Response Times in Time-to-Live Caching Hierarchies under Random Network Delays

Karim Elsayed
University of Duisburg-Essen
karim.elsayed@uni-due.de

Amr Rizk
University of Duisburg-Essen
amr.rizk@uni-due.de

***Abstract*—Time-to-Live (TTL) caches decouple the occupancy of objects in cache through object-specific validity timers. State-of-the-art techniques provide exact methods for the calculation of object-specific hit probabilities given entire cache hierarchies with random inter-cache network delays. The system hit probability is a provider-centric metric as it relates to the origin offload, i.e., the decrease in the number of requests that are served by the content origin server. In this paper we consider a user-centric metric, i.e., the response time, which is shown to be structurally different from the system hit probability. Equipped with the state-of-the-art exact modeling technique using Markov-arrival processes we derive expressions for the expected object response time and pave a way for its optimization under network delays.**

I. Introduction and Problem Statement

We consider Time to live (TTL) cache hierarchies where at each cache an object request results in a hit (object found in cache) or a miss (object has to be fetched from parent cache). When an object is admitted to some cache after a miss or when an object hit occurs at this cache the validity timer of the object *(its TTL)* is (re)started. Upon expiry of this timer the object is evicted from cache. TTL caches are appealing as they (i) decouple the object dynamics [1], (ii) allow deriving closed-form expressions for cache performance metrics [2]–[6], (iii) provide analytical relations to prominent cache algorithms such as LRU [5], [6], and (iv) allow per object optimizations.

We consider cache hierarchies as the example depicted in Fig. 1. In contrast to the majority of the related work that assumes instantaneous object spawning upon misses, we assume random network delays between the caches. There are several metrics to evaluate the performance of a cache hierarchy - most importantly the hit probability P_h, the occupancy and the response time. The system hit probability[1] is a provider-centric metric as it relates to the origin offload, i.e., the decrease in the number of requests that are served by the content origin server. The response time as a user-centric metric that affects the application quality of service. *The main goal of this paper is to calculate the expected response time for TTL cache hierarchies under random network delays.*

The authors in [7] derive the exact hit probability and response time of single TTL cache under non-zero delay. Going from the exact formulation for a single cache to approximate formulations for hierarchies, the work in [5] calculates the system hit probability for TTL cache hierarchies assuming zero network delays. approximation, which essentially assumes that the sum of renewal processes is a renewal process, may exhibit significant errors when compared to the exact cache performance [6]. The authors in [6] propose an exact model for TTL cache hierarchies using Markov arrival processes (MAPs) under zero network delays. Note that the classical model for the (TTL) cache hierarchies in [1]–[6], [8]–[10] assumes an ideal scenario of zero fetching delay. The work in [11] builds upon the MAPs model in [6] and provides a recursive MAP construction approach that exactly models TTL hierarchies under random network delays given the distributions of the TTL T, delay Δ and inter-request time X. The results in [11] show the impact of the network delays on the system hit probability and the object occupancy.

[1]i.e. the object hit probability for the entire hierarchy.

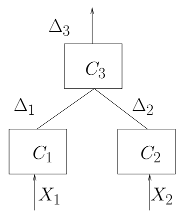

Fig. 1: TTL Cache hierarchy with a parent cache and two children caches: Object requests arrive to the C_1, C_2 according to renewal processes with PH distributed inter-request times X_i. Misses from the children are forwarded to the parent cache, where further misses are forwarded to the origin server. Object fetch delays along the different links are given by random variables Δ_i.

In this paper, we focus on TTL cache trees under non-zero random fetching delays as described in [11]. Such cache trees arise naturally, e.g., in video streaming infrastructures [12], [13]. We assume that the inter-request time, the TTL and the cache to cache (inter-cache) network delays are independent and identically distributed. The main problem and focus of this paper, unlike the works in [6], [11] is to derive expressions on the expected response time for a cache hierarchy. The difficulty in calculating the hierarchy response time lies in the fact that the system response time depends on the individual miss streams of the caches within the hierarchy as well as the state of the object in each cache. This stands in contrast to the calculation of the system hit probability which only depends on the outgoing stream of misses the top of the hierarchy towards the origin server (see Fig.1.) For the response time fetching an object from the root cache in a tree is different from having it located in a leaf cache. In addition, aggregate requests, i.e requests arriving during the fetching of the object, see different expected response times depending on the moment of arrival.

In this paper, we derive expressions for the expected response time from the MAP for all the input streams combined, which is denoted the *system expected response time*. We also derive response time expressions for individual input streams. Finally, we show how to calculate the expected response time for a request conditioned it being a hit or a miss.

II. Response time Calculation from MAPs

In this section, we follow [11] and consider a given MAP that models the object occupancy in a cache hierarchy under random network delays. We briefly review the model of cache hierarchies under network delays using MAPs from [11] before deriving the expected response time.

