and M. Gerla. Congestion-aware edge caching for adaptive video streaming in information-centric networking. In *CCNC'15*, 2015.