ON MODELS FOR PERFORMANCE ANALYSIS OF A CORE CACHE NETWORK AND POWER SAVE OF A WIRELESS ACCESS NETWORK

Nicaise Eric Choungmo Fofack

Université de Nice Sophia Antipolis – UFR Sciences
École Doctorale Sciences & Technologies de l'Information et de la Communication
EDSTIC

THÈSE

Présentée pour obtenir le titre de :

Docteur (Dr.) en Sciences de l' Université de Nice Sophia Antipolis

Philosophiæ Doctor (Ph.D.) of Sciences of University Nice Sophia Antipolis

Spécialité: INFORMATIQUE – Computer Science

par – by

Nicaise Eric CHOUNGMO FOFACK

Affiliations : Université de Nice Sophia Antipolis – Inria Sophia Antipolis (MAESTRO)

ON MODELS FOR

PERFORMANCE ANALYSIS OF A CORE CACHE NETWORK AND POWER SAVE OF A WIRELESS ACCESS NETWORK.

Soutenue à Inria le 21/ 02/ 2014, à 14 heures devant le jury composé de :

Président :	Ernst	BIERSACK	Eurecom, Sophia Antipolis, France
Directeurs :	Philippe	NAIN	Inria, Sophia Antipolis, France
	Sara	ALOUF	Inria, Sophia Antipolis, France
Rapporteurs :	Don	TOWSLEY	University of Massachusetts, Amherst, USA
	Emilio	LEONARDI	Politecnico di Torino, Torino, Italy
Examinateurs :	Alain	SIMONIAN	Orange Labs, Issy-les-Moulineaux, France
	Giovanni	NEGLIA	Inria, Sophia Antipolis, France

Thèse de Doctorat – Ph.D. Thesis

Sur des Modèles pour l'Évaluation de Performance des Caches dans un Réseau Cœur et de la Consommation d'Énergie dans un Réseau d'Accès Sans-Fil.

On Models for Performance Analysis of a Core Cache Network and Power Save of a Wireless Access Network.

Par – By

Nicaise Eric CHOUNGMO FOFACK

Février – February 2014

On Models for Performance Analysis of a Core Cache Network and Power Save of a Wireless Access Network.

Ph.D. Candidate: Nicaise Eric CHOUNGMO FOFACK

Thesis Advisors: Philippe NAIN and Sara ALOUF

Team-Project MAESTRO, Inria Sophia Antipolis, France

Abstract

Internet is a real ecosystem. It grows, evolves and adapts to the needs of users in terms of communication, connectivity and ubiquity of users. In the last decade, the communication paradigm has shifted from traditional *host-to-host* interactions to the recent *host-to-content* model; while various wireless and networking technologies (such as 3/4G smartphones and networks, online media streaming, social networks, clouds, Big-Data, information-centric networks) emerged to enhance content distribution. This development shed light on scalability and energy efficiency issues which can be formulated as follows. *How can we design or optimize such large scale distributed systems in order to achieve and maintain high-speed access to contents while (i) reducing congestion and energy consumption in the network and (ii) adapting to the temporal locality of users demand in a continuous connectivity paradigm?*

In this thesis we focus on two solutions proposed to answer this question: *In-network caching* and *Power save protocols* for scalability and energy efficiency issues respectively. Precisely, we propose analytic models for designing core cache networks and modeling energy consumption in wireless access networks. Our studies show that the prediction of the performance of general core cache networks in real application cases can be done with absolute relative errors of order of 1%–5%; meanwhile, dramatic energy save can be achieved by mobile devices and base stations, e.g., as much as 70%–90% of the energy cost in cells with realistic traffic load and the considered parameter settings.

Résumé

Internet est un véritable écosystème. Il se développe, évolue et s'adapte aux besoins des utilisateurs en termes de communication, de connectivité et d'ubiquité. Dans la dernière décennie, les modèles de communication ont changé passant des interactions machine-à-machine à un modèle machine-à-contenu. Cependant, différentes technologies sans-fil et de réseaux (tels que les smartphones et les réseaux 3/4G, streaming en ligne des médias, les réseaux sociaux, réseaux-orientés contenus) sont apparues pour améliorer la distribution de l'information. Ce développement a mis en lumière les problèmes liés au passage à l'échelle et à l'efficacité énergétique; d'où la question: *Comment concevoir ou optimiser de tels systèmes distribués qui garantissent un accès haut débit aux contenus tout en (i) réduisant la congestion et la consommation d'énergie dans le réseau et (ii) s'adaptant à la demande des utilisateurs dans un contexte connectivité quasi-permanente?*

Dans cette thèse, nous nous intéressons à deux solutions proposées pour répondre à cette question: le déploiement des *réseaux de caches* et l'implantation des *protocoles économes en énergie*. Précisément, nous proposons des modèles analytiques pour la conception de ces réseaux de stockage et la modélisation de la consommation d'énergie dans les réseaux d'accès sans fil. Nos études montrent que la prédiction de la performance des réseaux de caches réels peut être faite avec des erreurs relatives absolues de l'ordre de 1% à 5% et qu'une proportion importante soit 70% à 90% du coût de l'énergie dans les cellules peut être économisée au niveau des stations de base et des mobiles sous des conditions réelles de trafic.

ACKNOWLEDGMENTS

Oh God, You are wonderful. Thanks my Lord for all these people you've put the way you prescribe to me. Amen!

I am deeply grateful Sara ALOUF, Philippe NAIN, Giovanni NEGLIA, Laurie VERMEERSCH and each Maestri for their support both personal and professional, and also the knowledge they have kindly shared with me. They helped me to master advanced modeling techniques, to shape my research methodology, and find my own way in the scientific networking community.

Part of this research, namely *Section 4.2 of Chapter 4*, was done in collaboration with Orange Labs in the framework of the contract "Performance models for CCN architecture". I thank in particular Orange Labs researchers Bruno KAUFFMANN, Luca MUSCARIELLO and Alain SIMONIAN for the stimulating discussions and for the valuable feedback.

Thanks to Francis MONTAGNAC and Fabrice HUET; indeed, experiments in *Section 4.3 of Chapter 4* were possible to thank Francis (IT staff at Inria, Sophia Antipolis) who kindly collected the DNS traces and Fabrice for his help in processing the large amount of data.

The material presented in *Chapter 5* is based upon work supported by the National Science Foundation under Grant No. CNS-1040781, while I was visiting the Computer Networks Research Group (CNRG) of the School of Computer Science at the University of Massachusetts, Amherst, MA, USA. I would like to thanks Don TOWSLEY and Philippe NAIN for having made this research opportunity possible.

Nicaise Eric CHOUNGMO FOFACK
Nicaise.Choungmo_Fofack@inria.fr, NicaiseEric@gmail.com
Sophia Antipolis, France

Dedications

To my wife Stassia, my son Joy-Nathan,
my mother Suzanne and my father Jean,
my brothers and sisters Aziz Borel, Line Juvette, Rhode Lucie, Alex, Reine Olivia,
for their unquestionable love.

My friends *La Famille Bisso* for their presence and support.

CONTENTS

FIGURES

TABLES

1

INTRODUCTION

Networks are “ubiquitous”: almost all human interactions occur within networks. From tiny ant colonies to gigantic planets, ***Everything is connected***. Networks rapidly grow and continuously adapt to new solicitations or requirements; thereby, they become very complex such that their current behaviors are most of the time unexpected from their initial design. Fortunately, people have shown a particular interest for such stochastic systems. Today an entire discipline–called *Networking*–and a huge research community are actively devoted to better describe the interactions and predict the behaviors of communication networks as real and dynamic ecosystems. If networks appear so challenging, another difficulty is to find the “best” or appropriate (easy to understand) formalism to study them.

Probabilistic methods have emerged many years ago as unquestionable and flexible modeling tools; they have demonstrated their power in various situations. They have been successfully applied in several domains such finance, economy, bioinformatics, computer science and networking. The interest for these approaches and their versatility can be attributed to their ability (1) to describe and explain “realistic” phenomenons where uncertainty is intrinsically present, (2) to predict and analyze the behavior of complex systems, and more important (3) to provide elegant and implementable results.

Within the applied probability framework, this thesis focuses on two problems related to *content distribution* and *Power save* issues encountered in the core and access networks respectively of recent and real communication networks. Specifically we propose models for performance evaluation of two solutions *on-demand cache networks* and *energy efficient protocols* that are deployed in content-oriented architectures and 3/4G mobile cellular networks respectively to tackle issues we previously mentioned.

1.1 From *host-to-host* to *host-to-content* interactions

1.1.1 General context: content-oriented networking

The main principle design which is probably the one that made the success of Internet was to place the intelligence at the edges of the network. This choice leads to very simple *host-to-host* interactions such that the information is accessed by locating the location/host where it is stored. Nowadays, this communication model has shown its limitations in terms of scalability (e.g. congestion at end-hosts, at end-servers and in core-routers of the network, scarcity of bandwidth, ...). Several technologies such as resource virtualization (e.g. clouds, data centers, etc.), online media streaming, social networks, have recently emerged to enforce and improve information spreading or sharing among end-users. The rationale behind these solutions is that users have more interest about the *what* i.e. their contents and how fast they can access to them; rather than the *where* i.e. the location of contents.

This new communication paradigm—called *host-to-content*—and networks where this model of interactions is the common case—also known as *content-oriented networks*—are the general subject of this dissertation. More precisely, we will analyze cache networks which arise when contents are opportunistically stored at nodes of content-oriented architectures, and we will study the energy consumption of mobile and base stations when users are continuously connected.

1.1.2 On-demand caching against content placement

On-demand caches and cache/content placement are two caching solutions to provide fast access to popular contents. The former solution stores a content only when a cache miss occurs; while the latter solution consists of loading into caches contents that have shown to be popular among users. The main difference between these caching approaches is that on-demand caching is a self-organized and auto-adapting mechanism; while cache/content placement requires external actions such as monitoring requests and populating caches every time traffic patterns change.

Our choice to study on-demand cache networks is that on-demand caches are self-organized systems and are simpler to deploy in a distributed manner in large scale networks. Moreover, one can also preload contents into on-demand caches as done by content placements.

1.1.3 Energy sustainability in wireless access networks

Energy consumption in cellular networks represents a significant proportion of the operational expenditure of mobile operators. This energy cost is even more important because users would like to be always connected to the Internet using their mobile devices.

Modeling the energy consumption is economically and environmentally interesting since *power save protocols* can be used to address the scarcity of bandwidth, to increase the lifetimes of batteries both at mobile and base stations, and to clearly take advantage of contents moved close to users by mean of to on-demand caches for example.

1.2 In-network caching and power save models: state of the art

1.2.1 Isolated repository models

Consider a collection of distinct items and a single cache which can accommodate a subset of these items. Several algorithms have been proposed to manage the cache contents and abundant literature exists on performance of these replacement policies.

Move-To-Front (MTF). The Move-To-Front policy maintains a list of items sequentially ordered from first to last. Each time a requested item is found in the list, it is brought to the first position and items are moved one position down. Mc Cabe [18] and Bitner [15] provide respectively the closed-form expressions of the stationary and transient probability of finding a specific item at each position in the list respectively given that each initial list is equally likely.

King [62] established the stationary distribution of the list. In the literature Hendricks [47] is credited for this result; however, this result follows from [62, Lemma 6] by setting the size of the list equal to the number of items in which case the Least Recently Used policy under consideration is the MTF policy. Burville and Kingman [17] provide an alternative formula by computing the stationary probability of find an item at a given position of the list. Flajolet et *al.* [40] derived the Laplace-Stieltjes Transform (LST) of the search cost in the list to cope with the combinatorial explosion involved in computing using formulas in [17].

Jelenkovic [54] derived an asymptotic formula for the stationary search cost of an item in a given position of the list as the number of items and the position go to infinity. He considered the cases where the request distribution is heavy-tail and light-tail. Jelenkovic's result [54, Theorem 3] builds up on Fill's finding [37]. Fill [37, Proposition 4.4] showed that, as the number of items goes to infinity, the stationary search cost converges in distribution to a fluid limit which is a proper random variable with a simple LST. By using Karamata's Tauberian theorem on the latter LST, Jelenkovic [54, Theorem 3] showed that the distribution of the limiting search cost inherits the heavy-tail of the request distribution.

In many applications, consecutive requests are correlated and it is therefore interesting to investigate the performance of the MTF policy in such a framework. Existing work consider the case where requests are correlated by a Markov chain process. More precisely, they assume that

the request that occurs at a given instant depends on the current and last states of an underlying Markov chain. They also consider that the item requested and the state of the Markov chain at a given instant are independent given the last state of the Markov chain. The request process is said to be Markov-modulated

Introducing the reversed Markov chain of the modulating process and the corresponding modulation process Coffman and Jelenkovic [27, Theorem 1 (resp. Theorem 3)] obtained closed-form expressions of the transient and stationary expected search costs respectively of the MTF list when requests are Markov-modulated. However, the formulas derived are in general computationally untractable and give very little insight on the impact of correlated requests on the search cost.

A connection was made by Jelenkovic and Radanovic [55] who first showed that if the request process is stationary and ergodic, then for any initial configuration of the content of the list, the search cost converges in distribution to a proper random variable. Then, they showed if file access rates are modulated by a Zipf distribution with parameter larger than one, asymptotic formulas of the stationary search cost [54] still hold. This result actually holds if requests are modulated by a more general semi-Markov process and not only a Markov chain [55, Theorem 3]. In other words, as the number of items and the position of an item in the list become large, the distribution of the limit stationary search cost is asymptotically insensitive to the correlation in the request process.

Least Recently Used (LRU). Under the Least-Recently Used (LRU) policy the cache maintains an ordered list of finite number of items (as many items as the size of the cache) from the most recently requested item to the least recently requested item among those in the cache. This policy inserts a request items at the first position, and evicts the item at the last position when cache miss occurs.

We first note that the MTF cache could be seen as an infinite cache capacity which can accommodate any number of files; and LRU caches are MTF caches with a finite capacity constraint. Hence, most of results on performance metrics on LRU caches are directly derived from results established on the MTF policy by observing that the fault probability of a LRU cache is the search cost of an item in a position larger than the LRU cache capacity.

The problem with either formulas in [62, 17] of the stationary hit probability is their exponential complexity (in the size of the cache and in the number of items) which rules out any numerical computation even for moderate cache sizes. In [40, Eq. (29)] Flajolet et *al* obtain yet another formula with a lower complexity of the same order as the product of the number of items and the cache size.

Dan and Towsley [31, Eqs. (1–2)] have derived a simple recursive algorithm of identical complexity that outputs an approximation for the stationary hit probability. This approximation

works well as long as the number of items and the cache size are not "too small". When compared to simulations the relative error is found to be less than 0.1% in most cases that have been investigated, with a maximum relative error of 3% observed for small buffer sizes (less than 5). The basic idea is to approximate the stationary probability that an item is found in a given position by the conditional probability that this item moves into its current location given that an item has been shifted in the same way.

It has been shown by Jelenkovic and Radanovic [55] that in the presence of semi-Markov modulated requests and Zipf-distributed access probabilities with parameter larger than one, the distribution of the limiting stationary search cost of MTF cache is asymptotically identical to the search cost distribution in presence of independent and identically distributed (i.i.d.) requests i.e. under the Independent Reference Model (IRM) or Poisson request processes [37]. This result implies that, asymptotically, for large cache sizes, the cache fault probability in a LRU buffer is the same under semi-Markov modulated and i.i.d. requests. An interesting question is how small can a LRU cache be while still retaining this insensitivity property? An answer to this question was given by Jelenkovic et *al.* in [56]. In words, the answer is that the critical cache size is sub-linear with respect to the sojourn times of the modulating process that determines the dependency structure. More precisely, the transition in the cache performance occurs when the cache size is of the order of the expected sojourn time that the modulated process spends in a particular state [56].

Che et *al.* [23] introduce the notion of *characteristic time* defined as the maximum inter-request time that leads to a cache hit and they accurately approximate the hit probability of LRU caches by calculating the probability that any inter-request time is smaller or equal than the characteristic time. Under their approach, LRU caches can be seen as low-pass filters. Martina et *al.* [72] and Ahmed et *al.* [4] showed that the approach based on the characteristic time is still accurate when requests are described by renewal processes and non-homogeneous Poisson processes respectively.

Random replacement (RND) and First-In First-Out (FIFO). The Random policy inserts missing item in the cache and removes any item uniformly chosen at random from the cache when room is needed. The First-In First-Out policy maintains a list where missing items are added in the head and the file at the tail of the list is evicted when room is needed.

Gelenbe [43] established that the stationary distribution of caches running these replacement algorithms. He also showed that the fault probability under these policies is identical when requests are described by the Independent Reference Model (IRM).

Fagin and Price [34], Dan and Towsley [31] provide algorithms to approximate the hit probability of RND and FIFO caches. These algorithms have a low complexity of order of the cache capacity which is significant smaller than the combinatorial solution of [43]. Gallo et

al. [42], Olivier and Simonian [79] provide exact formulas of miss probabilities of the RND and FIFO caches using a technique based on probability generating functions. They also provide closed-form asymptotic expressions when request rates are modulated by a Zipf distribution.

Following a similar approach based on the characteristic time of [23], Fricker et *al.* [41] proposed an approximate model to calculate the per-content hit probabilities on caches running RND or FIFO policy under the IRM assumption.

1.2.2 Cache network models: interconnection of caches

Compared to the single cache scenario little work has been done on architectures with several interconnected caches. In fact, few single cache models extend easily to cache networks. One reason for that is that the output process of a cache (stream of requests which have not been satisfied at this cache) no longer obeys the IRM (i.e. it is not a Poisson process) even if the requests submitted to this cache are i.i.d. In other words, if one wants to develop an exact analysis of a network composed of several LRU caches one needs to consider the network in its entirety and derive a Markov chain analysis, clearly a highly difficult task due to the strong statistical correlations between the cache states. Thus, existing results are approximations derived under the Independent Reference Model i.e. when all processes involved in the network are approximated by Poisson processes.

Rosensweig et *al.* [82] derive an approximate model for general LRU cache networks which consists to analyze each LRU cache as in isolation, then assume that both arrival and miss streams obey to the IRM, finally apply in an iterative procedure the single LRU cache approximation of [31]. More specifically, they considered a network of caches (nodes) modeled by a graph. A file can be attached to one or more servers that are in turn connected to the network. A shortest path routing is used to locate a file. It is assumed that when a cache miss occurs, the file is instantaneously downloaded into the cache. The convergence of their algorithm is not established in [82] but the authors indicate that it converged in all cases that were tested. They calculated all metrics of interest (hit probability per file, hit probability at a cache, etc.) and they reported relative errors of order of 16%. Moreover, the authors investigate the source of errors and identify system parameters that determine the accuracy of their model.

In [23], Che et *al.* extend their single cache analysis—based on cache characteristic time—to two-level tree network of LRU caches. They assume that leaves are fed by independent Poisson request processes. It is also assumed that the time required to download a document whether or not this document is cached is zero. Miss streams at leaves are approximated by a renewal process where inter-miss times are drawn from a shifted-exponential distribution; meanwhile aggregate processes at the root is approximated by a renewal process having the inter-request times characterized by the exact formula of [69] for the superposition of independent renewal processes. Hence, the streams of requests submitted to root cache are no longer

Poisson processes. Simulations reported in [23] showed that this approach is highly accurate, but still limited to two-level tree cache networks. Martina et *al.* [72] extend the model of [23] to arbitrary hierarchical cache networks where requests are described by renewal processes and flow in the same direction (i.e. from leaves to the root).

[42] is probably the only modeling attempt that addresses tree network of RND caches. Gallo et *al.* [42] extend their single cache analysis based on probability generating functions to tree network by assuming that the IRM assumption holds on the cache miss processes.

We note that none of these existing models is general enough to provide insights on performance of general and heterogeneous cache networks where caches are running different replacement algorithms such LRU, RND, FIFO, etc. The interest for such a general network model is that such heterogeneous networks have been shown to provide better performance [42]; moreover it is more likely that is the case for large scale content-oriented networks where cache providers select their own caching strategy independently of others.

1.2.3 Power save models

Power save and sleep mode in cellular networks have been analytically and experimentally investigated in the literature, mainly from the user equipment (UE) viewpoint. [97, 45] study the UE by the means of a semi-Markov chain model. The authors of [87] and [7] propose an embedded Markov chain of a M/GI/1/N queue and an M/G/1 queue with repeated vacations to model the sleep mode of base stations in IEEE 802.16e. Using Laplace-Stieltjes Transform and Probability Generating Functions, [58] derives closed form expressions for the average power consumption and the average packet delay for an UE.

Analytical models, supported by simulations, were proposed by Xiao for evaluating the performance of the UE in terms of energy consumption and access delay in both downlink and uplink [96]. Almhana *et al.* provide an adaptive algorithm that minimizes energy subject to QoS requirements for delay [6].

The works [11, 12] are closely related to our proposal and they mainly focus on the analysis of the discontinuous reception mode in 3GPP LTE and IEEE 802.16m respectively. The authors consider both the uplink and downlink packets for the UE and show that uplink packets increase the power consumption and decrease the delay.

We note that none of these existing models include both user equipment and base stations (i.e. they are restricted to UE uplink and downlink analysis, while no attention is given to the total consumption of eNB) under real traffic patterns and/or 3GPP power save protocols.

1.3 Contributions of this thesis

In this dissertation we address problems in modeling cache networks and in modeling power saving algorithms in cellular networks.