A. Review of the MAP model of cache hierarchies

A Markov arrival process is governed by two Markov processes: a background process $J(t)$ and a counting process $N(t)$ [14]. The transitions in a MAP are represented using two Matrices $(\mathbf{D}_0, \mathbf{D}_1)$ where $\mathbf{D}_0$ is the hidden transition matrix that partially controls $J(t)$ while $\mathbf{D}_1$ contains the active transitions that not only control $J(t)$ but also exclusively controls the counting process $N(t)$. For modelling caching hierarchies, $N(t)$ is used as the counting process of the cache misses and $\mathbf{D}_1$ contains transitions that only contribute to that.

The MAP model representing the TTL cache hierarchy under network delays as given in [11] is recursively calculated using the superposition of the MAPs representing the single caches in the hierarchy. The final MAP derivation depends on two main operations: level superposition, i.e, the superposition of sibling cache MAPs and line superposition, i.e, the superposition of parent-children cache MAPs.

To explain how the MAP is recursively calculated, consider the example of a 3 cache binary tree as in Fig. 1. Each cache is represented by a MAP featuring the state of the object within it given the TTL, delay and inter-request time distribution. Fig. 2 shows an example of a single cache MAP with exponentially distributed inter-request time X with rate λ, where the delay and the TTL each follow an Erlang-2 distribution with rates λ_D and λ_T respectively. The object in the cache can be in one of three states: the object in the cache *states* I_i, the object out of the cache *state 0* and the object is being fetched *state* δ. The object stays in the cache according to the TTL Erlang-2 distribution. Similarly, the delay is represented by the two states δ_i. Here we consider a reset TTL, i.e, the TTL is renewed upon a hit. According to [11] the total hierarchy MAP is computed as sibling single MAPs are first superposed to form the children MAP which is then superposed with the parent MAP. Note that the model in [11] extends to general PH distributions while we depict a simple example here only for clarity of the exposition.

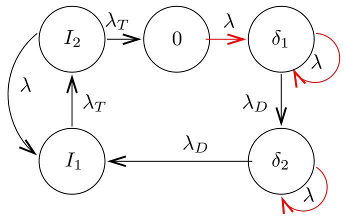
$$\begin{tikzcd}
I_2 \arrow[r, "\lambda_T"] \arrow[d, bend right, "\lambda"'] & 0 \arrow[r, "\lambda"] & \delta_1 \arrow[loop right, "\lambda"] \arrow[d, "\lambda_D"] \\
I_1 \arrow[u, "\lambda_T"'] & & \delta_2 \arrow[ll, "\lambda_D"] \arrow[loop right, "\lambda"]
\end{tikzcd}$$

Fig. 2: Single cache MAP. State 0: the object not in the cache, States I_i: the object in the cache and the TTL is counting down, States δ_i the object is being fetched. Red arrows: active transitions, black arrows: hidden transitions.

B. Computing the expected response time

In this section, we calculate the expected response time of a request in the cache hierarchy due to the random network delays. In the following, we will differentiate between the expected response time for all input streams vs. for individual streams.

Consider first the example of a parent and two child caches shown in Fig. 1. A request arriving at cache C_1 will have a random response time which is majorized when C_1 and C_3 do not have the object. An additional challenge to calculating the response time is the fact that aggregate requests (requests arriving at a cache in the fetching state) see an expected response time that varies depending on the time of arrival. This effect is irrelevant for exponentially distributed delays. For further clarification consider the MAP given in Fig. 2, a request arriving at the cache in state δ_1 or δ_2 will have an expected response time of $2/\lambda_D$ or $1/\lambda_D$ respectively. Thus, the expected response time has to be defined for each state in the MAP. We denote the vector of the state expected response time as $\boldsymbol{\alpha}$ with components $\alpha_i = \mathrm{E}[\Delta|S_i]$ where $S_i \in S$ is the ith state of the set of the states of the MAP S.

1) System expected response time: The expected system response time is the expected response time for all the request streams constituting the input to the hierarchy. We calculate the response time by iteratively accumulating the inter-cache expected response time for the output (miss) requests at each cache. By inter-cache expected response time we denote the expected time the output requests at a cache requires to obtain the object from its parent cache.

Theorem 1. *The expected object response time in a TTL treelike cache hierarchy with the ith cache at the root is*

$$\bar{R} = \frac{\sum_i \boldsymbol{\pi}^i \odot \boldsymbol{\alpha}^i \mathbf{D}_1^i \mathbf{1}}{\sum_{j \in L_s} \boldsymbol{\pi}_I^j \mathbf{D}_{1,I}^j \mathbf{1}}\ , \tag{1}$$

where $\mathbf{D}_1^i$ and $\boldsymbol{\pi}^i$ represent the active transition matrix and the steady state probability row vector of the tree with the ith cache as a root. The operation $\odot$ denotes the Hadamard (element-wise) product of two vectors. $\mathbf{D}_{1,I}^j$ and $\boldsymbol{\pi}_I^j$ are the active transition matrix and the steady state probability row vector for the MAP modelling the jth input request stream at the leaf caches, respectively. L_s is the set of indices of the leaf caches.

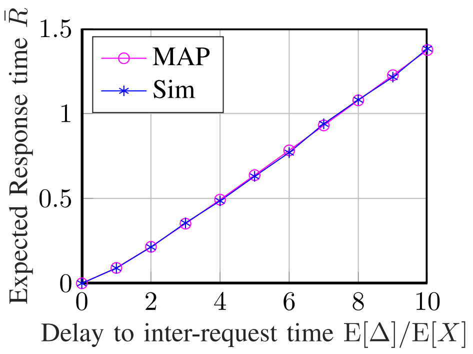

Fig. 3: Two caches in line, both having the TTL and the delay exponentially distributed with the same rate. The simulation validates the accuracy of the response time calculation from the MAP for different delay to inter-request time values and for $\mathrm{E}[T]/\mathrm{E}[X] = 2$.

The numerator of (1) represents the accumulation of the inter-cache delays. The miss stream at each cache i depends on the tree with root i. $\mathbf{D}_1^i$ is the active transition matrix representing the tree with root i. The MAP approach in [11] has the advantage that it constructs the MAP for the entire hierarchy recursively. As a result, in terms of computation, the MAPs of all the subtrees in the system are constructed independently before the recursive superposition. Hence, $\mathbf{D}_1^i$ is easily computed with the steps of the MAP construction in [11]. Now, we use $\boldsymbol{\pi}^i$ to denote the steady state probability vector for the cache subtree with the ith cache as root. This is computed from the total MAP steady state probability vector $\boldsymbol{\pi}^n$ where the superscript n denotes the index of the root of the entire hierarchy. Here, π_j^i is the jth element of $\boldsymbol{\pi}^i$. Similarly, S^n denotes the state set of the total hierarchy MAP and S^i is the state set for the subtree with the ith cache being its root. Here too, S_k^n denotes the kth element of S^n. Now, we obtain the elements of the steady state probability vector $\boldsymbol{\pi}^i$ for the MAPs of subtrees with index i as

$$\pi_j^i = \sum_{k: S_j^i \subset \{S_k^n\}} \pi_k^n \,. \tag{2}$$

Note that every miss observes a different delay conditional on the state of the cache, which appears in the Hadamard product in (1). An example of $\boldsymbol{\alpha}$ is given for the Erlang-2 delay distributions in Fig. 2 as $\alpha = [2/\lambda_D, 2/\lambda_D, 1/\lambda_D, 0, 0]$. In general, for an Erlang-m distributed network delay we find

$$\alpha_j = \begin{cases} \frac{m-k+1}{\lambda_D} & \text{for } S = \delta_k \\ \frac{m}{\lambda_D} & \text{for } S = 0 \\ 0 \;, & \text{otherwise} \,, \end{cases} \tag{3}$$

where S is a state in the corresponding MAP.

2) Per-input stream expected response time: To differentiate between the response time of input request processes at different leaf caches, i.e, to calculate the response time along one path from a leaf cache to the root, we only use the part of the active transition matrix corresponding to that input. The active transition matrix $\mathbf{D}_1^i$ for any tree with root i is a function of the request distribution parameters of all the inputs connected to it either directly or indirectly (to one of caches of its child trees). We divide $\mathbf{D}_1^i$ into sub matrices that are only dependent on one input stream j such that $\mathbf{D}_1^i = \sum_j^N \mathbf{D}_1^i(\boldsymbol{\lambda}_j)$. Here, N is the number of the input streams and $\boldsymbol{\lambda}_j$ is a vector of the parameters representing the jth input request distribution MAP. As a result, the expected response time for requests at leaf cache j is given by

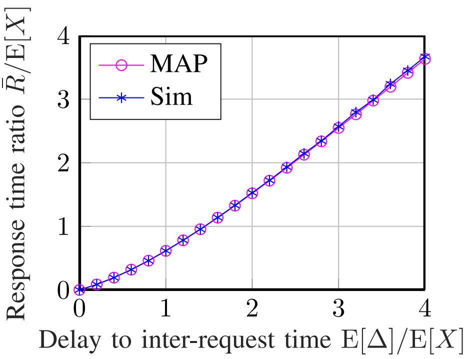

Fig. 4: Normalized expected response time in a tree hierarchy: A parent and two child caches from Fig.1 all having Erlang-2 TTL distribution, as well as, Erlang-2 delay distributions and assuming exponential inter-request times to each child cache with the homogeneous rates.