1.3.1 Performance evaluation of general and heterogeneous cache networks

Our contributions on cache network modeling are the following.

1. We propose Time-To-Live (TTL)-based models to describe the asymptotic behaviors of LRU, RND and FIFO caches in isolation when requests are described by general stationary processes.
2. We provide a unified framework for performance analysis of general TTL-based caches networks.
3. We show how our framework applies by studying three valid application cases of our TTL-based cache network model in the context of Content-Centric networks, Domain Name System, general and heterogeneous cache networks made of LRU, RND and FIFO caches.
4. Finally, we formulate a non-exhaustive list of open problems that we found of particular interest both for the networking research community and industry.

1.3.2 Power save analysis and its impact under continuous connectivity paradigm

Our contributions on this topic are as follows.

1. We provide a complete model for the behavior of users (UEs) and base stations (eNBs) in continuous connectivity and with realistic web traffic.
2. We provide a cost model that incorporates the different causes of operational costs.
3. We perform simulations in which each user has p parallel browsing sessions. The aim is to evaluate whether our study can be used when each user's traffic consists of superposed arrival processes.
4. We study the importance of a variety of model parameters by means of a *sensitivity analysis*, and we show how to use the model to minimize operational costs under QoS constraints.
5. We provide lessons learned from our study, summarizing our recommendations and suggesting a setup which achieves a good tradeoff between energy savings and QoS performance.

1.4 Mathematical frameworks

The models proposed in this thesis are built on top of several existing frameworks:

- *Theory of stationary point processes.* This modeling tool is used to describe requests generated by users in the system under analysis. It assumes that requests are correlated and their statistical properties remained unchanged with time shifting. Our main reference on this tool is the book by Bacceli and Brémaud [10].

- *Markov Renewal Theory.* This framework introduces a sub-class of stationary processes where requests are correlated by a Markov process. This tool provides a simple, versatile and analytically tractable model of correlation structure. We refer to the notes of Çinlar [22].

- *Renewal Theory.* This framework is of particular interest since it provides a class of processes that describes independent, identically and generally distributed events. It as been shown in several works [95, 5, 59] that traces and general processes may be approximated by renewal processes. Our main references on renewal theory are the books by Cox [28] and Kulkarni [65].

- *Queueing Theory.* This modeling tool is widely used in computer science. We apply it here to describe data packets buffered at mobile and base stations of next generation cellular networks.

1.5 Organization of dissertation

Jack and Harry were lost over a vast farmland while on their balloon ride. When they spotted a bicyclist on trail going through the farmland below, they lowered their balloon and yelled, "Good day, sir! Could you tell us where we are?"

The bicyclist looked up and said, "Sure! You are in a balloon!"

Jack turned to Harry and said, "This guy must be a mathematician!"

"What makes you think so?" asked Harry.

"Well, his answer is correct, but totally useless!"

The author sincerely hopes that a student, **a researcher, or an industrial** *mastering* **this thesis** *will be able to use* **our models** *to obtain correct as well as useful answers.* Kulkarni [65]

The four first chapters present our cache network model and three application cases. The fifth chapter shows our energy cost model of wireless access networks. The last chapter proposes a set of problems yet to be addressed.

Chapter 2 introduces the concept of *time-to-live* of popular replacement policies as LRU, RND, or FIFO and defines *TTL-based* models as their asymptotic behaviors. We also identify two classes of TTL-based caches which appear to be equivalent to *Geiger counters*. Other outcomes of this chapter are the proof of the characteristic time approximation on LRU caches [23, 72] and RND/FIFO caches [72]. This chapter describes the steps that might be followed if one want to derive new TTL-models for other caches. It has a particular research interest.

Chapter 3 provides a complete description of our unified framework for performance analysis of general and heterogeneous TTL-based cache networks where requests are correlated and described by Markov-renewal processes. We introduce the notion of *routing as a polytree* and we translate main network primitives into well-known operations on point processes. This chapter has theoretical contribution in the theory of counters since it extend the study of counters in isolation to network of counters.

Chapter 4 presents two application cases of our framework. The first application is in the context of content-centric networks where TTL-based policy appears as alternative to more popular replacement algorithms. The second application relies on TTL-based models to describe a recent behavior observed on DNS caches over Internet. This chapter has an applied research purpose.

Chapter 5 provides very accurate models of general and heterogeneous cache networks where caches may run either LRU, RND, or FIFO policies. This chapter tackles several limitations in terms of complexity of the exact solution we described in our framework. Its practical concern are to build an simple tool for engineers and cache network developers who want to large cache network.

Chapter 6 presents existing power save protocols and it derives their energy cost model. This chapter evaluates also the performance of the wireless access network under these protocols, reports optimal configurations of these protocols, and summarizes lessons learned from our energy cost model.

Chapter 7 poses a set of open problems on TTL-based cache networks partially or not solved by the time of the redaction of this manuscript.

Finally, *Chapter 8* concludes this thesis.

Chapters of this dissertation are self-contained and the reader may decide to go through the chapters in any order.

1.6 Publications of dissertation

Part of results obtained in this thesis were also published as conference and journal papers. Here is the list of our scientific contributions.

1. Conference papers:

   (a) Results obtained in Chapter 4, Section 4.3 are published in the proceedings of the 7th International Conference of Performance Evaluation Methodologies and Tools [73].

   N. Choungmo Fofack and Sara Alouf, “Modeling Modern DNS cache", *in Proc. ValueTools 2013*, Torino, Italy, Dec. 10-12, 2013.

   (b) Results derived in Chapter 4, Section 4.2 are published in the proceedings of the 6th International Conference of Performance Evaluation Methodologies and Tools [25].

   N. Choungmo Fofack, Philippe Nain, Giovanni Neglia and Don Towsley, “Analysis of TTL-based Cache Networks", *in Proc. ValueTools 2012*, Cargese, France, Oct. 9-12, 2012. **Best Student Paper Award**.

2. Journal papers:

   (a) Materials presented in Chapter 6 are published in the Pervasive and Mobile Computing journal of Elsevier [8]. It is an extended version of the earlier paper [70].

   Sara Alouf, Vincenzo Mancuso and N. Choungmo Fofack, “Analysis of Power Save and its Impact on Web Traffic in Cellular Networks with Continuous Connectivity", *Pervasive and Mobile Computing*, Vol. 8, No. 5, pp. 646-661, Oct. 2012.

   (b) An extended version of [25] is currently under the reviewing process of the Computer Networks journal.

   N. Choungmo Fofack, Philippe Nain, Giovanni Neglia and Don Towsley, “Performance Evaluation of Hierarchical TTL-based Cache Networks", *submitted to Computer Networks*.

2

FROM POPULAR CACHE REPLACEMENT POLICIES TO TIME-TO-LIVE (TTL)-BASED POLICIES

2.1 Summary

In this chapter we analytically study asymptotic models—that we refer to as *Time-To-Live*(TTL)-based models—for the performance analysis of classical cache management algorithms such as Least Recently Used (LRU), First-In First-Out (FIFO) and Random replacement (RND). These models are based on the characterization of the "maximum inter-request time of each file that leads to a cache hit" also known as the *characteristic time* [23]. These TTL-based models are categorized into two classes which appear to be equivalent to the general *Geiger counters of Type I* and *Type II* [22]. We present the most interesting properties of TTL-based models and we extend the concept of "expiration-based caches" by defining more general *TTL-based caches*. Unlike classical caches, a TTL-based cache is easy to analyze, fully configurable (e.g. it can mimic the behavior of existing replacement schemes), controllable (e.g. for cache optimization such as quality of service or service differentiation), and generic (e.g. for deployment in Software-Defined Network or Information-Centric Network).

Keyword 2.1 *Asymptotic analysis, Least Recently Used, First-In First-Out, Random replacement, characteristic time, Time-To-Live cache, stationary request processes.*

2.2 Introduction

Many systems such as computer memories employ caches to improve their access speed or to reduce the load on back-end components. These desired features are highly correlated with the availability of the information in the cache at memory reading instants. Hence, several cache management algorithms such as Least Recently Used (LRU), First-In First-Out (FIFO), Random replacement (RND), and Time-To-Live (TTL, see the *5-Minute Rule* by [44]) have been proposed to manage contents and maximize the efficiency of the cache under various use cases.

Meanwhile the design, configuration, and analysis of these cache systems pose significant challenges. It is therefore not surprising to note that first exact analytic results on the performance of LRU, FIFO and RND caching systems were established during the seventies by system researchers—namely King [62] and Gelenbe [43], after the invention of microprocessors in 1969 and before the industrial commercialization of the personal computer *Altair 8800* in 1975. An abundant literature exists on the performance (e.g. hit probability, search cost) of a single cache under the Independent Reference Model (IRM) or equivalently when requests are independent and identically distributed (i.i.d.) with exponentially distributed inter-request times. These requests are also said to be described by Poisson processes [38]. Under this latter assumption exact results based on the Markov chain analysis of the state of FIFO and RND caches can be found in [43]; the LRU or, its companion, the Move-to-Front (MTF) policy are studied in [15, 17, 18, 38, 40, 47, 62].

Recent Internet technologies for content distribution such as Content Distribution Networks, Peer-to-Peer networks, and Information-centric architectures have renewed the interest of the networking research community for modeling single caches with the goal of analyzing cache networks. With few exceptions, exact models of even a single cache are computationally intractable and too complex to be extended to cache networks. This situation yields the development of approximate methodologies based on Markov chain analysis [31], large deviation analysis [42], or characteristic time approximation [23, 72] for caches fed by i.i.d. requests, fluid analysis [27, 54, 55, 56, 57] when requests are described by semi-Markov processes, and again the characteristic time approximation [4] for non-stationary requests.

This latter method, the characteristic time approximation (CTA), has boosted the cache performance analysis and greatly simplified the evaluation of cache networks. The technique was introduced by Che et *al.* [23] and it consists of viewing the cache as a *low-pass filter* having a cut-off frequency whose inverse—called the *characteristic time*—is defined as the maximum inter-request time of any file that leads to a cache hit. The expiration-based caching policies introduced in [25] and [59] are concepts similar to the cache characteristic time. Many experiments have shown that the CTA leads to highly accurate results. However, only the 2012 paper by Fricker et *al.* [41] attempts to provide a theoretic explanation of experimental results

observed on LRU and RND caches in isolation fed by Poisson request processes.

In this chapter, we address this lack of theoretic framework that can explain why the CTA works, especially when the Poisson assumption does not hold (e.g. in a tandem of two LRU caches where the second cache is fed by the miss streams of the first cache which is known to be a non-Poisson process [57]). More precisely, we first provide other formulations (mathematically tractable) of the cache characteristic time of each of LRU, RND and FIFO replacement policies and we called them the TTL to remind the reader that the TTL of a cache is specific to the cache policy in contrast with the cache characteristic time (which is a general concept). Then we derive asymptotic models—called TTL-based models—for LRU, RND and FIFO caches when requests are described by *stationary and ergodic processes* and the size of the file catalog is large. Indeed, we characterize the cumbersome distribution of the TTL random variable for each of these caches and we show that this distribution converges under additional conditions to simple ones: a degenerate distribution (i.e. a deterministic TTL) for LRU and FIFO caches, or an exponential distribution for RND caches as already observed in [23, 66, 72]. Our convergence conditions appear to be assumed or satisfied in several (synthetic and trace-driven) experiments on caches that are available in the current literature [23, 41, 72] and also revisited in this chapter. We finally introduce *TTL-based caches* as a more general and generic class of caches that contains LRU, FIFO and RND replacement policies as special instances.

The chapter is organized as follows. In Section 2.3, we present relevant work close to our TTL-based modeling approach of the LRU, RND and FIFO caches. Section 2.4 introduces the notation, the statement of the problem and the general scope that will be covered in the chapter. In Section 2.5 we derive the TTL distribution of LRU, FIFO and RND caches in isolation fed by independent stationary request processes. Section 2.6 addresses the calculation of the performance metrics and sketches the differences between the LRU and FIFO/RND caches through their conceptual TTL-based models. We also show the benefits of the TTL-based approach in the sense that it greatly simplifies the description of the miss streams of a cache and later the analysis of heterogeneous cache networks. We revisit in Section 2.7 several existing work where our theoretic framework provides a mathematical foundation of the validity of their experimental results. Conclusions are found in Section 2.8.

2.3 Related work

The existing results related to our TTL approach are mainly obtained on LRU caches on very specific applications revisited in Section 2.7. Initially, the problem is stated as follows: files (labeled $n \in \{1, 2, \ldots, N\}$) of a catalog of size N are requested according to Poisson processes (or under IRM assumption [38]) on a LRU cache of capacity B, $B < N$. The file access rates

$\{\lambda_n\}_n$ are modulated by a popularity distribution.

In earlier work [54], Jelenkoviç proposed a fluid model (when N and B are large) for the fault/miss probability on LRU cache when the content-popularity distribution is either a generic light-tailed law i.e. $\lambda_n = ce^{-\delta n^\beta}$ where $c, \delta, \beta > 0$ or a generalized Zipf law i.e. $\lambda_n = 1/n^\alpha$ where $\alpha > 1$. For the latter Zipf popularity distribution, Jelenkoviç showed in [57] that performance metrics (i.e. the search cost) of the LRU cache are functions of a parameter defined as the root of a certain equation.

This parameter was defined a few years later by Che et *al.* [23] as the cache *characteristic time*. Assuming also Zipf-like distribution of content-popularity, [23] experimentally found that as N and B increase, the characteristic times (which are per-file defined random variables) in the LRU cache can be approximated by a deterministic value root of a certain equation; this is how the CTA is defined. The latter equation is obtained by equalizing the sum of stationary probabilities that each file is in the LRU cache at any time and the cache size B. Relying on the CTA, Laoutaris [66] proposed a closed-form method to compute the performance of a LRU cache by assuming a generalized Power law demand.

The paper mathematically closest to our work is a 2012 paper by Fricker *et al.* [41] which provided for the first time an explanation of the validity of the CTA on a LRU cache fed by Poisson requests processes with rates modulated by a Zipf-like distribution. Applying the Central Limit Theorem (CLT), the authors separate the validity of the CTA from the Zipfian popularity of requests and provide a use case with a Geometric law not covered in [23]. They also established sufficient conditions that do not require the cache size B being large. Finally they proposed a CTA for a RND cache where the characteristic time is approximated by a unique deterministic value. However, we prove in Section 2.5.2 that their last result *is only true in expectation*. We shall see that the characteristic times of files in a RND cache converge in distribution to an exponentially distributed random variable as already shown in experiments by Martina et *al.* [72].

The case of LRU caches under non-stationary traffic conditions such as in-homogeneous Poisson request processes was studied in [4]. Ahmed et *al.* [4] experimentally showed that the CTA can be derived to accurately approximate performance metrics of LRU caches in isolation. This non-stationary traffic model is also known as the Shot Noise Model (SNM, [94]) and it is addressed latter in *Chapter 7*.

Many results derived in this work significantly extend those of [41] on LRU, FIFO and RND caches. Here, requests are described by general stationary point processes. One of the motivations is that the IRM (or the Poisson process) assumption does not hold for the sequence of requests not served by a cache. If simulating a single cache in order to check the accuracy of approximations is affordable, it becomes critical when we think about large, general and heterogeneous cache networks. It is now the case and theoretic results are indeed needed.

2.4 Problem statement

2.4.1 Workload model

We consider a catalog of N different files labeled $n \in 1, \ldots, N$. Successive requests for file n follow **an orderly stationary point process** $\mathcal{R}_n := \{t_i^n\}_{i\in Z}$ such that $\cdots < t_{-1}^n < t_0^n \leq 0 < t_1^n < t_2^n < \cdots$ by convention. In other words, t_i^n ($i \geq 1$) is the occurrence time of the i-th request of file n after time $t = 0$. In this setting the successive inter-request time sequence $\{\tau_i^n\}_{i\in Z}$ for file n is stationary with $\tau_i^n := t_i^n - t_{i-1}^n$. Throughout we will assume that the expected inter-request time $E[\tau_1^n]$ is finite for each file n, so that the cumulative distribution function (CDF) of the first request for file n after time $t = 0$ is given by [10, Formula (4.2.4b)]

$$P(t_1^n < t) = \frac{1}{E[\tau_1^n]} \int_0^t P(\tau_1^n > u)du, \; n = 1, \ldots, N. \tag{2.1}$$

In the literature the random variable (rv) t_1^n is referred to as the forward recurrence time of the stationary process $\mathcal{R}_n$.
Pick one $n \in \{1, \ldots, N\}$ and assume without loss of generality (wlog) that $t_0^n = 0$, i.e. a request for file n is made at time $t = 0$. Alternatively stated, we consider the Palm probability, denoted by P^0, of the stationary point process $\{t_i^n\}_{i\in Z}$, which has the property that $P^0(t_0^n = 0) = 1$, see [10, Formula (3.1.1)]. When $E^0[t_1^n] < \infty$ (or equivalently when $E[\tau_1^n] < \infty$) then

$$P(t_1^n < t) = (E^0[t_1^n])^{-1} \int_0^t P^0(t_1^n > u)du, \tag{2.2}$$

with E^0 the expectation operator associated with P^0 [10, Formula (4.4.4)] and we retrieve (2.1). **We further assume that the stationary point processes $\mathcal{R}_1, \ldots, \mathcal{R}_N$ are independent**.

2.4.2 Cache model and other processes at hand

We consider a cache which can accommodate at most B different files of a catalog of size $N > B$. Files are accessed according to the workload model described above. Our aim objective is to characterize the TTL denoted $T_{N,n}$ of a file n in the cache. The random variable $T_{N,n}$ is the *maximum residency time in the cache of file n if this file was requested only once* (time-to-live definition) or the *maximum inter-request time of file n that leads to cache hit* (characteristic time definition). A mathematical definition of $T_{N,n}$ will be given next. In the following, the terms TTL and characteristic time will be interchangeably used to refer to the rv $T_{N,n}$ of file n.

We denote by $\mathcal{M}_n := \{m_i^n\}_{i\geq 1}$ the point process formed by the sequence of time instants where a request of file n yields a cache miss and $\mathcal{M} := \{m_i\}_{i\geq 1}$ the aggregated miss process of the cache. Regarding the process $\mathcal{M}$, the sequence of successive inter-miss times is defined as $\{\vartheta_i\}_{i\geq 1}$ where $\vartheta_i = m_i - m_{i-1}$. Similarly, we introduce the aggregated process $\mathcal{M}_{\bar{n}}$ resulting

from the superposition of the $N-1$ miss point processes $\{\mathcal{M}_j, j = 1,\dots,N : j \neq n\}$ and $\{\vartheta_i^{\tilde{n}}\}_{i\geq 1}$ its sequence of inter-miss times.
Note that the miss process $\mathcal{M}_n$ is a stationary marked point process with the marks corresponding to the total sojourn time of file n in the cache (see Remark 2.8, for more details); and thus, $\mathcal{M}$ and $\mathcal{M}_{\tilde{n}}$ are also a stationary point processes [10, Sect. 1.4.2].

2.5 Single cache: TTL-based models

2.5.1 Least Recently Used policy

The Least Recently Used (LRU) policy manages a cache as a list of capacity B where requested files are added in the head of the list and the file to be evicted when room is needed is the one at the tail. This policy has the advantage of keeping the most popular files in the cache.

Let us define the indicator functions $X_n(t) := 1(t_1^n < t)$ for $n = 1,\dots,N$. In other words, $X_n(t) = 1$ if there has been at least one request for file n in $[0,t)$ and $X_n(t) = 0$ otherwise. Since $t_0^n = 0$ i.e. file n entered in the cache at time $t = 0$, we have $X_n(t) = 1$; and **because we consider first that there is no other requests for file n after the one that occurs at t_0^n**, we define $M_{N,n}(t)$ as the number of different files in the catalog $\{1,\dots,N\}\backslash\{n\}$ which are requested in $[0,t)$. We have

$$M_{N,n}(t) = \sum_{j=1, j\neq n}^{N} X_j(t). \tag{2.3}$$

We may now define $T_{B,n}$ as

$$T_{B,n} = \inf\{t > 0 : M_{N,n}(t) = B\}. \tag{2.4}$$

The latter equation states that file n will leave the cache as soon as B different files in the catalog $\{1,\dots,N\}\backslash\{n\}$ will have been requested. We recall that file n in the cache at $t_0^n = 0$ and no further requests for file n occur (i.e. $t_1^n = \infty$ up to now). From (2.4) we note that

$$\begin{aligned} P(T_{B,n} < t) &= P(M_{N,n}(t) \geq B) \\ &= 1 - \sum_{k=0}^{B} P(M_{N,n}(t) = k). \end{aligned} \tag{2.5}$$

The last equation states that the CDF of the TTL $T_{B,n}$ is characterized once the probability mass function (PMF) of the discrete rv $M_{N,n}(t)$ is found.
For finite cache size B and bounded catalog size N, the exact PMF of $M_{N,n}(t)$ is given as follows.