$$\bar{R}_j = \frac{\sum_i \boldsymbol{\pi}^i \odot \boldsymbol{\alpha}^i \mathbf{D}_1^i(\boldsymbol{\lambda}_j)\mathbf{1}}{\boldsymbol{\pi}_I^j \mathbf{D}_{1,I}^j \mathbf{1}} \tag{4}$$

3) Conditional Hit/miss expected response time: Now, we compute the expected response time conditional on a hit at any cache in the hierarchy. The expected response time conditional on a hit is given by

$$\bar{R}^H = \frac{\sum_{i \neq n} \boldsymbol{\pi}^i \odot \boldsymbol{\alpha}^i \mathbf{D}_1^i \mathbf{1} - \sum_{j \in L_s} \boldsymbol{\pi}^n \mathbf{D}_1^n(\boldsymbol{\lambda}_j)\mathbf{1} \sum_{k \in p_j} \alpha_1^k}{\sum_{j \in L_s} \boldsymbol{\pi}_I^j \mathbf{D}_{1,I}^j \mathbf{1} - \boldsymbol{\pi}^n \mathbf{D}_1^n \mathbf{1}} \,, \tag{5}$$

where n is the root index of the hierarchy and α_1^i is the mean response time when cache i is in state 0. The set p_j contains the indices of the caches along the path of input stream j except for the root cache.

III. Evaluation results

In this section, we present analytical and simulation-based evaluation results for the expected response time given different TTL cache hierarchies. Fig. 3 serves as a benchmark where we depict the expected response time for a hierarchy consisting on a single child cache and a single parent cache. For simplicity, we use iid delays on the different links. We vary the ratio of the expected delay $\mathrm{E}[\Delta]$ to the expected inter-request time $\mathrm{E}[X]$ as in $\mathrm{E}[\Delta]/\mathrm{E}[X]$. We fix the ratio of the expected TTL to the expected inter-request time as $\mathrm{E}[T]/\mathrm{E}[X] = 2$. The simulation results are shown for validation.

Fig. 4 shows the ratio of the expected system response time to the expected inter-request time $\bar{R}/E[X]$ for a two level binary tree cache hierarchy as in Fig.1. This normalization is beneficial when comparing different system designs at different input and delay parameters.

References

[1] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE JSAC*, vol. 20, no. 7, pp. 1305–1314, Sep 2002.

[2] N. Choungmo Fofack and S. Alouf, "Modeling modern DNS caches," in *Performance Evaluation Methodologies and Tools*, 2013, pp. 184–193.

[3] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *Proc. of IEEE Conference on Computer Communications, INFOCOM*, April 2014, pp. 2040–2048.

[4] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for lru cache performance," in *Proc. of the Teletraffic Congress (ITC)*, 2012, pp. 1–8.

[5] N. Choungmo Fofack, P. Nain, G. Neglia, and D. Towsley, "Performance evaluation of hierarchical ttl-based cache networks," *Computer Networks*, vol. 65, pp. 212 – 231, 2014.

[6] D. S. Berger, P. Gland, S. Singla, and F. Ciucu, "Exact analysis of TTL cache networks," *Performance Evaluation*, vol. 79, pp. 2 – 23, 2014.

[7] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables," in *Proc. of ACM Conference on Information-Centric Networking*, 2015, p. 69–78.

[8] K. Schomp, O. Bhardwaj, E. Kurdoglu, M. Muhaimen, and R. K. Sitaraman, "Akamai DNS: Providing Authoritative Answers to the World's Queries," in *Proc. of ACM SIGCOMM*, 2020, p. 465–478.

[9] E. J. Rosensweig, D. S. Menasche, and J. Kurose, "On the steady-state of cache networks," in *Proc. of IEEE Conference on Computer Communications, INFOCOM*, 2013, pp. 863–871.

[10] A. Rizk, M. Zink, and R. Sitaraman, "Model-based design and analysis of cache hierarchies," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, 2017, pp. 1–9.

[11] K. Elsayed and A. Rizk, "On the impact of network delays on time-to-live caching," *CoRR*, vol. abs/2201.11577, 2022.

[12] B. Alt, T. Ballard, R. Steinmetz, H. Koeppl, and A. Rizk, "CBA: Contextual Quality Adaptation for Adaptive Bitrate Video Streaming," in *IEEE Conference on Computer Communications, INFOCOM*, 2019, pp. 1000–1008.

[13] C. Koch, J. Pfannmüller, A. Rizk, D. Hausheer, and R. Steinmetz, "Category-Aware Hierarchical Caching for Video-on-Demand Content on Youtube," in *Proceedings of the ACM Multimedia Systems Conference, MMSys*, 2018, p. 89–100.

[14] S. Asmussen, *Applied Probability and Queues*, ser. Stochastic Modelling and Applied Probability. Springer New York, 2008.