Proposition 2.1 (PMF of $M_{N,n}(t)$) *The rv $M_{N,n}(t)$ has a Poisson Binomial distribution with parameters $\{p_j(t), j = 1, \ldots, N : j \neq n\}$ under the enforced assumptions that the request processes $\{\mathcal{R}_n\}_n$ are stationary and independent.*

This proposition results from the very basic definition of both $M_{N,n}(t)$ (see (2)) and of a Poisson Binomial distribution (sum of independent non-identical Bernoulli trials). In the caching context the result in Proposition 2.1 has a limited interest since calculating the PMF of $M_{N,n}(t)$—and therefore the CDF of $T_{B,n}$ thanks to (2.5)—will be impossible for typical values of N as a catalog may contain a prohibitively large number of files. Recently, a numerical algorithm proposed in [48] to efficiently calculate the Poisson Binomial distribution for values of N up to (approx.) 15000. If N does not exceed this threshold we may therefore use this algorithm to compute the CDF of $T_{B,n}$ in closed-form. This result is in contrast with the simulations carried out in [66], [41] and [23] (for $N = 1000$, $N = 10000$ and $N = 20000$, respectively) to validate the approximate models developed therein.

Next, we investigate the case when N goes to infinity. It is of particular interest to study the convergence of the sum $M_{N,n}(t)$ in order to calculate its PMF; and thus, find the CDF of $T_{B,n}$.

Proposition 2.2 (Convergence result) *Fix $t > 0$. Under the enforced assumptions that the request processes $\{\mathcal{R}_n\}_n$ are stationary and independent, the rv $M_{N,n}(t)$ converges almost surely (a.s.) to a finite rv $M_n(t)$ as $N \to \infty$ if and only if $\lim_{N\to\infty} \mu_{N,n}(t) < \infty$, where $\mu_{N,n}(t)$ is the mean of $M_{N,n}(t)$ given in* (2.6).

Proof Let $\mu_{N,n}(t) := E[M_{N,n}(t)]$, $\sigma^2_{N,n}(t) := Var(M_{N,n}(t))$, $c^2_{N,n}(t) := \dfrac{\sigma^2_{N,n}(t)}{\mu^2_{N,n}(t)}$ and $\gamma_{N,n}(t) := E\left[\left(\dfrac{M_{N,n}(t) - \mu_{N,n}(t)}{\sigma_{N,n}(t)}\right)^3\right]$ be the expectation, the variance, the squared coefficient of variation (scv) and the skewness of $M_{N,n}(t)$, respectively.

Moments of $M_{N,n}(t)$ Having $p_n(t) = P(t^n_1 < t)$ given in (2.1), routine algebra for a finite sum of independent Bernoulli random variables shows that for fixed $t > 0$ the mean, the variance, the scv and the skewness of $M_{N,n}(t)$ are respectively given by:

$$\begin{aligned}
\mu_{N,n}(t) &= \sum_{j=1, j\neq n}^{N} p_j(t) \qquad (2.6)\\
\sigma^2_{N,n}(t) &= \sum_{j=1, j\neq n}^{N} p_j(t)(1 - p_j(t))\\
c^2_{N,n}(t) &= (\sigma_{N,n}(t)/\mu_{N,n}(t))^2\\
\gamma_{N,n}(t) &= \sigma^{-3}_{N,n}(t) \sum_{j=1, j\neq n}^{N} p_j(t)(1 - p_j(t))(1 - 2p_j(t))
\end{aligned}$$

Inequalities on the moments Also the following inequalities trivially hold ($t > 0$):

$$0 \leq \sigma^2_{N,n}(t) \leq \mu_{N,n}(t) \tag{2.7}$$

$$0 \leq c^2_{N,n}(t) \leq \mu^{-1}_{N,n}(t) \tag{2.8}$$

$$0 \leq |\gamma_{N,n}(t)| \leq \sigma^{-1}_{N,n}(t) \tag{2.9}$$

Convergence criteria Using the latter inequalities, one can easily establish that

$$\sum_{j=1,j\neq n}^{N} P(|X_j(t)| > 1) = 0 < \infty$$

$$\sum_{j=1,j\neq n}^{N} E(X_j(t)\ 1_{\{|X_j(t)|\leq 1\}}) = \mu_{N,n}(t)$$

$$\sum_{j=1,j\neq n}^{N} Var(X_j(t)\ 1_{\{|X_j(t)|\leq 1\}}) = \sigma^2_{N,n}(t) \leq \mu_{N,n}(t)$$

where the latter inequality comes from (2.7). Letting $N \to \infty$ in the above shows that these three sums are finite if and only if $\lim_{N\to\infty} \mu_{N,n}(t) < \infty$ (note that the limit exists as $\mu_{N,n}(t)$ is a non-decreasing sequence in the variable N). From the Kolmogorov's three-series Theorem (see e.g. [32, p.70]) we may conclude that $M_{N,n}(t)$ converges almost surely if and only if $\lim_{N\to\infty} \mu_{N,n}(t) < \infty$. This concludes the proof. ∎

Proposition 2.2 is an existence result that does not characterize $M_n(t)$, the (a.s.) limit of $M_{N,n}(t)$ when $\lim_{N\to\infty} \mu_{N,n}(t) < \infty$. The following results are based on two classical asymptotic approximations of the Poisson-Binomial distribution: a Poisson approximation [91, Corollary 1.3] and a Gaussian approximation [88, Theorem 1].

Proposition 2.3 (Poisson CDF) *Under the enforced assumptions that the request processes $\{\mathcal{R}_n\}_n$ are stationary and independent, $M_{N,n}(t)$ may be approximated by a Poisson random variable with mean ρ if $\rho = \mu_{N,n}(t) \in (0, 3]$. More precisely, for $k \in \{1, \ldots, N-1\}$,*

$$\left| P(M_{N,n}(t) \leq k) - \sum_{l=1}^{k} \frac{\rho^l e^{-\rho}}{l!} \right| \leq \Delta_{\mathcal{P}}(\rho, k) \tag{2.10}$$

where $\rho = \mu_{N,n}(t)$ and $\Delta_{\mathcal{P}}(\rho, k)$ is given by

$$\Delta_{\mathcal{P}}(\rho, k) = (1 - e^{-\rho}) \min\left(1, \frac{e^{\rho}}{k+1}\right). \tag{2.11}$$

The bound in (2.11) is obtained by combining the inequality $\sum_{j=1,j\neq n} p_j^2(t) \leq \rho$ and the result in [91, Formula (1.10)]. The range of application of the result in Proposition 2.3 is limited since $\mu_{N,n}(t)$ cannot exceed three for the bounds in (2.10) to hold. The next proposition provides another approximation of the (a.s.) limit of the rv $M_{N,n}(t)$ for a wide range of values of $\mu_{N,n}(t)$.

Proposition 2.4 ("Refined" Normal CDF) *Under the enforced assumptions that the request processes $\{\mathcal{R}_n\}_n$ are stationary and independent, $M_{N,n}(t)$ may be approximated by a "refined" Gaussian rv with mean $\mu_{N,n}(t)$ and variance $\sigma^2_{N,n}(t)$ if $\sigma^2_{N,n}(t) \geq 25$.*
More precisely, for $k \in \{1,\ldots,N-1\}$,

$$\left| P(M_{N,n}(t) \leq k) - R\left(\frac{k+0.5-m}{s}\right) \right| \leq \Delta_{\mathcal{N}}(s) \tag{2.12}$$

where $m = \mu_{N,n}(t)$, $s = \sigma_{N,n}(t)$, the error bound $\Delta_{\mathcal{N}}(s)$ and the CDF $R(x)$ are respectively given in [76, Formula (1.16)] and [76, Formula (1.3)] by

$$\Delta_{\mathcal{N}}(s) = \frac{0.3056}{s^2} \tag{2.13}$$

and

$$R(x) = \Phi(x) + \frac{1}{6}\gamma_{N,n}(t) \times (1-x^2)\phi(x) \tag{2.14}$$

with $\Phi(x)$ and $\phi(x)$ denoting the CDF and PDF of the standard Normal rv.

Although, it is claimed in [76, Remark 1.2] that the bound $\Delta_{\mathcal{N}}(s)$ is valid for $\sigma^2_{N,n}(t) > 0$, we provide a simple comparison of $\Delta_{\mathcal{N}}(s)$ and $\Delta_{\mathcal{P}}(\rho,k)$ in order to choose the most accurate approximation of $M_{N,n}(t)$.

Comparison of the bounds $\Delta_{\mathcal{P}}(\rho,k)$ and $\Delta_{\mathcal{N}}(s)$ We have $\Delta_{\mathcal{P}}(\rho,k) = (1-e^{-\rho})\min\left(1, \frac{e^{\rho}}{k+1}\right)$ and $\Delta_{\mathcal{P}}(s) = \frac{0.3056}{s^2}$ where $\rho = \mu_{N,n}(t)$ and $s^2 = \sigma^2_{N,n}(t)$ are the mean and the variance of the rv $M_{N,n}(t)$ respectively. In the following we perform a worst case analysis when $M_{N,n}(t) \leq B$ (see (2.5)) with $B > 1$ is the cache size.
Assuming the mean ρ and the variance s^2 are known:

1. if $\rho < 1$, then $s^2 < 1$ by (2.7) and the bound $\Delta_{\mathcal{P}}(\rho,B)$ is tighter than $\Delta_{\mathcal{N}}(s)$ since

$$\sup_{\rho}\left(\Delta_{\mathcal{P}}(\rho,B)\right) < 1-e^{-1} = 0.9502 \text{ and } \Delta_{\mathcal{N}}(s) > 0.3056$$

2. otherwise, the bound $\Delta_{\mathcal{N}}(s)$ is preferable to $\Delta_{\mathcal{P}}(\rho,k)$.

The result in Proposition 2.4 generalizes that in [41, Proposition 1] which states that the Gaussian approximation exists if $\sigma^2_{N,n}(t) \to \infty$ as N goes to infinity. The following remarks explain why the Gaussian approximation [41, Proposition 1] and the *CTA* [23] work well.

Remark 2.1 (Gaussian CDF [41]) *As* $\sigma_{N,n}(t) > 5$ *becomes larger but finite, the factor* $\gamma_{N,n}(t)$ *in* (2.14) *vanishes thanks to* (2.9) *and the CDF* $P(M_{N,n}(t) \leq k)$ *converges to a Gaussian distribution* $\Phi\left(\frac{k+0.5-\mu_{N,n}(t)}{\sigma_{N,n}(t)}\right)$.

Remark 2.2 (Step Function [41]) *According to* (2.7) *if* $\sigma_{N,n}(t)$ *is large, so is the mean* $\mu_{N,n}(t)$ *and the scv* $c^2_{N,n}(t)$ *vanishes by* (2.8). *Therefore the CDF* $P(M_{N,n}(t) \leq k)$ *converges also to a degenerate distribution* $\mathbf{1}(\mu_{N,n}(t) = B)$ *and the rv* $M_{N,n}(t)$ *may be approximated by a deterministic function equals to its mean* $\mu_{N,n}(t)$. *Hence, the rv* $T_{B,n}$ *may be approximated by a constant* $t_{B,n}$.

Propositions 2.3 and 2.4, Remarks 2.1 and 2.2 characterize the asymptotic behaviors of the rv $M_{N,n}(t)$ when $\mu_{N,n}(t)$ is finite or equivalently when $M_{N,n}(t)$ converges (by Proposition 2.2). Now we will investigate the case when $\mu_{N,n}(t)$ diverges.

Proposition 2.5 (Degenerate CDF) *As* N *goes to infinity, if the series* $\mu_{N,n}(t)$ *diverge then* $M_{N,n}(t)$ *does not converge almost surely and it behaves as a deterministic function which is its mean* $\mu_{N,n}(t)$. *Moreover, the TTL* $T_{B,n}$ *may be approximated by a constant* $t_{B,n}$ *which is the unique solution of the following equation*

$$\mu_{N,n}(t_{B,n}) = B. \tag{2.15}$$

Proof The proof of this proposition follows by first applying the convergence result in Proposition 2.2. Then thanks to (2.8) the *scv* $c^2_{N,n}(t)$ vanishes as $\mu_{N,n}(t)$ diverges. The solution of (2.15) is unique because $\mu_{N,n}(t)$ is an increasing function (2.6) (sum of positive and increasing CDFs $\{p_j(t), j = 1, \ldots, N : j \neq n\}$). ∎

Our Proposition 2.5 generalizes [41, Proposition 2] since that our condition $\mu_{N,n}(t)$ diverges is weaker and more general than the condition $\sigma_{N,n}(t)$ diverges needed in [41, Proposition 1]. Also, Proposition 2.5 and Remark 2.2 explain why $M_{N,n}(t)$ and thus $T_{B,n}$ may be approximated by deterministic quantities; this was not clear in [41].

Since N is very large but finite in practice, $\mu_{N,n}(t)$ may be finite and large enough so that (2.15) holds. Its solution $t_{B,n}$ is the *deterministic/constant approximation of the characteristic time* first observed in experiments on LRU caches fed by Poisson request processes [23] and later with renewal request processes [72]. We also provide later in Section 2.7 approximations of the limit of $\mu_{N,n}(t)$ such that $T_{B,n}$ may be approximated by the solution of the equation $\mu_{N,n}(t_{B,n}) = B$ under Poisson request processes.

2.5.2 Random Replacement policy

The Random replacement (RND) policy adds files in the cache without any ordering and removes one file selected uniformly at random from the cache when a cache miss occurs and room is needed. This policy has the benefit to have a low complexity [79, 42].

Assume wlog that the request of file n which occurs at time instant $t_0^n = 0$ yields a cache miss (i.e. $m_0 = t_0^n$). File n enters in the cache and can be only evicted if another cache miss occurs later at each time instant $\{m_i^{\bar{n}}\}_{i\geq 1}$ with probability $1/B$.
We denote by $M_{N,n}$ the number of cache misses that yields the eviction of file n from the cache. It is obvious that:

$$M_{N,n} \stackrel{d}{=} \texttt{GEO}(1/B). \tag{2.16}$$

Hence, the TTL value $T_{B,n}$ which is the sojourn time of file n in the cache is given as follows

$$T_{B,n} := m_1^{\bar{n}} + \sum_{i=1}^{M_{N,n}-1} \vartheta_{i+1}^{\bar{n}}. \tag{2.17}$$

Since the process $\mathcal{M}_{\bar{n}}$ is a stationary process, the inter-miss times $\{\vartheta_i^{\bar{n}},\ i \geq 1\}$ form a stationary sequence. Therefore, $\{\vartheta_i^{\bar{n}},\ i \geq 1\}$ are also identically distributed; moreover, they are independent realizations of the same random variable $\vartheta^{\bar{n}}$ since the minimum of $\{M_{N,j}, j \neq n\}$ is geometrically distributed (a cache miss on a file different from n occurs when this minimum counter fires). For finite cache capacity B and finite catalog size N, the CDF of $T_{B,n}$ is given as follows.

Proposition 2.6 (CDF of $T_{B,n}$) *The exact distribution of $T_{B,n}$ is given by*

$$P(T_{B,n} < t) = \sum_{i=1}^{\infty} 1/B(1-1/B)^{i-1} \times H \star G^{(i-1)}(t) \tag{2.18}$$

where $H(t) = P(m_1^{\bar{n}} < t)$, $G(t) = P(\vartheta_1^{\bar{n}} < t)$, $\star$ denotes the convolution and $G^{(i)}$ is the i-th fold convolution of the function G with itself (by Convention $G^0 \equiv 1$).

The proof of Proposition 2.6 is straightforward since the CDF of $T_{B,n}$ defined as the sum of i.i.d. random variables (cf. (2.17)) is a convolution. However, the interesting part about the formulations (2.17) and (2.18) is the following convergence result proved in [89] and also known as *Refined Rényi's Theorem* [75, Theorem 2].

Proposition 2.7 ("Refined" Exponential CDF) *As the cache size $B \gg 1$, the CDF $P(T_{B,n} < t)$ may be approximated by a "refined" exponential distribution*

$$P(T_{B,n} < t) = 1 - \exp\left(-\frac{\nu_{\bar{n}}}{B}t\right) + \frac{1}{B}e_1\left(\frac{\nu_{\bar{n}}}{B}t\right) + O\left(\frac{1}{B^2}\right), \tag{2.19}$$

where $\nu_{\bar{n}} = 1/E[\vartheta_1^{\bar{n}}]$ is the aggregated miss rate over all the files but file n and

$$e_1(x) = \left[0.5\frac{E[(\vartheta_1^{\bar{n}})^2]}{(E[\vartheta_1^{\bar{n}}])^2} - \frac{E[m_1^{\bar{n}}]}{E[\vartheta_1^{\bar{n}}]} + \left(1 - 0.5\frac{E[(\vartheta_1^{\bar{n}})^2]}{(E[\vartheta_1^{\bar{n}}])^2}\right)x\right] e^{-x}$$

Remark 2.3 (Exponential limit) *When* $1/B \to 0$ *the CDF of the rv* $T_{B,n}$ *converges to the exponential distribution* $P(T_{B,n} < t) \approx 1 - \exp(-t/t_{B,n})$ *where* $t_{B,n} = \frac{B}{\nu_{\tilde{n}}}$ *as proven in [75, Theorem 1]. The condition* $B \gg 1$ *of Proposition 2.7 is satisfied in most of experiments as we can see in [72] and [41] where* $B \in [10, 10^6]$ *and* $B \in [1, 10^4]$ *respectively.*

2.5.3 First-In First-Out policy

The First-In First-Out (FIFO) policy manages a cache as a list by inserting new files at the top and removing the file at the tail of the list respectively. This policy has the advantage to be fair for all files and becomes of a particular interest in the context of quality of service.

Assume wlog that the request of file n which occurs at time instant $t_0^n = 0$ yields a cache miss (i.e. $m_0 = t_0^n$). File n enters in the cache at time t_0^n and will be evicted when the B-th cache miss will occur. If $M_{N,n}$ denotes the number of cache misses that yields the eviction of file n from the cache, then it is obvious that:

$$M_{N,n} \overset{d}{=} B. \tag{2.20}$$

Hence, the TTL value $T_{B,n}$ which is the sojourn time of file n in the cache is given as follows

$$T_{B,n} := m_1^{\tilde{n}} + \sum_{i=1}^{B-1} \vartheta_{i+1}^{\tilde{n}}. \tag{2.21}$$

Defining $\tilde{B}_K = \sum_{k=1}^{K} \xi_k$ where ξ_k is an exponentially distributed rv with mean B/K, we have

$$B = \lim_{K\to\infty} \tilde{B}_K \quad \text{and} \quad B_K = \lfloor \tilde{B}_K \rfloor \le \tilde{B}_K < \lfloor \tilde{B}_K \rfloor + 1.$$

Since B $(= \lim_{K\uparrow\infty} \tilde{B}_K)$ is an integer which is bounded by two consecutive integers B_∞ and $B_\infty + 1$, it follows that

$$B = \lim_{K\to\infty} \tilde{B}_K = \lim_{K\to\infty} B_K \quad \text{where} \quad B_K = \left\lfloor \sum_{k=1}^{K} \xi_k \right\rfloor = \left\{ \sum_{k=1}^{K} \lfloor \xi_k \rfloor \quad \text{or} \quad \sum_{k=1}^{K} \lfloor \xi_k \rfloor + 1 \right\}.$$

Assuming wlog that $B_K = \sum_{k=1}^{K} \lfloor \xi_k \rfloor$, the random variable $T_{B,n}$ can be written as following

$$T_{B,n} = \lim_{K\to\infty} T_{B,n}^{(K)}; \quad T_{B,n}^{(K)} = m_1^{\tilde{n}} + \sum_{i=1}^{B_K - 1} \vartheta_{i+1}^{\tilde{n}}.$$

Replacing B_K in the latter equation, we obtain

$$T_{B,n}^{(K)} = m_1^{\tilde{n}} + \sum_{i=1}^{\lfloor \xi_1 \rfloor - 1} \vartheta_{i+1}^{\tilde{n}} + \cdots + \sum_{i=\lfloor \xi_1 \rfloor + \cdots + \lfloor \xi_{K-1} \rfloor}^{\lfloor \xi_1 \rfloor + \cdots + \lfloor \xi_{K-1} \rfloor + \lfloor \xi_K \rfloor - 1} \vartheta_{i+1}^{\tilde{n}}.$$

Hence, (2.21) becomes

$$T_{B,n} = \lim_{K\to\infty} T_{B,n}^{(K)} = \lim_{K\to\infty} \sum_{k=1}^{K} T_{B,n,k} \tag{2.22}$$

where

$$T_{B,n,1} = m_1^{\tilde{n}} + \sum_{i=1}^{\lfloor \xi_1 \rfloor - 1} \vartheta_{i+1}^{\tilde{n}} \quad \text{and} \quad T_{B,n,k} = \sum_{i=1}^{\lfloor \xi_k \rfloor - 1} \vartheta_{i+1}^{\tilde{n}},\ 2 \le k \le K.$$

Since the rvs $\{\lfloor \xi_k \rfloor\}_{k\ge 1}$ are geometrically distributed with parameter $1 - e^{-K/B}$, the rv $T_{B,n,1}$ is identically distributed as the TTL of a RND cache of size $1/(1-e^{-K/B})$ (See Proposition 2.6) and the distribution of the rv $T_{B,n,k},\ k \ge 2$ is obtained by applying Proposition 2.6 with $H(t) = 1$ (or equivalently [75, Corollary of Theorem 2]).
The exact distribution of $T_{B,n}$ is cumbersome; however, each of the rvs $T_{B,n,k}, k \ge 1$ converges in distribution to an exponentially distributed random variable with rate $\nu_{\tilde{n}}(1 - e^{-K/B}) \approx \frac{\nu_{\tilde{n}} K}{B}$ when B is large enough i.e. $B \gg K$ for fixed value of K (See Remark 2.3). The next proposition gives the asymptotic distribution of $T_{B,n}$ when the cache size B and the catalog size N are large.

Proposition 2.8 (Degenerate CDF) *As the cache size* B *and the catalog size* N *become large, the CDF* $P(T_{B,n} < t)$ *may be approximated by a degenerate distribution*

$$P(T_{B,n} < t) \approx \mathbf{1}\{t > t_{B,n}\}, \quad \textit{where} \quad t_{B,n} = \frac{B}{\nu_{\tilde{n}}}. \tag{2.23}$$

Proof The proof of this proposition is done as follows. Fix $K > 1$. As B (and N, since $N > B$) becomes large i.e. $B \gg K$, the rv $T_{B,n}^{(K)}$ may be approximated by an Erlang random variable of K exponential stages each of mean $\frac{B}{\nu_{\tilde{n}} K}$. This result follows from the fact that $T_{B,n}^{(K)}$ is the sum of K random variables $\{T_{B,n,k}, k = 1, \ldots, K\}$ where each of these variables, say $T_{B,n,k}$, converges in distribution to an exponential random variable with mean $\frac{B}{\nu_{\tilde{n}} K}$ when $B \gg K$. Then, letting K go to infinity, $T_{B,n}^{(K)}$ converges in distribution to a deterministic value equals to $\frac{B}{\nu_{\tilde{n}}}$. Hence, $T_{B,n} = \lim_{K\to\infty} T_{B,n}^{(K)} \approx \frac{B}{\nu_{\tilde{n}}}$. The proof of Proposition 2.8 is then complete. ∎

Remark 2.4 (TTL of RND and FIFO caches) *It is interesting to note that the TTLs of RND and FIFO caches of identical capacity* B *are approximately equal in expectation under the same traffic conditions. We verify this claim when* $\{\mathcal{R}_n\}_n$ *are Poisson Processes or Interrupted Poisson Processes (i.e. a renewal processes with hyper-exponentially distributed inter-request times). This observation will have a significant impact on the complexity of approximate models introduced in Chapter 5.*

2.6 TTL-based caches and performance metrics

In the previous section, we characterize the Time-To-Live (TTL) of LRU, RND and FIFO caches. Now we will present the TTL-based model of each of the latter policies. The main contribution of this section is how we use each of these TTL-based models to calculate the cache performance metrics and describe the miss processes.

2.6.1 Properties of TTL-based models

Infinite capacity and files decoupling. The characterization of the TTL $T_{B,n}$ obtained in Section 2.5 provides interesting descriptions of LRU, FIFO or RND caches. They can be seen as *expiration-based caches* with *infinite capacity* where the *eviction* of file n from the cache occurs only when its TTL $T_{B,n}$ expires. Hence, the main advantage of this description is that *files can be decoupled* and *studied separately* as shown in Figure 2.1. The analysis carried out for a single file will apply for others; only the parameters will change with respect to the file.

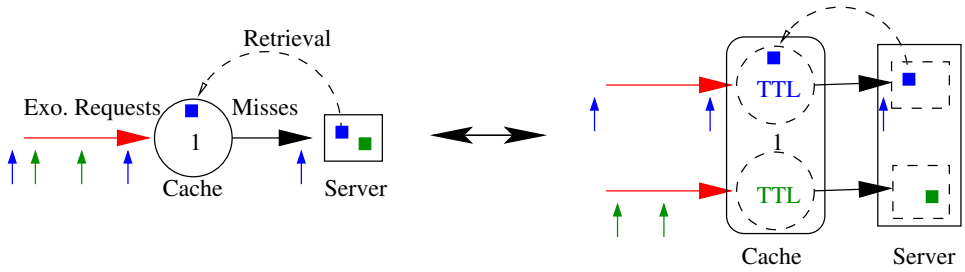


Figure 2.1: Infinite cache capacity and TTL decoupling effect.

Per-file state behavior. Since we can focus on a single file, the description of the miss process on this file is greatly simplified as we can see in Figures 2.3 and 2.2. The TTL-based model of FIFO caches initializes the timer of a file only when a cache miss occurs on that file. Meanwhile, the model of LRU caches draws the TTL of a file at each cache miss instant, but also when a hit occurs on the file.
Since RND caches may be asymptotically described by exponentially distributed TTL-based caches, it is interesting to mention that they can be studied using both TTL-based models depicted in Figures 2.3 and 2.2. This is a direct consequence of the memory-less property of the exponential distribution. More precisely, the remaining TTL at cache hit instants has the same asymptotic exponential distribution as the initial TTL set at the miss instant.

Proposal of taxonomy for TTL-based caches. The TTL-based behaviors shown in Figures 2.3 and 2.2 are not inherent properties of FIFO/RND and LRU caches respectively. They have been

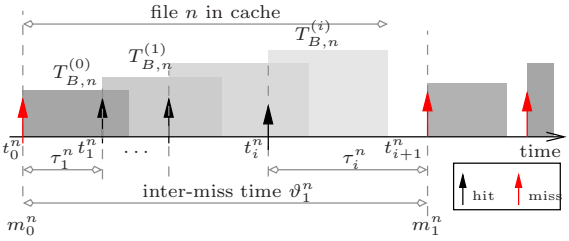


Figure 2.2: TTL-based model of a single file in a single LRU cache, request instants $\{t_i^n\}$, timers $\{T_{B,n}^{(i)}\}$ and miss instants $\{m_i^n\}$.

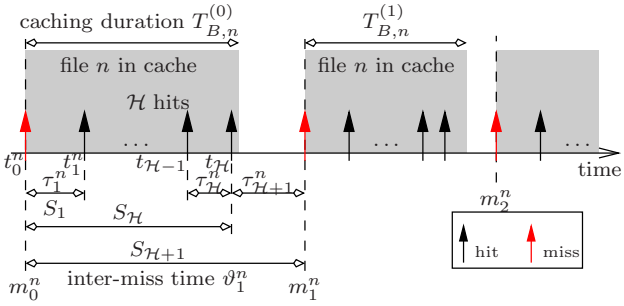


Figure 2.3: TTL-based model of a single file in a single FIFO cache, request instants $\{t_i^n\}$, timers $\{T_{B,n}^{(i)}\}$ and miss instants $\{m_i^n\}$.

described also on traditional and modern DNS caches by [59] and [19] respectively. Moreover, if we allow the rv $T_{B,n}$ to have a general distribution, we define a *TTL-based cache*.
Until now, we have identified two classes of TTL-based caches: TTL-non-renewing caches (see Figure 2.3) and TTL-renewing caches (see Figure 2.2).

Remark 2.5 (Geiger Counters) *We note that the two classes of TTL-based caches coincide with the well-known Geiger counters of Type I and Type II [22, Sect. 10.e and Sect. 10.f, pp. 179-183].*

2.6.2 Single TTL-based cache analysis

For the sake of readability, we first introduce our main assumptions and our notation for the simple architecture of a single TTL-based cache and a server connected in tandem, as shown in Figure 2.1. The terminology and the formalism introduced here will be extended later to general TTL-based cache networks in Chapter 3. From now on the words “cache” and “node” will be used interchangeably. Also, a cache will always be a TTL-based cache unless otherwise specified. We now introduce a key assumption for our approach:

Assumption 2.1 (Infinite capacity) *The TTL-based cache has an infinite capacity.*

A consequence is that content items are evicted from the cache only when their TTL expires and not because space is needed to allocate other contents. Assumption 2.1 allows us to decouple the management of the different content items and study each of them separately as illustrated in Figure 2.1. For this reason, in what follows we will refer to a *single* content item or file n. The effect of capacity constraint is considered latter in this section.
In order to keep the model as simple as possible we also assume that file n processing and transfer times are negligible:

Assumption 2.2 (Zero delay) *There is a zero processing time at each node and a zero transmission delay between nodes including the server.*

In fact, the model presented in this section can be easily extended to consider *non-zero processing time* and/or *delay*. This latter case will be investigated in *Chapter 7*.
For the time being we assume the following minimal assumptions:

Assumption 2.3 (Stationary arrivals and TTLs) *The point process $\mathcal{R}_n = \{t_i^n, i \in \mathbb{Z}\}$ is simple (i.e. there are no simultaneous requests), stationary (i.e. $\{\tau_i^n, i \in \mathbb{Z}\}$ is a stationary sequence), and independent of the sequence of TTLs $\{T_{B,n}^{(i)}, i \in \mathbb{Z}\}$ which is also assumed to be stationary. Furthermore, the intensity $\lambda_n := 1/E[\tau_1^n]$ of the point process $\mathcal{R}_n$ is non-zero and finite, its associated counting process $\{\mathcal{N}(t), t \geq 0\}$ is stationary, and $0 < 1/\mu_n := E[T_{B,n}^{(1)}] < \infty$.*

Under Assumption 2.3 the cache is in steady-state (in particular) at time $t = 0$ and from now on we will only observe its behavior at times $t \geq 0$. We denote by $F_n(t) = P(\tau_i^n < t)$ and $T_n(t) = P(T_{B,n}^{(i)} < t)$ the CDFs of τ_i^n and $T_{B,n}^{(i)}$, respectively.

From now on we assume that each cache satisfies Assumptions 2.1, 2.2 and 2.3.

Requests for a specific file n are generated at times $\{t_i^n, i \in \mathbb{Z}\}$ such that $\ldots < t_{-1}^n < t_0^n \leq 0 < t_1^n < \ldots$ by convention, where $\mathbb{Z}$ denotes the set of all integers. We recall that $\tau_i^n = t_{i+1}^n - t_i^n$ is the inter-arrival time between i-th and $i+1$-st requests. Also, let $T_{B,n}^{(i)}$ ($i \in Z$) being the TTL duration generated for the content after the arrival of the request at time t_i^n.

Remark 2.6 (Multiple sources of requests) *A cache may be fed by independent stationary streams of requests. The resulting process is also a stationary process and the CDF $F_n(t) = P^0(\tau_1^n < t)$ of the first inter-arrival times is given under its Palm probability by [10, Sect. 1.4.2, Eq. 1.4.6].*

We will study single cache in its steady-state regime, and we will calculate the following performance metrics:

1. **The hit probability.** Denoted here by $H_{P,n}$ for a file n, it is the stationary probability that the file is in the cache at request instants $\{t_i^n\}_{i>1}$. The **miss probability** $M_{P,n}$ is the complementary probability $1 - H_{P,n}$.

2. **The occupancy.** Denoted here by $O_{P,n}$ for a file n, it is the stationary probability that the file is in the cache at any random instant t.

3. **The miss rate.** $M_{R,n}$ is the rate at which the cache forwards requests to the server.

4. **The sojourn time.** It is defined as the total time that a file stayed in the cache. Let Q_n denote the sojourn time of file n in the cache.

The hit probability is clearly a fundamental performance metric for a caching system. The occupancy probability is equal to the fraction of time that a content spends in the cache and then it can be used to characterize the stationary buffer distribution. For a single cache network the miss rate quantifies the load on the server, but for a hierarchical cache network a miss at one cache causes the request to be forwarded to higher-level caches. Hence, we need to characterize the miss process to be able to evaluate the hit probability at higher-level caches. Finally, the sojourn times of a file in RND, FIFO and LRU caches are expressed as follows:

Remark 2.7 (Sojourn times)

$$\textit{For RND caches,}\quad Q_n = T_{B,n}\ ;$$

$$\textit{For FIFO caches,}\quad Q_n = T_{B,n}\ ;$$

$$\textit{For LRU caches,}\quad Q_n = T_{B,n}\ \mathbf{1}(T_{B,n} > \tau_1^n) + (\tau_1^n + Q_n)\ \mathbf{1}(T_{B,n} < \tau_1^n).$$

The quantities (occupancy, hit probability and sojourn time) of a file n are related by a classical result of Queueing Theory (see Proposition 2.9).

We recall that the *miss process* is the sequence of successive time instants $0 \le m_0^n < m_1^n < \cdots$ at which misses occur in $[0, \infty)$, which are also the times at which the server forwards a copy of file n to the cache. We denote by $\vartheta_i^n = m_{i+1}^n - m_i^n$ the time interval between the i-th and the $(i+1)$-st misses for $i \ge 0$ and $G_n(t) = P(\vartheta_i^n < t)$ the CDF of ϑ_i^n. Stronger statistical assumptions on the sequences $\{\tau_i^n, k \in \mathbb{Z}\}$ and $\{T_{B,n}^{(i)}, k \in \mathbb{Z}\}$ will quickly become necessary only for the purpose of characterizing the miss process of the cache—cf. Chapter 3 and Chapter 4.

Remark 2.8 (On the miss process) *Under Assumption 2.3 the sequence $\{t_i^n, Q_{n,i}\}_{i\ge 0}$, where the random variables $\{Q_{n,i}, i \ge 0\}$ are successive realizations of the sojourn time Q_n, defines a stationary marked point process [10, Sect. 1.1.3] which corresponds to the miss process $\mathcal{M}_n$. Note also that miss instants are regenerative points for the cache; hence, the analysis of the cache can be carried out within an inter-miss time interval (or between two consecutive regenerative points).*

Table 2.1: Notation for a single-cache network

λ_n	Request arrival rate of file n (single cache)
$1/\mu_n$	Expected TTL of file n (single cache)
$F_n(t)$	CDF exogenous arrivals of file n (single cache)
$G_n(t)$	CDF inter-miss times of file n(single cache)
$T_n(t)$	CDF TTL duration of file n (single cache)
$Z^*(s)$	LST of CDF $Z(t)$
$H_{P,n}, M_{P,n}$	Hit, miss probability resp. of file n (single cache)
$H_{R,n}, M_{R,n}$	Hit, miss rate resp. of file n(single cache)
$O_{P,n}$	Occupancy probability of file n (single cache)

The next proposition relates the hit probability, the occupancy and the sojourn time of a file n in any cache running a replacement algorithm.

Proposition 2.9 (Mean-Value Formula or Generalized Little's Law for Caches)

$$O_{P,n} = \lambda_n \times (1 - H_{P,n}) \times \bar{Q}_n \tag{2.24}$$

where $\lambda_n = \frac{1}{E[\tau_1^n]}$ *is the intensity of the stationary request process* $\mathcal{R}_n$*,* $H_{P,n}$ *and* $\bar{Q}_n = E^0[Q_n]$ *are functions of parameters of the TTL distribution.*

Proof The proof of this proposition is done by applying the *Mean-Value Formulas* in [10, Formula (1.3.2)] in two steps: (i) substitute $Z(t) = \mathbf{1}(T_{B,n}^{(0)} > t)$, $T_1 = \vartheta_1^n$, $g(z) = \mathbf{1}(z > 0)$ and the integral in the expectation by Q_n, and (ii) replace the intensity $(E^0[\vartheta_1^n])^{-1}$ of the miss process $\mathcal{M}_n$ i.e. the miss rate by $(1 - H_{P,n}) \times (E^0[\tau_1^n])^{-1}$. ∎

Regardless the caching policy, the hit probability $H_{P,n}$ and the occupancy probability $O_{P,n}$ differ in general. They are equal if the arrival process $\mathcal{R}_n = \{t_i^n, i \in \mathbb{Z}\}$ is a Poisson process thanks to the *PASTA* property. This result was first used in [41, Sect. 6] for RND caches fed by Poisson request processes. Our Proposition 2.9 is valid for all cache policies where request traffic is modeled by stationary point processes.

Performance metrics of TTL-renewing caches. Consider the request submitted at time t_0^n (the process for requests submitted at times t_i^n with $i \neq 0$ is the same). There is a *cache hit* (resp. *cache miss*) at time t_0^n if file n is present (resp. is not present) in the cache at this time, which corresponds to the situation where $t_0^n \leq t_{-1}^n + T_{B,n}^{(-1)}$ (resp. $t_0^n > t_{-1}^n + T_{B,n}^{(-1)}$). In the case of a cache miss the request is *instantaneously* (because of Assumption 2.2) forwarded to the server at time $m_0^n = t_0^n$ and file n is retrieved from the server. By convention, file n is permanently stored

in the server. Once file n is fetched from the server, a copy of it is *instantaneously* transmitted to the cache and the request is resolved at time t_0^n, while a copy is kept at the cache. At time t_0^n the TTL of file n is set to $T_{B,n}^{(0)}$ both for a cache hit and for a cache miss. The next cache miss after time m_0^n will occur at time $m_1^n = t_j^n$ with $j = \min\{l > 0 : t_l^n > t_{l-1}^n + T_{B,n}^{(l-1)}\}$—see Figure 2.2 where $j = i + 1$.

Resetting the TTL tends to increase the total time spent in cache and the hit probability on the cache especially for the most popular contents. This corresponds to the general objective to move popular documents as close as possible to the users.

Before moving to the analysis of the single cache, we need to introduce more notation. For any non-negative random variable (rv) ξ with a CDF $F(t) = P(\xi < t)$ $(\forall\, t \geq 0)$, we denote by

$$F^*(s) = E[e^{-s\xi}] = \int_0^\infty e^{-st} F(dt) = \int_0^\infty e^{-st} dF(t), \quad s \geq 0$$

the Laplace-Stieltjes Transform (LST) of ξ. The notation $F(dt)$ is used because the Probability Density Function (PDF) $f(t)$ of the rv ξ may not exist; otherwise, $F(dt)$ is replaced by $f(t)dt$ as commonly seen. For any number $a \in [0, 1]$, $\bar{a} := 1 - a$ by definition.

Define $L_n(t) := P(\tau_i^n < t, \tau_i^n < T_{B,n}^{(i)})$ the stationary probability that the inter-arrival time between two successive requests is smaller than t and smaller than the TTL associated with the former request. Because arrivals and TTLs are independent we have

$$L_n(t) = \int_0^t (1 - T_n(x)) dF_n(x), \; t \geq 0. \tag{2.25}$$

Using the notation in Table 2.1, the following proposition provides exact formulas for two of the performance metrics of interest.

Proposition 2.10 (Hit probability and miss rate of TTL-renewing caches) *Under Assumption 2.3 the (stationary) hit probability $H_{P,n}$ and the (stationary) miss rate $M_{R,n}$, are respectively given by*

$$H_{P,n} = \int_0^\infty (1 - T_n(x)) dF_n(x) = L_n(\infty), \tag{2.26}$$

$$M_{R,n} = \lambda_n (1 - H_{P,n}) \tag{2.27}$$

where $\lambda_n = 1/E[\tau_1^n]$ is the request arrival rate of file n.

Proof The stationary hit probability $H_{P,n}$ is defined as the probability that an arriving request finds file n in the cache, i.e. the TTL has not expired yet, namely,

$$H_{P,n} = P(\tau_i^n \leq T_{B,n}^{(i)}) = \int_0^\infty P(x \leq T_{B,n}^{(i)}) dF_n(x) = \int_0^\infty (1 - T_n(x)) dF_n(x).$$

The stationary miss probability is $M_{P,n} = 1 - H_{P,n}$ so that the miss rate is given by (2.27). ∎

The next result provides a closed-form formula for the *cache occupancy* $O_{P,n}$.

Proposition 2.11 (Occupancy probability of TTL-renewing caches) *Under Assumption 2.3 the stationary cache occupancy* $O_{P,n}$ *is given by*

$$O_{P,n} = \lambda_n \int_0^\infty (1 - T_n(t))(1 - F_n(t))dt. \tag{2.28}$$

Proof Let $\chi(t) \in \{0, 1\}$ be the cache occupancy at time t with $\chi(t) = 1$ if file n is in the cache at time t and $\chi(t) = 0$ otherwise. Since the cache is in steady-state at time $t = 0$ under Assumption 2.3, we have $O_{P,n} = E[\chi(0)] = P(\chi(0) = 1)$.

Letting $Z(t) = \chi(t)$, $T_1 = \tau_1^n$ and $g(z) = \mathbf{1}(z > 0)$ in [10, Formula (1.3.2), p. 21] yields

$$O_{P,n} = \lambda_n E^0 \left[\int_0^{\tau_1^n} \chi(t)dt \right]$$

with E^0 the expectation operator under the Palm probability P^0 of the stationary point process $\{t_n, n \in \mathbb{Z}\}$ with associated marks $\{T_{B,n}^{(i)}, k \in \mathbb{Z}\}$. P^0 has the property that $P^0(t_0 = 0) = 1$ (see [10, Definition (1.2.1), p. 14]) which implies $\chi(t) = \mathbf{1}(t < T_{B,n}^{(0)})$ for $t \in [0, \tau_1^n]$ under P^0. Hence,

$$\begin{aligned}
O_{P,n} &= \lambda_n E^0 \left[\int_0^{\tau_1^n} \mathbf{1}(T_{B,n}^{(0)} > t)dt \right] \\
&= \lambda_n \int_0^\infty E^0 \left[\int_0^x \mathbf{1}(T_{B,n}^{(0)} > t)dt \right] dF_n(x) \qquad (2.29) \\
&= \lambda_n \int_0^\infty \left(\int_0^x (1 - T_n(t))dt \right) dF_n(x) \\
&= \lambda_n \int_0^\infty (1 - T_n(t)) \left(\int_t^\infty dF_n(x) \right) dt \\
&= \lambda_n \int_0^\infty (1 - T_n(t))(1 - F_n(t))dt
\end{aligned}$$

where (2.29) is obtained by conditioning on the rv τ_1^n with CDF $F_n(t)$ and by using the independence of the rvs τ_1^n and $T_{B,n}^{(0)}$. This completes the proof. ∎

Performance metrics of TTL-non-renewing caches We consider the request submitted at time t_0^n and we assume that file n is not in the cache. There is a *cache hit* (resp. *cache miss*) at time t_i^n if file n is present (resp. is not present) in the cache at this time, which corresponds to the situation where $t_i^n - t_0^n \leq T_{B,n}^{(0)}$ (resp. $t_i^n - t_0^n > T_{B,n}^{(0)}$). In the case of a cache miss the request is *instantaneously* (because of Assumption 2.2) forwarded to the server at time $m_0^n = t_0^n$ and file n is retrieved from the server. By convention, file n is permanently store in the server. Once file n is fetched from the server, a copy of it is *instantaneously* transmitted to the cache and the request is resolved at time t_0^n, while a copy is kept at the cache. At time t_0^n the TTL of file n is set to $T_{B,n}^{(0)}$ for a cache miss. The next cache miss after time m_0^n will occur at time $m_1^n = t_j^n$ with

$j = \min\{l > 0 : t_l^n - t_0^n > T_{B,n}^{(0)} \geq t_{l-1}^n - t_0^n\}$—see Figure 2.3 where $j = \mathcal{H} + 1$ and $\mathcal{H} \stackrel{d}{=} \mathcal{N}(T_{B,n}^{(0)})$ is the number of hits on file n.

Proposition 2.12 (Hit and occupancy probabilities of TTL-non-renewing caches) *Under Assumption 2.3, the stationary hit probability* $H_{P,n}$ *and the stationary occupancy probability* $O_{P,n}$ *are given by:*

$$H_{P,n} = 1 - \left(1 + E^0\left[\mathcal{N}(T_{B,n}^{(0)})\right]\right)^{-1} \quad (2.30)$$

$$O_{P,n} = \lambda_n \times \mu_n^{-1}(1 - H_{P,n}) \quad (2.31)$$

Proof Under Assumption 2.3 the miss process is also a stationary process; hence, we can compute the hit probability within a stationary inter-miss time. The expected number of requests with an inter-miss time is $E^0\left[\mathcal{N}(T_{B,n}^{(0)})\right] + 1$. Therefore, the fraction of requests that hit on the cache is $E^0\left[\mathcal{N}(T_{B,n}^{(0)})\right] / \left(E^0\left[\mathcal{N}(T_{B,n}^{(0)})\right] + 1\right)$. And (2.30) follows. The proof of (2.31) follows from Proposition 2.9 and Remark 2.7 where $\bar{Q}_n = E^0[T_{B,n}^{(0)}] = \mu_n^{-1}$. ∎

2.6.3 Accounting for finite cache capacity

Case of LRU, RND and FIFO caches: Characteristic Time Approximation (CTA) The CTA was initially obtained on LRU caches fed by Poisson request processes as follows. First, Che et *al.* [23] approximate $T_{B,n}$ by a constant $t_{B,n}$ (See Remark 2.2). Then, they consider that $t_{B,n} = t_B, \forall\, n$ i.e. the TTL is identical for all files. Their experiments and the ones conducted in [41, 66] confirmed the accuracy of these approximations on LRU caches.

Fricker et *al.* [41] extended the CTA to RND caches by noting that the expected sojourn time (or equivalently the TTL) of a file in RND cache is independent to the file selected i.e. $E[T_{B,n}] \approx t_B,\ \forall n$. They intuitively explained that the approximation "$E[T_{B,n}] \approx t_B,\ \forall n$" is correct if the intensity λ_n of the request process $\mathcal{R}_n$ of file n is negligible in comparison to the overall intensity $\lambda = \sum_n \lambda_n$ of the aggregated request process $\mathcal{R} = \bigotimes_{n=1}^{N} \mathcal{R}_n$.

Martina et *al.* [72] also generalized the CTA to other replacement policies when $\{\mathcal{R}_n\}_n$ are renewal processes. They experimentally showed that the CTA should be stated as $E[T_{B,n}] = t_{B,n} \approx t_B,\ \forall n$ when N is large and they observed that the TTL of a RND cache may be approximated by an exponentially distributed random variable. This observation is very important especially when we describe the miss processes.

In order to provide new insights to the validity of the CTA, we will keep the dependency of the rv $T_{B,n}$ for each file n. Thanks to our definition of the TTL of a given cache, we experimentally show that the rvs $\{T_{B,n}, n = 1, \ldots, N\}$ can be approximated by a single random variable T_B (See Figure 2.4). $\{\mathcal{R}_n\}_n$ are assumed to be Poisson processes with rates modulated

by a Zipf law of parameter $\alpha = 0.65$ and the total request rate equals 1. Requests are generated from a catalog of size $N = 10^4$ and the cache capacity is set to $B = 100$. Figure 2.4 shows the CDFs of the characteristic time of files of popularity rank $n = 1, 10, 100, 1000$. As already proved by our asymptotic analysis in Section 2.5, we observed that $T_{B,n}$ may be safely approximated by a deterministic value for LRU or FIFO caches, and an exponentially distributed random variable for RND caches.
Having this extension of the CTA in hand, we derived new results on the bound of the expected value of the characteristic time $E[T_{B,n}]$. These bounds hold for all cache policies when applying the CTA (i.e. when we approximate $E[T_{B,n}] = t_{B,n}$ by the same constant t_B).

Proposition 2.13 (General inequalities) *When* $t_{B,n} \approx t_B$, $\forall n$*, the following inequalities hold*

$$t_B \leq \quad \bar{Q}_n \quad , \forall\, n \tag{2.32}$$

$$\lambda_n(1 - H_{P,n})t_B \leq \quad O_{P,n} \quad \leq \lambda_n t_B, \ \forall\, n \tag{2.33}$$

$$t_{B,\min} = \frac{B}{\lambda} \leq \quad t_B \quad \leq t_{B,\max} = \frac{B}{\lambda \times \bar{M}_P} \tag{2.34}$$

where $\bar{M}_P = \sum_{n=1}^{N} \frac{\lambda_i}{\lambda}(1 - H_{P,n})$ *denotes the average miss probability of the cache and* $\lambda = \sum_{n=1}^{N} \lambda_n$ *is the aggregate request rate on the cache.*

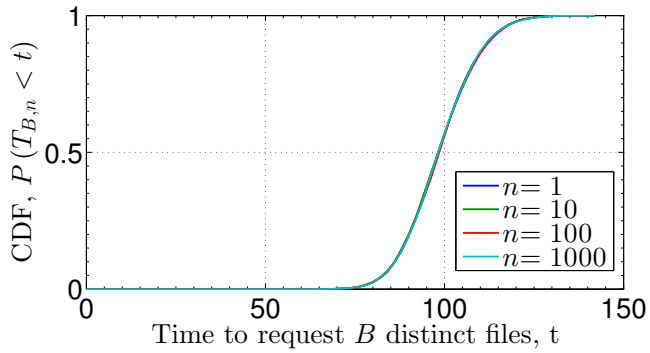
Proof The inequality (2.32) follows directly from the definition of the sojourn time Q_n and the TTL $T_{B,n}$ of file n. If a file sojourns in the cache, it must stay at least for a duration equal to the characteristic time of the cache.
The left part of (2.33) follows from Proposition 2.9 and (2.32). Meanwhile, the right part is easily obtained by noting that $O_{P,n}$ is the expected number of file n in the cache ($O_{P,n} \in [0; 1]$) and $\lambda_n t_B$ is the expected number of requests for file n within the time t_B ($\lambda_n t_B \geq 1$). The upper and lower bounds in (2.34) are obtained by summing (2.33) over all files. ∎

Proposition 2.13 will be of a particular utility, especially the lower bound $t_{B,\min}$ characterized in (2.34), as we will see in *Chapter 5* where we derive very efficient algorithms to approximate performance metrics on cache networks.

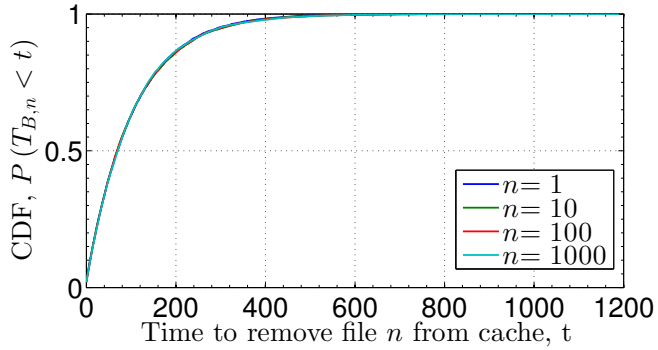
Case of TTL-based caches Proposition 2.9 can be applied with the CTA to define a more general procedure that can accurately approximate the expected TTL value of a cache under finite capacity constraints.

- First, we take the sum of $O_{P,n}$ over all the files n which is equal to the cache size B

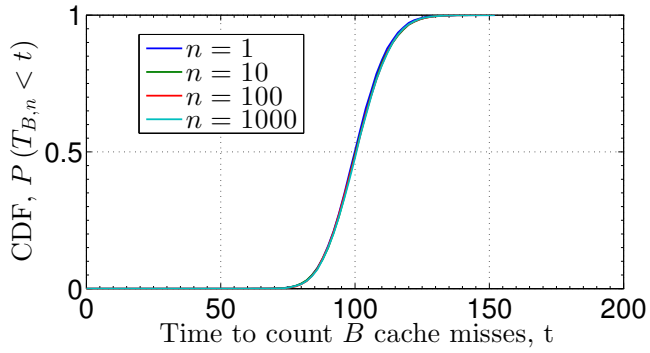
$$\sum_{n=1}^{N} O_{P,n} = B \tag{2.35}$$



(a) LRU



(b) RND



(c) FIFO

Figure 2.4: CDFs of per-file characteristic time (or TTL) of three cache replacement policies.

- Thanks to the CTA, we approximate $T_{B,n}$ by a file-independent random variable i.e.

$$T_{B,n} \approx T_B, \ \forall n.$$

- Finally, we solve the general fixed-point equation:

$$B = \sum_{n=1}^{N} \lambda_n \times (1 - h(n, E[T_B])) \times q(n, E[T_B])$$

where $H_{P,n} \approx h(n, E[T_B])$ and $\bar{Q}_n \approx q(n, E[T_B])$ are functions which depend on $E[T_B]$ and the characteristics of the request processes $\{\mathcal{R}_n\}_n$.

Since (2.35) holds in average, it does not guarantee that no more than B files might have a positive TTL at a given time instant $t > 0$. One should use instead the formulation:

$$P\left(\left|\sum_{n=1}^{N} \mathbf{1}(\tau_t^n < T_{B,n}) - B\right| > \eta\right) < \varepsilon \ , \ \forall \varepsilon, \eta > 0, \tag{2.36}$$

where $\tau_t^n = t - t_j^n$ with $j = \max\{l : t_l^n < t\}$. τ_t^n is also known as the age of the last request for file n at time t. Assuming that $T_{B,n} \stackrel{d}{=} T_B, \ \forall n$, (2.36) reduces to

$$P\left(\left|\sum_{n=1}^{N} \mathbf{1}(\tau_t^n < T_B) - B\right| > \eta\right) < \varepsilon \ , \ \forall \varepsilon, \eta > 0.$$

2.7 Applications and existing experiments revisited

In this section we revisit some experiments on caches. First we recall the main assumptions used for these simulations and then we show that our theoretic result justifies their accuracy. In most of studies on RND and FIFO caches [41, 72] the cache capacity B is large, say $B \in [10^2, 10^6]$. This setting is enough to justify the accuracy of the CTA on RND and FIFO caches thanks to Remark 2.3 and Proposition 2.8 respectively. Therefore, we focus on experiments conducted on LRU caches.

2.7.1 LRU caches under IRM assumption: general results

We consider that $\mathcal{R}_n$ is a Poisson process with rate λ_n. Since the deterministic limit is accurate when $\mu_N(t)$ diverges (see Proposition 2.5), we shall restrict our analysis to the case where $\mu_N(t)$ converges. We give sufficient conditions under which $\mu_N(t)$ converges and we shall provide approximations of the sum $\mu_\infty(t)$ needed to apply the CTA.

Proposition 2.14 (Convergence and Approximation for Poisson processes) *If the aggregated rate $\sum_n \lambda_n$ converges to a value λ, then the mean $\mu_N(t)$ and the variance $\sigma^2_N(t)$ converge as N goes to infinity and for any finite $t > 0$. Moreover, $\mu_N(t)$ is upper bounded by $\lambda \times t$.*
Assuming also that $\lambda_n = \phi(n)$ where $\phi(x)$ is a positive and decreasing function in $x \in [0, +\infty)$, the mean $\mu_N(t)$ can be approximated by $\mu(t)$ with an error bound $0 \leq \mu_\infty(t) - \mu(t) \leq \varepsilon_1(t)$ given as following

$$\mu(t) = \int_1^\infty (1 - e^{-\phi(x)t})dx = \int_0^{\phi(1)t} g(x,t)dx \quad , \quad \varepsilon_1(t) = \int_{\phi(1)t}^{\phi(0)t} g(x,t)dx \tag{2.37}$$

where the function $g(x,t) = (1 - e^{-x})\,(-t^{-1})(\phi^{-1})'(xt^{-1})$ for any finite $t > 0$.

Proof This proposition is proved using the inequality $1 - e^{-xt} \leq tx$ for $t, x \geq 0$ as following

$$0 \leq \sigma^2_N(t) \leq \mu_N(t) \leq t \sum_{n=1}^{N} \lambda_n \longrightarrow_N \lambda t.$$

The second part of this proposition follows from classical results on series-integrals convergence since $\mu_N(t) = \sum_{n=1}^{N}(1 - e^{-\phi(n)t})$ where the inner function $\phi(x)$ is a positive decreasing function in $[0, \infty)$. The function $(\phi^{-1})'(x)$ is the first derivative of the inverse function of $\phi(x)$. ∎

The condition "$\lambda = \sum_n \lambda_n$ converges" of Proposition 2.14 is sufficient to have the convergence of the mean $\mu_N(t)$. This condition simply states that the rate of the aggregated request process $\mathcal{R}$ is **strictly positive** and **finite**; this implies that there is no explosion i.e. a finite number of requests within any finite duration and the aggregated request process $\mathcal{R}$ is ergodic. This condition always holds in practice. Under the conditions of Proposition 2.14, the next corollary gives the approximate values of the total rate λ and the variance $\sigma^2_N(t)$.

Corollary 2.1 (Total rate and variance) *The total rate $\lambda = \sum_n \lambda_n$ is approximated by $\hat{\lambda}$ with an error bound $0 \leq \lambda - \hat{\lambda} \leq \varepsilon_0$ given as following*

$$\hat{\lambda} = \int_1^\infty \phi(x)dx = \int_0^{\phi(1)} (-x)(\phi^{-1})'(x)dx \quad , \quad \varepsilon_0 = \int_0^1 \phi(x)dx \tag{2.38}$$

The variance $\sigma^2_N(t)$ can be approximated by $\sigma^2(t)$ with an error bound $0 \leq \sigma^2_\infty(t) - \sigma^2(t) \leq \varepsilon_2(t)$ given as following

$$\sigma^2(t) = \mu(2t) - \mu(t) \quad , \quad \varepsilon_2(t) = \varepsilon_1(2t) - \varepsilon_1(t) \tag{2.39}$$

where $\mu(t)$ and $\varepsilon_1(t)$ are given in proposition 2.14.

Proof The first part of this corollary follows from classical results on series-integrals convergence. The second part of the corollary is proved using the inequality (2.7) and the equality $\sigma^2_N(t) = \mu_N(2t) - \mu_N(t)$ from [41, Lemma 1]. ∎

The CTA on LRU caches consists in approximating $T_{B,n} \approx t_B$ as a constant which is the root of the equation $\mu_\infty(t_B) = B$. We will instead solve the approximate equation $\mu(t_B) = B$ which is accurate if the cache capacity B is large. This claim is justified by the following corollary.

Corollary 2.2 (Coefficient of variation) *The coefficient of variation* $c_N^2(t)$ *may be approximated by* $c^2(t)$ *and bounded by* $\varepsilon_3(t)$ *given as following*

$$c^2(t) = \frac{\mu(2t) - \mu(t)}{\mu^2(t)} \leq \varepsilon_3(t) = (\mu(t))^{-1} \approx (\hat{\lambda}t)^{-1},\ \forall t \geq 1 \tag{2.40}$$

where $\mu(t)$ *is obtained in Proposition 2.14 and* $\hat{\lambda}$ *is given in Corollary 2.1.*

Proof The approximation $c^2(t)$ of the coefficient of variation $c_N^2(t)$ follows directly from Proposition 2.14, Corollary 2.1 and Equation (2.8); or more precisely, we have

$$c_N^2(t) = \frac{\sigma_N^2(t)}{\mu_N^2(t)} = \frac{\mu_N(2t) - \mu_N(t)}{\mu_N^2(t)} \approx c^2(t) = \frac{\mu(2t) - \mu(t)}{\mu^2(t)} \leq \frac{1}{\mu(t)}$$

since

$$\mu(2t) = \int_1^\infty (1 - e^{-2t\phi(x)})dx = \int_1^\infty (1 - e^{-\phi(x)t})(1 + e^{-\phi(x)t})dx \leq 2\mu(t).$$

The bound $\varepsilon_3(t)$ is a positive and decreasing function given that $\mu(t)$ is an unbounded and strictly increasing function of $t > 0$. Since $\phi(x)$ is positive and decreasing, we have for $\forall t \geq 1$

$$\phi(x)(1 + \phi(1)t)^{-1} \leq \phi(x)(1 + \phi(x)t)^{-1}$$

and it follows from the inequality $x(1+x)^{-1} \leq 1 - e^{-x} \leq x$, $x > 0$ that

$$(1 + \phi(1)t) \times t\int_1^\infty \phi(x)dx \leq m(t) = \int_1^\infty (1 - e^{-\phi(x)t})dx \leq t\int_1^\infty \phi(x)dx.$$

Then,

$$0 \leq \varepsilon_3(t) - (\hat{\lambda}t)^{-1} \leq \frac{\phi(1)}{\hat{\lambda}} = o(1)\ ,\ \hat{\lambda} = \int_1^\infty \phi(x)dx \approx \lambda = \sum_{n\geq 1} \phi(n).$$

■

Clearly, the CTA on LRU caches i.e. $T_{B,n} \approx t_B$ using $\mu(t_B) = B$ instead of $\mu_\infty(t_B) = B$ will be more accurate when the expected number of requests (of the aggregated process $\mathcal{R}$) within t_B is large i.e. $\lambda t_B \gg 1$. A sufficient condition for the latter to hold is to assume a large cache capacity $B \gg 1$. To see this, we rely on (2.34) where $\lambda t_B \geq \lambda t_{B,\min} = B$. We recall that the equation $\mu(t_B) = B$ has to be solved numerically. However, a closed-form expression of the constant t_B can be found in very few cases especially under the conditions of Proposition 2.14.

Corollary 2.3 (Approximation of the characteristic time) *If the request rate of process $\mathcal{R}_n$ can be decomposed as $\lambda_n = \phi(n) = \theta(N)\psi(n/N)$ where $\theta(.)$ and $\psi(.)$ are respectively defined in $[1, +\infty)$ and $[0, 1]$, then*

$$t_B = \Psi^{-1}(B/N)/\theta(N) + o\left(1/\theta(N)\right) \tag{2.41}$$

where the function $\Psi(t) = 1 - \int_0^1 e^{-\psi(x)t}dx$ and $\Psi^{-1}(.)$ is its inverse function.

Proof Note that if the decreasing rate function $\lambda_n = \phi(n) = \theta(N)\psi(n/N)$, then

$$\mu_N(t/\theta(N)) = \sum_{n=1}^{N} 1 - e^{-\lambda_n t/\theta(N)} = N \times \sum_{n=1}^{N} \frac{1 - e^{-\psi(n/N)t}}{N} = N \times \int_0^1 (1 - e^{-\psi(x)t})dx + o(N).$$

By substitution, we obtain $\mu_N(t) = N\ \Psi(t \times \theta(N)) + o(N)$.
Taking $t_B = \Psi^{-1}(B/N)/\theta(N) + o\left(1/\theta(N)\right)$ in this latter equation, we find $\mu_N(t_B) = B$. ∎

This corollary generalizes the results in [41, Proposition 3] established for Poisson processes modulated by a Zipf popularity law. We can easily check that when modulated by Zipf popularity law, request rates λ_n satisfy $\lambda_n = \theta(N)\psi(n/N)$.

2.7.2 LRU caches under Poisson request processes: special popularity laws

In this section we apply our convergence results of Sections 2.5 and 2.7.1 to theoretically explain the validity of the CTA which consists to approximate the expected TTLs of files by a unique constant. We consider that $\{\mathcal{R}_n\}_n$ are Poisson processes with rates λ_n modulated by a file-popularity law.

Zipf-like distribution In this case $\lambda_n = 1/n^\alpha$, $\alpha > 0$. By applying the equivalence principle of series, one can show that the series given by the mean $\mu_N(t)$ and the variance $\sigma_N^2(t)$ are the same nature as the sum $\sum_n (-t)/n^\alpha$. In fact, $p_n(t) = 1 - e^{-t/n^\alpha} \to_n 0$ and $p_n(t)/(-tn^{-\alpha}) \to_n 1$ justify the following equivalences:

$$p_n(t)(1 - p_n(t)) \sim p_n(t) \sim (-t)/n^\alpha.$$

It is well known that the Riemann series $\sum_n 1/n^\alpha$ converges if and only if $\alpha > 1$. Therefore,

- when $\alpha \leq 1$ (this is the case in the experiments done in [23]) the mean $\mu_N(t)$ and variance $\sigma_N^2(t)$ both diverge; thus, the TTL T_B may approximated by a constant t_B solution of the equation $\mu_N(t_B) = B$ thanks to Proposition 2.5.
- if $\alpha > 1$ (this is the case of the fluid analysis in [54]) then the total rate $\lambda = \sum_n \lambda_n$, the mean $\mu_N(t)$ and the variance $\sigma_N^2(t)$ all converge (see Proposition 2.14). Moreover their

limit are bounded using the series-integral convergence by $\mu(t) \leq \mu_\infty(t) \leq \mu(t) + \varepsilon_1(t)$ and $\sigma^2(t) \leq \sigma^2_\infty(t) \leq \sigma^2(t) + \varepsilon_2(t)$ where

$$\begin{aligned} \lambda &= \zeta(\alpha) \\ \mu(t) &= t^{\alpha^{-1}}\gamma(1-\alpha^{-1}, t) - (1-e^{-t}) \qquad (2.42) \\ \sigma^2(t) &= t^{\alpha^{-1}}\left(2^{\alpha^{-1}}\gamma(1-\alpha^{-1}, 2t) - \gamma(1-\alpha^{-1}, t)\right) - e^{-t}(1-e^{-t}), \qquad (2.43) \end{aligned}$$

respectively with the error bounds $\varepsilon_0 = 0$, $\varepsilon_1(t) = 1 - e^{-t} + t^{\alpha^{-1}}\Gamma(1-\alpha^{-1}, t)$ and $\varepsilon_2(t) = \varepsilon_1(2t) - \varepsilon_1(t)$ where $\gamma(s,x)$ and $\Gamma(s,x)$ are the lower and upper incomplete gamma function and $\zeta(x)$ is the zeta Riemann function.

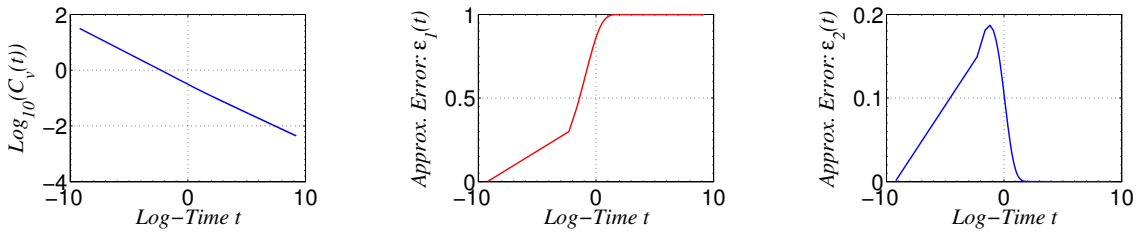


Figure 2.5: Zipf popularity, $\alpha = 1.1$

Figure 2.5 shows $c(t) = \sigma(t)/m(t)$, the errors $\varepsilon_1(t)$ and $\varepsilon_2(t)$. As we can see, the instant $t = 1$ (or $\log(t) = 0$) has a significant importance. First, we note that $c(t = 1)$ is already small, $\varepsilon_1(t = 1) \approx 1$ and $\varepsilon_2(t = 1) \approx 0$. Then, the curve of $\varepsilon_1(t)$ suggests that an accurate approximation of $\mu_\infty(t)$ is given by $\mu(t) + 1$. Finally, for $t < 1$ and $\alpha = 1.1$, we know that $\mu_\infty(t)$ is bounded by $\lambda \times t$ the average total number of requests where $\lambda = \zeta(1.1) \approx 10.5844$. Therefore if the cache capacity B is greater than 10, it will take in average more than $t_{B,\min} = \frac{B}{\lambda} > 1$ second to have B different files requested. Hence, we can expect the average value of T_B i.e. t_B to be greater than 1 second. Indeed, $c_\nu(t_B)$ becomes negligible. Thus the CTA of $T_{B,n}$ by a constant t_B solution of $\mu(t_B) = B - 1$ (instead of $\mu_\infty(t_B) = B$) is still accurate.

Geometric popularity law Fricker et *al.* [41] proved that the CTA is accurate even for any content popularity distribution if the condition "*the variance* $\sigma^2_N(t)$ *diverges as* $N, t \uparrow \infty$" holds. Moreover, they considered a Geometric popularity law (i.e. $\lambda_n = \rho^n$) for which they found that the latter condition is not satisfied and the CTA still works. They found that $\mu_\infty(t)$ grows unboundedly and $\sigma^2_\infty(t)$ is asymptotically constant when $t \uparrow \infty$ in order to explain why the CTA works since $c^2_N(t) \longrightarrow_{t,N} 0$. We shall prove this result using the convergence of series $\mu_N(t)$ and $\sigma^2_N(t)$ for finite $t > 0$ not necessarily when $t \uparrow \infty$.

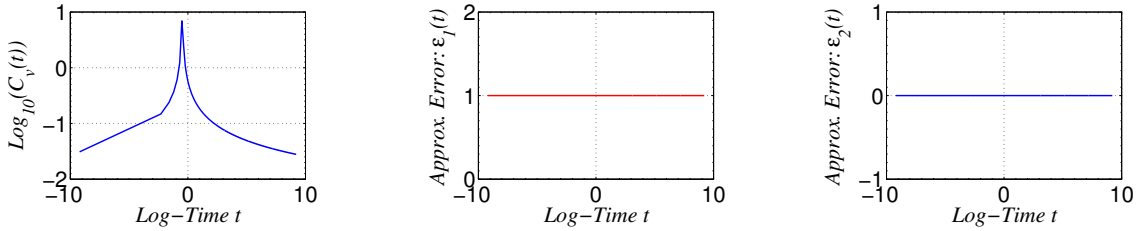


Figure 2.6: Geometric popularity, $\rho = 0.9$

- $\rho \geq 1$. In this case $p_n(t) = 1 - e^{-\rho^n t} \to_n 1$ and the mean $\mu_N(t)$ diverges. By Proposition 2.5, the TTL T_B may be approximated by a constant t_B. Let us see how the variance $\sigma^2_N(t)$ behaves. By the equivalence principle the variance $\sigma^2_N(t)$ and the series of functions $\sum_n (1 - p_n(t))$ have the same nature and both converge by applying the D'Alembert criteria to $\sum_n (1 - p_n(t))$ as follows.

$$\frac{1 - p_{n+1}(t)}{1 - p_n(t)} = (e^{-t})^{\rho^n(\rho-1)} \to_n 0 < 1 \ .$$

  So, the mean $\mu_N(t)$ diverges and the variance $\sigma^2_N(t)$ converges for finite $t > 0$ as $N \uparrow \infty$. In practice with large (but finite) value of N, approximating T_B by a deterministic value t_B solution of $\mu_N(t_B) = B$ is indeed accurate.

- $\rho < 1$. The total rate $\lambda = \sum_n \rho^n$, the mean $\mu_N(t)$ and the variance $\sigma^2_N(t)$ both converge by applying Proposition 2.14 or D'Alembert criteria to the series $\sum_n p_n(t)$.

$$\frac{p_{n+1}(t)}{p_n(t)} = \frac{1 - e^{-\rho^{n+1} t}}{1 - e^{-\rho^n t}} = \frac{1 - e^{-\rho^{n+1} t}}{-\rho^{n+1} t} \times \left(\frac{1 - e^{-\rho^n t}}{-\rho^n t} \right)^{-1} \times \rho \to_n \rho < 1.$$

  We have $\mu(t) \leq \mu_\infty(t) \leq \mu(t) + \varepsilon_1(t)$ and $\sigma^2(t) \leq \sigma^2_\infty(t) \leq \sigma^2(t) + \varepsilon_2(t)$ where

$$\begin{aligned} \lambda &= \rho(1-\rho)^{-1} \\ \mu(t) &= (\log \rho^{-1})^{-1} \left(\gamma + \log(\rho t) + \Gamma(0, \rho t) \right) \\ \sigma^2(t) &= (\log \rho^{-1})^{-1} \left(\log(2) - (\Gamma(0, \rho t) - \Gamma(0, 2\rho t)) \right), \end{aligned}$$

  respectively with error bounds $\varepsilon_0 = 0$, $\varepsilon_1(t) = 1 - (\log \rho^{-1})^{-1} (\gamma(0, t) - \gamma(0, \rho t))$ and $\varepsilon_2(t) = \varepsilon_1(2t) - \varepsilon_1(t)$ where $\gamma = 0.577$ is the Euler's constant and $\log(.)$ is the natural logarithm.

Figure 2.6 shows $c(t) = \sigma(t)/m(t)$ and the approximation errors for $\rho = 0.9$. Since the average number of different requested files is not greater than $\lambda \times t$ where $\lambda = 9$, no more than

10 different files are requested within $[0, t)$ given that $t \leq 1$. Hence, if B is greater than 10 which is usually the case, it will take in average more than $t_{B,\min} \geq 1$ seconds to request B different files. This explains why [41] noticed that the CTA works in the case of Geometric popularity with $\rho < 1$ (in other words, when the mean and the variance converge) even for small values of time t. The curve of $\varepsilon_1(t)$ suggests that $\mu_\infty(t) \approx \mu(t) + 1$ and T_B may be approximated by the solution of $\mu(t_B) = B - 1$.

Light-tailed popularity law Jelenkoviç [54] considered a generic Light-tailed popularity law (i.e. $\lambda_n = ce^{-\delta n^\beta}$ where $c, \delta, \beta > 0$) for which he derived the asymptotic miss probability [54] using fluid approximation when B and N are infinite. The same author experimentally verifies that his fluid model is still accurate when B is finite. For this latter use case, we shall provide a very simple characterization of the TTL T_B which can be use to easily compute the hit probability of the cache under this light-tailed modulated request traffic. Since $\lambda_n = \phi(n)$ is a positive and decreasing rate function, it follows that the aggregated rate $\lambda = \sum_n \lambda_n$ converges and so the mean $\mu_N(t)$ by Proposition 2.14. A straightforward calculation shows that

$$\lambda \approx \beta^{-1}\delta^{-\beta^{-1}} \int_0^{ce^{-\delta}} \left(\log(cx^{-1})\right)^{\beta^{-1}-1} dx \tag{2.44}$$

$$\mu_N(t) \approx \mu(t) = \beta^{-1}\delta^{-\beta^{-1}} \int_0^{tce^{-\delta}} \frac{1-e^{-x}}{x} \left(\log(tcx^{-1})\right)^{\beta^{-1}-1} dx, \tag{2.45}$$

and $\sigma_N^2(t) \approx \sigma^2(t) = \mu(2t) - \mu(t)$ with error bounds ε_0, $\varepsilon_1(t)$ and $\varepsilon_2(t)$ respectively given in Proposition 2.14 and Corollary 2.1. The constant t_B that approximates T_B can be numerically found solving $\mu(t_B) = B$.

2.7.3 LRU caches under renewal request processes

Hyper-exponential renewal processes Martina et *al.* [72] consider caches fed by renewal request processes with inter-request times drawn from a 2-stage hyper-exponential distribution. These 2-stage hyper-exponential renewal processes are equivalent to Interrupted Poisson Processes [39]. They assumed the intensities of renewal processes $\{\mathcal{R}_n\}_n$ being modulated by a Zipf popularity of parameter $\alpha = \{0.7, 1\}$ i.e. $\lambda_n = n^{-\alpha}$ (we recall that typical values for α found with real traces are within $[0.65, 1]$). Finally, they defined the rates of exponential stages as $\lambda_{n,1} = \lambda_n z$ and $\lambda_{n,2} = \lambda_n z^{-1}$ that incorporate a temporal locality into $\mathcal{R}_n$ through the parameter z.

Using our convergence results, we will prove that the CTA they assumed and experimentally verified is indeed true under their settings. The CDF of inter-request times of file n is $F_n(t) = 1 - pe^{-\lambda_{n,1}t} - (1-p)e^{-\lambda_{n,2}t}$ where $p = 1 - (1+z)^{-1}$. A simple calculation of $p_n(t)$ the CDF of

the forward recurrence time (2.1) leads to the following inequality and equivalence.

$$p_n(t) \geq \lambda_n \int_0^t p e^{-\lambda_n z x} d(x) = p z^{-1}(1 - e^{-\lambda_n z t}) \sim p\lambda_n t.$$

Since we know that the Riemann serie $\sum_n n^{-\alpha}$ diverges for $\alpha < 1$, it follows from the equivalence principle and the previous inequalities that $\sum_n p z^{-1}(1 - e^{-\lambda_n z t})$ and $\mu_N(t) = \sum_n p_n(t)$ diverge as well. By Proposition 2.5, we can approximate the characteristic time T_B by a constant t_B as it was assumed in [72] without proof.

Hypo-exponential and shifted-exponential renewal processes We investigate other classes of renewal request processes whose inter-request times have a coefficient of variation $c < 1$.

First, we consider that $\mathcal{R}_n$ is a hypo-exponential renewal process with rate λ_n. In this case, $c^2 \in [1/2, 1[$ and the CDF of inter-requests times is $F_n(t) = 1 - p e^{-\lambda_{n,1} t} - (1-p) e^{-\lambda_{n,2} t}$ where $\lambda_{n,i} = 2\lambda_n \left(1 + (-1)^{(i-1)} \sqrt{2c^2 - 1}\right)^{-1}$, $i = 1, 2$ and $p = \frac{\lambda_{n,2}}{\lambda_{n,2} - \lambda_{n,1}}$. We assume wlog that $\lambda_{n,2} > \lambda_{n,1}$ i.e. $p > 0$ and $1 - p < 0$. The CDF of the forward recurrence times $p_n(t)$ is given by:

$$p_n(t) = 1 - \frac{p\lambda_n}{\lambda_{n,1}} e^{-\lambda_{n,1} t} + \frac{(p-1)\lambda_n}{\lambda_{n,2}} e^{-\lambda_{n,2} t} \geq 1 - \frac{p\lambda_n}{\lambda_{n,1}} e^{-\lambda_{n,1} t} = 1 - c_0 e^{-\lambda_n c_1 t}$$

where $c_1 = (1 + \sqrt{2c^2 - 1})^{-1}$ and $c_0 = \frac{(1+\sqrt{2c^2-1})^2}{2\sqrt{2c^2-1}}$.

It is obvious that if $\lambda_n = \phi(n)$ where $\phi(.)$ is a non-increasing function (the Zipf-like distribution is a special case $\lambda_n = n^{-\alpha}$, $\forall \alpha > 0$), $p_n(t)$ does not converge to zero as $n \to \infty$. Hence, the mean $\mu_N(t) = \sum_n p_n(t)$ diverges. By Proposition 2.5, we can safely approximate the characteristic time T_B by a constant t_B solution of the equation $\mu_N(t_B) = B$ since N is large but finite in practice.

We now consider that $\mathcal{R}_n$ is a renewal process with rate λ_n and inter-request times drawn from a shifted-exponential distribution with rate parameter $\lambda_{n,0} = \frac{\lambda_n}{c}$ and shift-parameter $d = \frac{1-c}{\lambda_n}$. Here c^2 may take any value less than 1 and the CDF of inter-request times is $F_n(t) = 1 - e^{-\lambda_{n,0}(t-d)}, t \geq d$.

- Consider that $t \geq d$, $p_n(t) = 1 - c e^{(1-c)c^{-1}} \times e^{-c^{-1}\lambda_n t}$. If $\lambda_n = \phi(n)$ where $\phi(.)$ is a non-increasing function (the Zipf-like distribution is a special case $\lambda_n = n^{-\alpha}$, $\forall \alpha > 0$), $p_n(t)$ does not converge to zero unless $c = 1$ which is not possible since we assumed that $c < 1$ in this paragraph. Therefore $\mu_N(t)$ diverges, Propositions 2.5 and 2.13 ensure that we can safely approximate the characteristic time T_B by a constant t_B solution of the equation $\mu_N(t_B) = B$ where N is large but finite in practice. The solution t_B of the latter equation always exists within $[d, \infty[$ if $t_{B,\min} = \frac{B}{\lambda} > d$ otherwise $\mu_N(t)$ should be calculated for $t < d$.

- Consider that $t < d$, $p_n(t) = \lambda_n t$, $\mu_N(t)$ and $\lambda = \sum_n \lambda_n$ have the same nature. This case is interesting only if $t_{B,\min} < d$. Suppose λ_n follows a Zipf-like distribution, then the mean $\mu_N(t)$ diverges if $\alpha \leq 1$ and converges otherwise. When $\mu_N(t)$ diverges we rely on Proposition 2.5 to calculate the approximate value of T_B as the solution of $\mu_N(t_B) = B$. However, when $\alpha > 1$, we find that $\mu_\infty(t) = \zeta(\alpha)t$ and $\sigma^2_\infty(t) = \zeta(\alpha)t - \zeta(2\alpha)t^2$ for $t < d$ and $t_{B,\min} < d$. By Corollary 2.2, we can approximate T_B by a constant t_B solution of $\mu_\infty(t) = B$ within the interval $[0, d[$. Note that the solution of $\mu_\infty(t) = B$ may not exist in $[0, d[$; in this case, we can find t_B using $\mu_N(t) = B$ defined for $t \geq d$.

2.8 Conclusions

In this chapter, we derive asymptotic models of classical replacement policies such LRU, FIFO and RND when requests are generated by stationary and ergodic request processes from a large catalog of files. We showed that caches running these policies can be studied as instances of two general classes of TTL-based caches, namely *TTL-renewing* and *TTL-non-renewing* caches. We noted that our classes of TTL-based caches coincide with the *Geiger Counters of Type I* and *Type II*, thus making the "*Theory of Counters*" a basic block of a unifying framework to the performance analysis of caching systems.

We revisit the Characteristic Time Approximation (CTA) which consists in approximating the expected TTL values of files by an identical and constant value. The current definition of the CTA can be seen as an assumption which states that the TTL of all files are equal in expectation. Our result suggests that the CTA might be reformulated as follows: *the TTL of all files are equal in distribution*. We also investigate why the CTA works well on several existing experiments. We found that in all these studied cases their basic assumptions are enough to theoretically justify the reliance on the CTA instead of having the CTA as an additional assumption for their models. Our theoretic results clarify the conditions under which the CTA is expected to work well, especially when the Poisson assumption does not hold and more generally with stationary request traffic. We provide new results regarding the lower bound of the characteristic time and a general procedure to calculate the TTL of a cache replacement policy under the CTA.

We showed that TTL-based models and TTL-based caches are of particular interest due to certain properties they exhibit: infinite cache capacity and TTL decoupling effect. These properties greatly simplify the calculation of performance metrics, the description of the states of each file in the memory, the characterization of miss processes, and as we shall see in next chapter, the performance analysis of heterogeneous cache networks.

3

A UNIFIED FRAMEWORK FOR PERFORMANCE ANALYSIS OF TTL-BASED CACHE NETWORKS

3.1 Summary

In this chapter we present a theoretic framework for performance analysis of general and heterogeneous TTL-based caches networks. We assume that caches may be either *TTL-renewing* or *TTL-non-renewing*. Moreover, we consider that requests are correlated and described by stationary *Markov-renewal* processes. On a single cache, miss streams of requests are characterized and closed-form formulas are derived for the calculation of the metrics of interest. For cache networks with arbitrary topologies, we first introduce the notion of *routing on polytrees* consisting of: (i) allowing requests of each file to flow towards one or many servers independently of other files and (ii) avoiding request forwarding loops by maintaining independence among several streams of requests (of a given file) that may feed a cache. Then, we address two network operations related to the *merging* and *splitting* of request streams that feed and leave a cache respectively. Finally, exact and iterative procedures are presented to calculate the performance metrics at each cache of our general network.

Keyword 3.1 *Exact analysis, TTL-based cache, general cache networks, routing on polytrees, Markov-renewal theory, theory of counters, superposition and thinning of stationary processes.*

3.2 Introduction

The exponential growth of Internet and the tremendous variety of new technologies—such as online media streaming, social networks, information-centric networks—dedicated to content distribution shed light on very challenging issues. How can we successfully achieve and maintain high-speed access to contents while reducing congestion and adapting to the temporal or geographical locality of the demands on such large scale distributed systems?

While the earlier solution on Internet was to deploy many regional repositories or mirrors of parts of available contents (a.k.a. cache and content placement problem), other design choices rely on caches which can communicate by sending queries among them in order to locate contents in a distributed manner. History shows that the latter choice is very efficient thanks to the success stories of the Domain Name Service [50, 59], Content Distribution Networks [86], and Peer-to-Peer networks [90]. Recent proposals for the future Internet architecture such as Content-Centric Network (CCN) [53] explicitly or implicitly assume that the routers have storage facilities in order to cache part of the traffic according to the mobility of users.

There is a general agreement that caching is desirable, however network designers lack tools to model and evaluate the performance of inter-connected caches. The analysis of such cache networks pose significant challenges due to the various correlations they exhibit on orthogonal components: (i) requests generated by users are naturally dependent (*statistical correlation*), (ii) network topology (*structural correlation*), and (iii) cache management policies (*space correlation*). As shown by Rosensweig et *al.* [83], these three types of correlations can severely impact the performance of cache networks at equilibrium.

Considerable effort has been recently devoted to the analysis of cache networks [20, 23, 42, 72] (for tree cache networks) and [82] (for general cache networks). However, these models are restricted to homogeneous network where caches are running the same replacement policy, namely, Least Recently Used (LRU), Random Replacement (RND), or First-In First-Out (FIFO), etc.). Moreover, the proposed models assume that request streams at any point of the network obey the Independence Reference Model (IRM) i.e. they are generated by Poisson processes [38]. The IRM assumption has been shown to not hold on the miss streams of LRU caches in isolation [57].

Clearly, there is a lack of analytic tools to study general and heterogeneous network of caches (made of LRU, FIFO and RND caches for example) under correlated requests. In this chapter, we develop a unified framework to tackle this problem. We present a versatile cache modeling approach called *TTL-based model* and we study general TTL-based cache networks where requests are correlated. TTL-based caches were shown to be more general than LRU, RND or FIFO caches in *Chapter 2*. However, Chapter 2 is not a prerequisite to understand the current chapter.

Our framework is based on the following building blocks:

- *Markov-renewal theory*: to describe exogenous and correlated requests;
- *Theory of counters*: to characterize the states and miss processes of TTL-based caches;
- *Thinning of processes*: to describe the splitting of request streams;
- *Superposition of processes*: to characterize a process resulting from the merging of two or more independent request streams.

This chapter is organized as follows. Section 3.3 introduces the notation, our main assumptions, and preliminary results regarding the calculation of performance metrics and the characterization of the miss process of a single TTL-based cache. In Section 3.4 we present our model of a general cache network and our concept of "routing as polytrees". Then we recall several existing results from the theory of processes and we apply them to derive an exact and recursive procedure for the performance analysis of cache networks. Section 3.5 concludes this chapter.

3.3 Single TTL-based cache under Markov-correlated requests

Throughout this chapter we mainly focus on particular instances of TTL cache tree networks with infinite buffer sizes. This key hypothesis allows us to decouple the management of the different contents and study each of them separately. For this reason, in what follows we will simply refer to a single file, content, or data chunk (simply called the file). From now on the words "node" and "cache" will be used interchangeably. Also, a cache will always be a TTL-based cache unless otherwise specified

New requests for a file can be generated at any node of the network according to a point process $\mathcal{R}$; these requests are referred to as exogenous requests or arrivals and the sequence of these exogenous request instants $\{t_i, i \geq 1\}$ is called the exogenous request process. If upon the arrival of a new request the file is not present in the cache, the request is *instantaneously forwarded* (i.e. zero processing time) to the server. Once the data is found at a server, a copy of it is *instantaneously transmitted* (i.e. zero delay) to the cache. A new TTL is *independently* set for the new copy of the data. If the cache parameter r is set to one (case: $r = 1$), requests occurring before the TTL expires will reset the timer of the file; otherwise (case: $r = 0$) the TTL is not renewed when a cache hit occurs as shown in Figure 3.1. By convention, the file is permanently stored at the server.

Without loss of generality, we assume that the file is requested and enters in the cache at time $t_0 = 0$. We denote by $\tau_i = t_i - t_{i-1}$ the inter-arrival time between the $(i-1)$-th and i-th request ($i \geq 1$) of the file. We also define the *miss process* at a cache as the successive instants

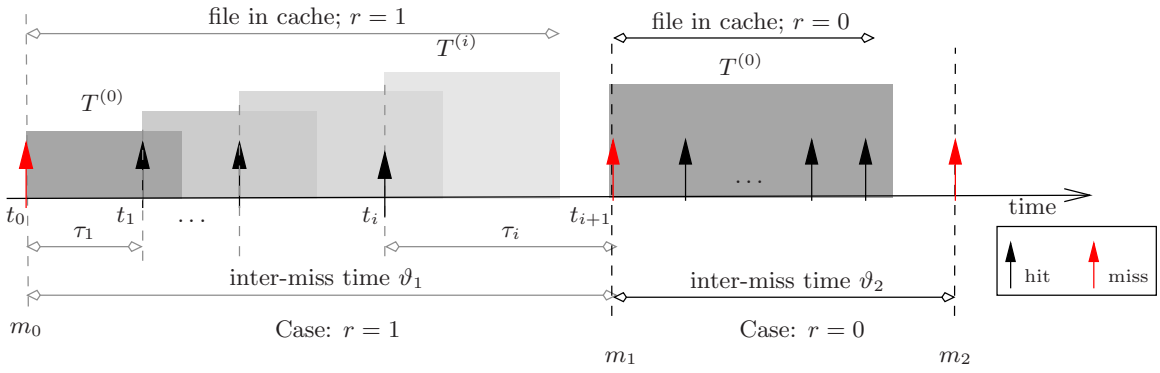


Figure 3.1: Behaviors of a single file in a TTL-based cache (if $r = 1$ TTL is renewed otherwise $r = 0$ TTL is not renewed), request instants $\{t_i\}$, timers $\{T^{(i)}\}$ and miss instants $\{m_i\}$.

at which misses occur at this cache, namely, the times $\{m_i, i \geq 0\}$ at which the file is requested and not found in the cache.

Let us denote by $\{\xi_i, i \geq 0\}$ a homogeneous, irreducible, discrete-time Markov chain on a finite state-space $\mathcal{S} = \{1, 2, \dots, J\}$. The Markov chain $\{\xi_i, i \geq 0\}$ will be referred to as the embedded Markov chain (EMC) and the subscript i refers to the discrete time counter. We denote by $\pi(j)$ the stationary probability that the chain is in state $j \in \mathcal{S}$ and let $\underline{\pi} := (\pi(1), \dots, \pi(J))$ be the stationary vector. Throughout we assume that the system is in steady-state at time $t = 0$. This implies, in particular, that $\pi_i(j) = \pi(j)$ for all $i \geq 0$, $j \in \mathcal{S}$. Since the TTL values are set independently, we denote by $T(t) = P(T^{(i)} < t)$ their cumulative distribution function (CDF).

Assumption 3.1 *(Markov renewal requests) The request process $\mathcal{R} = \{t_i, i \geq 0\}$ is a stationary Markov renewal process; $\forall i \geq 0$, $t > 0$, and $(j, k) \in \mathcal{S}^2$*

$$P(\tau_{i+1} < t, \xi_{i+1} = j \,|\, t_1, \dots, t_i, \xi_1, \dots, \xi_i = k) = P(\tau_{i+1} < t, \xi_{i+1} = j \,|\, \xi_i = k).$$

In other words, the sequence $(t_i, \xi_i)_{i \geq 0}$ defines a stationary Markov renewal process (MRP). Let $\mathbf{F}(t) := [F_{j,k}(t)]_{j,k}$ denote the kernel of this MRP; $\mathbf{F}(t)$ is the $J \times J$ matrix with (j, k)-entry given by $F_{j,k}(t)$ in (3.1). We also define the following matrices $\mathbf{L}(t) := [L_{j,k}(t)]_{j,k}$ and $\mathbf{R}(t) := [R_{j,k}(t)]_{j,k}$ with coefficients $L_{j,k}(t)$ and $R_{j,k}(t)$ respectively given by (3.2) and (3.3)

$$F_{j,k}(t) = P(\tau_i < t, \xi_{i+1} = j \,|\, \xi_i = k) \tag{3.1}$$

$$L_{j,k}(t) = \int_0^t (1 - T(x)) dF_{j,k}(x) \tag{3.2}$$

$$R_{j,k}(t) = \int_0^t (1 - T(x)) dM_{j,k}(x) \tag{3.3}$$

where $\mathbf{M}(t) = \sum_{l \geq 0} \mathbf{F}_{(l)}(t)$ and $\mathbf{F}_{(l)}(t)$ is the l-fold convolution of $\mathbf{F}(t)$. We also introduce the matrix $\mathbf{T}(t) := T(t) \times \mathbf{I}$ where $\mathbf{I}$ is the $J \times J$ identity matrix. For any non-negative rv X with

cdf $\chi(t) = P(X < t)$ $(t \geq 0)$, $\chi^*(s) = E[e^{-sX}] = \int_0^\infty e^{-st} d\chi(t)$ $(s \geq 0)$ denotes its Laplace-Stieltjes Transform (LST). For any number $a \in [0,1]$, $\bar{a} := 1 - a$. In particular, if $\chi(t)$ is a CDF, $\bar{\chi}(t) = 1 - \chi(t)$ is the corresponding Complementary Cumulative Distribution Function (CCDF). The integral and derivative operators are taken element-wise for matrices.

Now we are ready to analyze our TTL-based cache. The next proposition gives a characterization of the miss process $\mathcal{M}$ of a TTL-based cache fed by the MRP $\mathcal{R}$ described above.

Proposition 3.1 (Miss process of TTL-non-renewing cache under MRP—Case $r = 0$) *Under Assumption 3.1, the miss process $\mathcal{M}$ of a single TTL-non-renewing cache with parameter $r = 0$ is a Markov renewal process. Its kernel is given by*

$$\mathbf{G}^{(0)}(t) = \mathbf{F}(t) - \int_0^t d\mathbf{R}(x)(\mathbf{I} - \mathbf{F}(t-x)) \tag{3.4}$$

and the LST $\mathbf{G}^{(0)}(t)$ is given by

$$\mathbf{G}^{(0)*}(s) = \mathbf{I} - (\mathbf{I} + \mathbf{R}^*(s))\ (\mathbf{I} - \mathbf{F}^*(s)). \tag{3.5}$$

Proposition 3.2 (Miss process of TTL-renewing cache under MRP—Case $r = 0$) *Under Assumption 3.1, the miss process $\mathcal{M}$ of a single TTL-renewing cache with parameter $r = 1$ is a Markov renewal process. Its kernel is given by*

$$\mathbf{G}^{(1)}(t) = \mathbf{F}(t) - \mathbf{L}(t) + \int_0^t d\mathbf{L}(x)\mathbf{G}^{(1)}(t-x), \tag{3.6}$$

and the LST of $\mathbf{G}^{(1)}(t)$ is given by

$$\mathbf{G}^{(1)*}(s) = \mathbf{I} - (\mathbf{I} - \mathbf{L}^*(s))^{-1}\ (\mathbf{I} - \mathbf{F}^*(s)). \tag{3.7}$$

These results follow from the analogy between our TTL-based caches and the *Geiger counters of Type* I and *Type* II (See Remark 2.5 in Chapter 2). Classical results of the Theory of counters [22, Theorem 10.20] and [22, Theorem 10.33] establish that the miss process (analogously, the sequence of registered particles) of a TTL-based (analogously, a counter) is also a Markov renewal process under Assumption 3.1. The kernel $\mathbf{G}^{(0)}(t)$ and $\mathbf{G}^{(1)}(t)$ of the miss process are given in [22, Formula (10.21)] and [22, Formula (10.34)] when $r = 0$ and $r = 1$ respectively. Finally, (3.4) and (3.6) are obtained by simple matrix algebra.

Two cache performance metrics are of particular interest: the hit probability $H_P(j)$ defined as the probability that an arriving request finds the file in the cache, and the occupancy $O_P(j)$ which is the probability that the file is in the cache at any time given that the EMC is in state j.

Proposition 3.3 (Cache performance) *When the environment is in state $j \in \mathcal{S}$, the stationary probability $H_P(j)$ that a request finds the file in the cache is obtained as follows*

$$H_P(j) = \bar{r}\left(1 - (1 + \mathbf{e}_j\mathbf{R}(\infty)\mathbf{1})^{-1}\right) + r\,\mathbf{e}_j\mathbf{L}(\infty)\mathbf{1}. \tag{3.8}$$

Moreover, the stationary probability $O_P(j)$ to find the file in the cache at any time is given by

$$O_P(j) \;=\; \lambda_j\left(\bar{r}\,\mu^{-1}(1 + \mathbf{e}_j\mathbf{R}(\infty)\mathbf{1})^{-1} + r\,\left[\mu^{-1} - \int_0^\infty \mathbf{e}_j(\mathbf{I} - \mathbf{T}(t))\mathbf{F}(t)\mathbf{1}\right]\right) \tag{3.9}$$

where $\lambda_j^{-1} = E[\tau_1|\xi_1 = j] = \int_0^\infty (1 - \sum_k F_{j,k}(t))dt$, $\mu^{-1} = E[T]$, $\mathbf{e}_j$ is a row vector of zeros but 1 at the j-th position, and $\mathbf{1}$ is a column vector of ones.

Proof When the EMC is in state $j \in \mathcal{S}$ at time t_0, the hit probability $H_P(j)$ is the probability that an arriving request finds the data in the cache i.e. the TTL has not expired yet; while the occupancy $O_P(j)$ is the steady-state probability that a data is in the cache at any time.
If $r = 0$ then $H_P(j)$ is the ratio of the number of requests that occur during T by the total number of requests needed to observe a cache miss. If $r = 1$ then $H_P(j) = P(\tau_1 < T^{(1)}|\xi_0 = j)$. Hence,

$$H_P(j) = r\,P(\tau_1 \leq T^{(1)}|\xi_0 = j) + \frac{\bar{r}\,\mathbf{e}_j\,\mathbf{R}(\infty)\,\mathbf{1}}{1 + \mathbf{e}_j\,\mathbf{R}(\infty)\,\mathbf{1}}$$

The miss probability is $M_P(j) = 1 - H_P(j)$. Since λ_j is the arrival rate of requests at cache in state j, the stationary hit and miss rates at cache is $H_R(j) = \lambda_j H_P(j)$ and $M_R(j) = \lambda_j(1 - H_P(j))$ respectively.
Equation (3.9) follows by applying Proposition 2.9 in Chapter 2 that provides the relation between $H_P(j)$ and $O_P(j)$; hence the hit probability and the occupancy are given by

$$H_P = \underline{\pi} \times (H_P(1), \ldots, H_P(J))^t$$

$$O_P = \underline{\pi} \times (O_P(1), \ldots, O_P(J))^t$$

where $(.)^t$ is the vector transpose operator. ∎

The previous results can be easily extended to the case where r is a Bernoulli random variable such that the cache adopts a TTL-renewing behavior with probability $E(r)$ and a TTL-non-renewing behavior with the complementary probability. In this case, one can easily prove the following result:

Corollary 3.1 (Miss process—Case r is a Bernoulli rv) *Under Assumption 3.1, the miss process $\mathcal{M}$ of a single cache with parameter $r \stackrel{d}{=}$ Bernoulli$(E(r))$ is a Markov renewal process. And its kernel is given by*

$$\mathbf{G}(t) = (1 - E(r))\,\mathbf{G}^{(0)}(t) + E(r)\,\mathbf{G}^{(1)}(t), \tag{3.10}$$

where $\mathbf{G}^{(0)}(t)$ *and* $\mathbf{G}^{(1)}(t)$ *are provided by* (3.4) *and* (3.6) *respectively. The LST* $\mathbf{G}^*(s)$ *of the latter kernel is given by*

$$\mathbf{G}^*(s) = \mathbf{I} - \left[E(r)(\mathbf{I} - \mathbf{L}^*(s))^{-1} + (1 - E(r))(\mathbf{I} + \mathbf{R}^*(s))\right](\mathbf{I} - \mathbf{F}^*(s)). \tag{3.11}$$

Thanks to the characterization of TTL-based caches in isolation, we are now ready to study cache networks. Propositions 3.1 and 3.3 will be repeatedly used at each cache.

3.4 Heterogeneous TTL-based cache networks

3.4.1 Model description

Network model Let $\mathcal{G} = (V, E)$ be the graph representing our cache network, $V = \{v_1, \ldots, v_N\}$ the set of caches or nodes, and $E \subset V \times V$ the set of cache inter-connections. Since each TTL-based cache in our network has an infinite capacity (and thus files are decoupled), we can focus on the description of a single file over this network $\mathcal{G}$.

Workload model At some of the nodes $\{v_n \in V\}$ in this system, a stream of file access requests arrives exogenously. We denote by $\mathcal{R}_n = \{t_{n,i}\}_{i \geq 0}$ the exogenous request process of the file at node v_n where $t_{n,i}$ is the arrival instant of the i-th request. Let λ_n denotes the rate of exogenous arrival at cache v_n. $\mathcal{R}_n$ is assumed to be a Markov renewal process (Assumption 3.1).

Routing model We consider that requests of our file are routed on a *polytree* that is a directed graph without any undirected cycles. This polytree, noted $\mathcal{P}$, consists of nodes of the general graph $\mathcal{G}$ and it is built for each file (possibly independently of other files) by a routing protocol or forwarding rule implemented at each node of our network in order to avoid routing loops. We also consider that the file is stored permanently at one or more custodians or public servers that are attached to the network at the root(s) of $\mathcal{P}$. An illustration of this routing model is shown in Figure 3.2. In this chapter, we will consider a *probabilistic routing* meaning that in the case of several paths (e.g. at cache v_2 in Figure 3.2), requests are forwarded among several destinations according to a probability distribution (e.g requests are sent from v_2 to v_3 with probability p and from v_2 to v_5 with probability $1 - p$ in Figure 3.2). This routing model guarantees that multiple streams of requests at each cache of the network are all independent. It is interesting to note that if the network topology of $\mathcal{G}$ is a tree then requests are all routed on this tree; and the independence among request streams becomes more obvious.

File and cache network states When a request for the file arrives at a cache, it generates a *hit* if the file is located at the cache and a *miss* if not. In the event of a miss, the request is forwarded

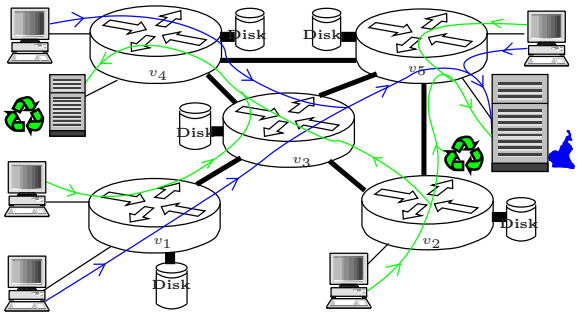


Figure 3.2: Five nodes general cache network with two files: network topology (in black), requests for *blue* file are routed as a tree (in blue) and that of the *green* file are routed as a polytree (in green).

to other nodes in the network based on the routing model (or routing table) at each cache, until the file is located in a cache or at the server storing the file. Then the file is forwarded along the reverse path taken by the request, and stored at each cache along the way. A timer is set for each new copy of the file at each node where the cache miss occurred during the request forwarding process. Meanwhile, the TTL is reset (resp. not reset) at the node where the file was found according to its class of the TTL-based cache i.e. if the cache parameter $r_n = 1$ (resp. $r_n = 0$) as shown in Figure 3.1. We further assume zero delay and processing time i.e. requests and copies of the file are instantaneously propagated in the network.

Now we are ready to analyze our general cache network. First, we will present classical results on theory of processes that enable us to describe network primitives such as *merging* and *splitting* streams of requests. And finally we will describe exact and recursive procedures to calculate the metrics of interest at each cache of the network.

3.4.2 Aggregating request streams

In our network model, request streams that arrive on a cache may be of one the two kinds: *exogenous* (e.g. directly from users) or *endogenous* (namely, miss processes of other caches). We denote by $\mathcal{C}(n)$ the subset of nodes in V that belong to $\mathcal{P}$ and forward their requests to node v_n. Note that if $\mathcal{G}$ has a tree topology or if requests of the file are routed as a tree, the set $\mathcal{C}(n)$ is simply the "children" nodes of node v_n.

We start the description of aggregate processes at the leaves of the routing polytree $\mathcal{P}$. At these nodes, the aggregate request process is exactly the exogenous request process. Under Assumption 3.1 the miss process at each leaf is also a Markov renewal process by Proposition 3.1.

Then we move to higher level caches of $\mathcal{P}$, say node v_n. This cache is fed by the request process $\mathcal{A}_n$ resulting from the aggregation of the exogenous request process $\mathcal{R}_n$ and the miss

request processes $\{\mathcal{M}_j,\ j \in \mathcal{C}(n)\}$. By Assumption 3.1 and Proposition 3.1, $\mathcal{R}_n$ and $\{\mathcal{M}_j,\ j \in \mathcal{C}(n)\}$ are all Markov renewal processes; furthermore, they are all independent due to our Routing model based on the polytree $\mathcal{P}$. Therefore, the aggregated request process $\mathcal{A}_n$ is a stationary point process [10, Sect. 1.4.2] and the CDF of its first inter-request time under the Palm probability of the process $\mathcal{A}_n$ is given by [10, Formula (1.4.6)]. Moreover $\mathcal{A}_n$ is also a stationary Markov renewal process; this conclusion follows from a classical result by Korolyuk [64].

3.4.3 Splitting a request stream

In our routing model, missed requests of a cache are probabilistically forwarded in the case of two or more destinations. Hence, we shall describe each component process resulting from the splitting of the original process. This operation is also known as *thinning a process* from the theory of point processes [52]. We denote by $\mathcal{F}(n)$ is the set of cache in the forwarding table of node v_n and we consider that a request is sent from v_n to $i \in \mathcal{F}(n)$ with probability $p_{n,i}$ such that $\sum_{i\in\mathcal{F}(n)} p_{n,i} = 1$. This probabilistic routing is also called a *Bernoulli routing*. Since all the processes of our network are Markov renewal processes, another classical result by Manor [71] establishes that a Bernoulli thinned process denoted $\mathcal{T}_{n,i}$ of a Markov renewal process $\mathcal{M}_n$ (or $\mathcal{A}_n$) with probability $p_{n,i}$ at a node v_n in the network is also a stationary Markov renewal process. More complex routing schemes such as *Markov Chain-based routing* can be implemented using the more general results on Markov chain thinning of a stationary point process by Isham [52].

In light of all these classical results, we note that the main network primitives, namely *(cache) filtering, aggregating*, and *splitting*, transform a Markov renewal process into another (simple or complex) one. Hence, all the processes involved in our network are exactly described and all belong to the same class of Markov renewal processes under Assumption 3.1.

3.4.4 Exact procedures for cache networks

We present exact, unified, and iterative procedures to study general and heterogeneous TTL-based cache networks introduced in Section 3.4.1. We consider that each file is routed as a polytree $\mathcal{P}$ and caches set the timer of a file independently of other files.

Case of infinite cache capacity Our algorithm consists in selecting a file and its routing polytree $\mathcal{P}$, then calculating the aggregated process, the miss process and the thinned processes, if any, recursively at each cache of the routing polytree $\mathcal{P}$. We summarize the procedure in the following algorithm.

Algorithm 1: Exact and Iterative Procedure for Performance Analysis of General and Heterogeneous TTL-based Cache Networks – Case of infinite cache capacity.

input : Graph $\mathcal{G} = (V, E)$, Files $F = \{f,\ f = 1, \dots, |F|\}$, Routing tables $\{\mathcal{P}(f),\ f = 1, \dots, |F|\}$, Exo. procs & TTLs $\{(\mathcal{R}_{n,f}, T_{n,f}, r_{n,f}),\ \forall (v_n, f) \in V \times \{1, \dots, |F|\}\}$

output: Performance Metrics $\{(H_{P,n,f}, O_{P,n,f}),\ \forall\, v_n \in V,\ f = 1, \dots, |F|\}$

for $f \leftarrow 1, \dots, |F|$
  $\mathcal{P} \leftarrow \mathcal{P}(f)$
  do
    $v_n \leftarrow$ select a node in $\mathcal{P}$ ▷ Starting from leaves
    $\mathcal{A}_n \leftarrow$ aggregate request streams [64] $\{\mathcal{M}_j, \mathcal{T}_{i,n}, \mathcal{R}_n,\ j \in \mathcal{C}(n),\ n \in \mathcal{F}(i)\}$
    $\mathcal{M}_n \leftarrow$ characterize miss stream, Proposition 3.1 $\mathcal{A}_n$
    $(H_{P,n,f}, O_{P,n,f}) \leftarrow$ calculate hit & occ. probs, Proposition 3.3 $\{\mathcal{A}_n, \mathcal{M}_n, T_{n,f}, r_{n,f}\}$
    $\{\mathcal{T}_{n,i}, i \in \mathcal{F}(n)\} \leftarrow$ split request streams [71] $\{\mathcal{M}_n, p_{n,i},\ i \in \mathcal{F}(n)\}$
  while v_n is not a server
end

Case of finite cache capacity: Polytree and Tree based network topology In this case, $\mathcal{G}$ is a polytree (or simply a tree) graph as shown in Figure 3.3. The cache capacity constraints B_n are given at each node. These capacity constraints are translated into the following terms: the cache occupancy of v_n should not exceed B_n in average.

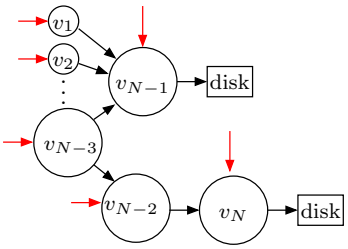


Figure 3.3: Polytree cache network.

Hence, the expected TTL values $E[T_{n,f}]$ must satisfy the following equation:

$$\sum_{f=1}^{|F|} O_{P,n,f} \leq B_n,\ \forall v_n \in V \tag{3.12}$$

where $O_{P,n,f}$ is given in (3.9) as a function of the expected TTL $\mu_{n,f}^{-1} = E[T_{n,f}]$.
Therefore, Algorithm 1 is modified as follows.

Note that requests of all files are flowing in the "*same direction*" i.e. towards the root(s) of

Algorithm 2: Exact and Iterative Procedure for Performance Analysis of Heterogeneous TTL-based Polytree Cache Networks—Case of finite capacity.

input : Graph $\mathcal{G} = (V, E)$, Files $F = \{f,\ f = 1, \dots, |F|\}$, Routing tables $\{\mathcal{P}(f),\ f = 1, \dots, |F|\}$, Exo. procs & TTLs $\{(\mathcal{R}_{n,f}, r_{n,f}),\ \forall (v_n, f) \in V \times \{1, \dots, |F|\}\}$

output: Performance Metrics $\{(H_{P,n,f}, O_{P,n,f}, T_{n,f}),\ \forall\, v_n \in V,\ f = 1, \dots, |F|\}$

do
  $v_n \leftarrow$ select a node in $\mathcal{G}$ ▷ Starting from leaves
  for $f \leftarrow 1, \dots,\ |F|$
    $\mathcal{A}_{n,j} \leftarrow$ aggregate request streams [64] $\{\mathcal{M}_{j,f}, \mathcal{T}_{i,n,f}, \mathcal{R}_{n,f},\ j \in \mathcal{C}(n),\ n \in \mathcal{F}(i)\}$
    $(O_{P,n,f}(E[T_{n,f}])) \leftarrow$ calculate functs, Prop. 3.3 $\{\mathcal{A}_{n,f}, r_{n,f}\}$
  end
  $\{E[T_{n,f}], f = 1, \dots, |F|\} \leftarrow$ calculate expected TTLs, Eq. (3.12) $\{B_n, O_{P,n,f}(E[T_{n,f}])\}$
  for $f \leftarrow 1, \dots,\ |F|$
    $(H_{P,n,f}, O_{P,n,f}) \leftarrow$ calculate hit & occ. probs, Proposition 3.3 $\{E[T_{n,f}]\}$
    $\mathcal{M}_{n,f} \leftarrow$ characterize miss stream, Proposition 3.1 $\mathcal{A}_{n,f}$
    $\{\mathcal{T}_{n,i,f}, i \in \mathcal{F}(n)\} \leftarrow$ split request streams [71] $\{\mathcal{M}_{n,f}, p_{n,i,f},\ i \in \mathcal{F}(n)\}$
  end
while v_n is not a server

the graph $\mathcal{G}$ and Algorithm 2 takes advantage of this polytree structure to solve Eq. (3.12) step by step at each node of the network before characterizing the miss process on each file.

Case of finite cache capacity and arbitrary network topology The main difference with the previous case is that requests may flow in "*opposite direction*". This case has been shown to possibly lead to situations with very poor network performance since states of caches are dependent [83]. The dependence of the caches states on each other prevents us from calculating the expected TTL values $\mathrm{E}[\mathrm{T}_{n,f}]$ in one step as previously done. In fact all processes arriving on a cache might have not been characterized. In order to clarify this claim, we illustrate the situation on a basic network: a tandem of two caches (see Figure 3.4).

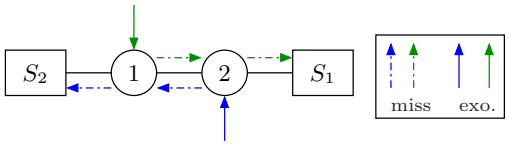


Figure 3.4: Tandem of two caches and two files: one is green and the other is blue.

We express the capacity constraints of this simple network as follows

$$\mathrm{O}_{\mathrm{P},1,g}\left(\mathrm{E}[\mathrm{T}_{1,g}]\right) + \mathrm{O}_{\mathrm{P},1,b}\left(\mathrm{E}[\mathrm{T}_{2,b}], \mathrm{E}[\mathrm{T}_{1,b}]\right) \leq \mathrm{B}_1; \tag{3.13}$$

$$\mathrm{O}_{\mathrm{P},2,g}\left(\mathrm{E}[\mathrm{T}_{1,g}], \mathrm{E}[\mathrm{T}_{2,g}]\right) + \mathrm{O}_{\mathrm{P},2,b}\left(\mathrm{E}[\mathrm{T}_{2,b}]\right) \leq \mathrm{B}_2. \tag{3.14}$$

where the occupancy $\mathrm{O}_{\mathrm{P},1,b}$ (resp. $\mathrm{O}_{\mathrm{P},2,g}$) of the blue file b (resp. green file g) at cache 1 (resp. cache 2) depends on both expected TTL values $\mathrm{E}[\mathrm{T}_{1,b}]$ and $\mathrm{E}[\mathrm{T}_{2,b}]$ (resp. $\mathrm{E}[\mathrm{T}_{1,g}]$ and $\mathrm{E}[\mathrm{T}_{2,g}]$). Therefore the capacity constraints (3.13) and (3.14) are coupled by the expected TTL values $\{\mathrm{E}[\mathrm{T}_{1,f}], \mathrm{E}[\mathrm{T}_{2,f}], f = g, b\}$. Our general network algorithm will initialized the expected TTL values and increases their values until the equality is met in the capacity constraints (3.13) and (3.14).

The initialization of $\mathrm{E}[\mathrm{T}_{n,f}] = \frac{\mathrm{B}_n}{\sum_f \Lambda_{n,f}}$ in step 3 of Algorithm 3 is obtained as follows. We first note that $\mathrm{O}_{\mathrm{P},n,f}$ is the expected number of file f in cache n. This implies that $\mathrm{O}_{\mathrm{P},n,f} \in [0, 1]$. Meanwhile, $\mathrm{O}_{\mathrm{P},n,f}$ is upper bounded by $\Lambda_{n,f}\mathrm{E}[\mathrm{T}_{n,f}]$ the expected number of requests for file f that a Poisson process with same rate as the aggregated request process $\mathcal{A}_{n,f}$ would have generated during the period $\mathrm{E}[\mathrm{T}_{n,f}]$. By initializing the expected TTL value to same value $\frac{\mathrm{B}_n}{\sum_f \Lambda_{n,f}}$, the network capacity constraints are not violated. Then, Algorithm 3 will adjust these expected values at each iteration by actualizing the miss processes of each cache.

Algorithm 3: Exact and Iterative Procedure for Performance Analysis of Heterogeneous TTL-based Polytree Cache Networks

input : Graph $\mathcal{G} = (V, E)$, Files $F = \{f,\ f = 1, \dots, |F|\}$, Routing tables $\{\mathcal{P}(f),\ f = 1, \dots, |F|\}$, Exo. procs & TTLs $\{(\mathcal{R}_{n,f}, r_{n,f})\ ,\ \forall (v_n, f) \in V \times \{1, \dots, |F|\}\}$

output: Performance Metrics $\{(H_{P,n,f}, O_{P,n,f}, T_{n,f})\ ,\ \forall\ v_n \in V,\ f = 1, \dots, |F|\}$

$\mathcal{M}_{n,f} \leftarrow \mathcal{R}_{n,f}$ ▷ initialize miss stream, as if $M_{P,n,f} = 1$

$\mathcal{A}_{n,j} \leftarrow$ initialize request streams [64] $\{\mathcal{M}_{j,f}, \mathcal{T}_{i,n,f}, \mathcal{R}_{n,f},\ j \in \mathcal{C}(n),\ n \in \mathcal{F}(i)\}$

$\{E[T_{n,f}], f\} \leftarrow$ initialize expected TTLs $\left\{\frac{B_n}{\sum_f \Lambda}_{n,f}, O_{P,n,f}(E[T_{n,f}])\right\}$ ▷ $\Lambda_{n,f}$ is the rate of $\mathcal{A}_{n,f}$

do

  $v_n \leftarrow$ select a node in $\mathcal{G}$ ▷ Starting from edges

  for $f \leftarrow 1, \dots,\ |F|$

    $\mathcal{M}_{n,f} \leftarrow$ characterize miss stream, Proposition 3.1 $\mathcal{A}_{n,f}$

    $\mathcal{A}_{n,j} \leftarrow$ aggregate request streams [64] $\{\mathcal{M}_{j,f}, \mathcal{T}_{i,n,f}, \mathcal{R}_{n,f},\ j \in \mathcal{C}(n),\ n \in \mathcal{F}(i)\}$

    $(O_{P,n,f}(E[T_{n,f}])) \leftarrow$ calculate functs, Prop. 3.3 $\{\mathcal{A}_{n,f}, r_{n,f}\}$

  end

  $\{E[T_{n,f}], f\} \leftarrow$ update TTLs $\{B_n, O_{P,n,f}(E[T_{n,f}])\}$ ▷ Network cache capacity constraints

  for $f \leftarrow 1, \dots,\ |F|$

    $(H_{P,n,f}, O_{P,n,f}) \leftarrow$ calculate hit & occ. probs, Proposition 3.3 $\{E[T_{n,f}]\}$

    $\mathcal{M}_{n,f} \leftarrow$ characterize miss stream, Proposition 3.1 $\mathcal{A}_{n,f}$

    $\{\mathcal{T}_{n,i,f}, i \in \mathcal{F}(n)\} \leftarrow$ split request streams [71] $\{\mathcal{M}_{n,f}, p_{n,i,f},\ i \in \mathcal{F}(n)\}$

  end

while $\sum_f O_{P,n,f} = B_n,\ \forall v_n \in V$

3.5 Conclusions

We summarize the main contributions of this chapter. We presented a unique model for single TTL-based caches. Then we derived the cache performance metrics and we characterize the cache miss processes under the assumption that requests are correlated by Markov renewal processes. We presented a unified framework for the performance analysis of general and heterogeneous TTL-based cache networks. This framework is built on top of existing results of *Theory of counters* and *Theory of stationary point processes* (superposition and thinning of stationary point processes). We proposed three algorithms to analyze heterogeneous cache network having general topology, implementing a probabilistic routing based on per-file polytrees, and receiving requests correlated by Markov renewal processes. Our analytic findings and our algorithms lead in theory to exact results at a price of a huge complexity. We will see in next chapter that our assumption of *Markov-correlated* requests can be strengthened and stronger statements such as *requests described renewal processes or Markov-Arrival processes* can be assumed in several applications without loosing much in the accuracy of our framework.

4

APPLICATION CASES: CONTENT-CENTRIC NETWORKS AND DOMAIN NAME SYSTEM

4.1 Summary

In this chapter, we are interested in TTL-based cache networks where requests may be described by renewal processes i.e. they are assumed to be independent and identically distributed. We study TTL-based caches within two contexts: (i) Content-Centric Networks (CCNs) and (ii) Domain Name System (DNS). In the first case a TTL-based policy is presented as proposal of cache management algorithm of content-routers; while, TTL-based models are used to describe a recent behavior of DNS caches over Internet in the second case. In both application cases, our studies show that performance metrics of these cache networks can be predicted with high accuracy i.e. the relative errors are within 1%–5%.

Keyword 4.1 *Performance analysis, Content-Centric Networks, (modern) Domain Name System, TTL-based cache, renewal theory, optimal TTL distribution.*

4.2 Content-Centric Networks

Many researchers have been working on performance analysis of cache networks where caches may run various replacement policies like Least Recently Used (LRU), First-In First-Out (FIFO) or Random (RND). However, no exact results are provided, and many approximate models do not scale even for the simple tandem of two caches. In this section, we study a

Time-To-Live based policy (TTL), that assigns a timer to each content stored in the cache and redraws the timer each time the content is requested (at each hit/miss). These TTL-based caches are called *TTL-renewing caches* in previous chapters. We derive the analysis of networks of these TTL-based caches under the assumption that requests are independent and identically distributed i.e. generated by *renewal processes*. In particular, we determine exact formulas for the metrics of interest of linear-star networks which generalize tandem and two-level tree of caches. For more general and hierarchical networks, our approximate solution is very accurate with the relative errors smaller than 10^{-2} for both Matrix-Exponentially Distributed and constant TTLs.

4.2.1 Introduction

Caches are widely used in networks and distributed systems for improving performance. They are integral components of the Web [23], the Domain Name Service (DNS) [60], and Content Distribution Networks (CDNs) [86]. More recently there has been a growing emphasis on *content networks* (like Content-Centric Networks) where content or data item is centric and host-to-content interaction is the common case [53]. Many of these systems give rise to hierarchical (tree) cache topologies and to even more general irregular topologies. The design, configuration, and analysis of these cache systems pose significant challenges. A few approximations have been proposed, instead, for a simple two-level LRU cache network [23] and general LRU cache networks [82]. However, their inaccuracies can be significant as reported in [82] where the relative error reaches 16%.

When an uncached data item is brought back into the cache due to a cache miss, a local TTL is set. TTL values can be different for different data, but also for the same data item at different caches[1]. All requests to that data item before the expiration of the TTL are cache hits; the first request for that data item to arrive after the TTL expiration will yield a cache miss. In that case, the cache may forward the request to a higher-level cache, if any, or to the server. When located, the data item is routed on the reverse-path and a copy is placed in each cache along the path.

This proposal makes the case that TTL-based policies are interesting alternatives to replacement algorithms such as LRU or RND for two reasons. First, a TTL policy is more configurable (it can implement service differentiation, quality of service as shown in *Chapter 7*) and in particular it can mimic the behavior of other replacement policies thanks to a proper choice of parameters (i.e. the TTL distributions, see Section 4.2.8). Second, while LRU or RND cache networks have defied accurate analysis, networks of TTL-based caches appear simpler to study as we shall show in Sections 4.2.3 and 4.2.4.

[1]This is then different from the TTLs in DNS system where the TTL for a given data item is in general set to a common value determined by the authoritative name server (see Section 4.3).

We develop a set of building blocks for the performance evaluation of hierarchical TTL cache networks where TTLs are set at each request instant. These blocks allow one to model exogenous requests at different caches as *independent renewal processes* and to describe TTL duration by *arbitrary distributions* as long as requests and timers are independent of each other. The building blocks consist of:

- a renewal model of a *single content* TTL cache when fed by a renewal request stream,
- a renewal process approximation of the superposition of independent renewal processes.

The first block forms the basis to evaluate the performance metrics and to describe the output sequence of requests (the miss stream) of a cache. Meanwhile, the second block is used to characterize the resulting process of the superposition of several independent streams of requests consisting of exogenous requests from users and/or missed requests from other caches if any. These blocks are applied to assess the performance metrics of hierarchical TTL-based cache networks. We then show how the computational cost of our approach simplifies when TTLs and the inter-arrival times of the exogenous request streams at every cache are Matrix-Exponentially Distributed (MED). We refer to this case in short as a *MED cache network*. The class of Matrix-Exponential distributions coincides with the class of distributions having a rational Laplace-Stieltjes Transform that can be used to fit properties of general processes [35, 46, 61, 77]. We derive exact results for some cases but when they are not, the relative errors are extremely small. Precisely, event-driven and Monte-Carlo simulations on instances of MED cache networks reveal that the relative errors between the simulated networks and our model predictions are less than 10^{-2} for all metrics of interest. Thus, we believe our approach is promising and capable of accurately modeling a richer class of network topologies.

The contributions of this section are:

- the proposal of TTL-based replacement policies for content-routers of ICN architectures,
- an analytic tool to assess the performance of hierarchical TTL-based cache networks.

The remainder of this section is organized as follows. In Section 4.2.2, we introduce the notation and the model assumptions. Section 4.2.3 contains our model of a single TTL-based cache and provides the exact characterization of the performance metrics and the miss process. We describe in Section 4.2.4 and Section 4.2.5 general procedures to study any hierarchical TTL-based cache network. The key points in these sections are how we extend the exact analysis of single cache to cache network in an exact manner and how we model the combined exogenous and miss requests streams as a renewal point process thanks to a result from [69, Eq.(4.1)], [10, Eq.(1.4.6)] regarding the computation of the marginal inter-arrival distribution for a superposition of independent renewal processes respectively. Simplified procedures are

also derived for MED cache networks. The accuracy of the general and of the simplified procedures is evaluated in Section 4.2.6 and a discussion of the computational complexity of our analytic approach can be found in Section 4.2.7. Section 4.2.8 discusses how our TTL-based model can be implemented under finite capacity constraints, and how the TTL policy can mimic different policies like LRU or RND. Our proposal is summarized in Section 4.2.9.

4.2.2 Definitions and assumptions

Throughout this section we mainly focus on particular instances of TTL-based cache tree networks with infinite buffer sizes. This key hypothesis allow us to decouple the management of the different contents and study each of them separately. For this reason, in what follows we will simply refer to a *single* content or data chunk (simply called the data). The effect of finite buffer is considered in Section 4.2.8. From now on the words "node" and "cache" will be used interchangeably. Also, a cache will always be a TTL cache unless otherwise specified.

New requests for a data item can be generated at any node of the network according to mutually independent processes; these requests are referred to as *exogenous* requests or arrivals and the sequence of these exogenous request instants is called the *exogenous request process*. If upon the arrival of a new request the data item is not present in the cache, the request is *instantaneously* forwarded to the next level of the tree and the process repeats itself until the data item is found. In case the data item cannot be found along the path toward the root, the root retrieves it from a server. Once the data item is found, either at a cache or at a server, a copy of it is *instantaneously* transmitted to each cache along the path between the cache where the data item was found and the cache that issued the request[2]. A new TTL is set for each new copy of the data item and the TTL is redrawn at the cache, if any, where the data item was found (by convention, the TTL at the server is infinite). This is in contrast with the model in [59] where there is no TTL reset upon a cache hit. Resetting the TTL also at each cache hit increases the occupancy and the hit probability specially for popular contents (high request rate). This choice is motivated by the host-to-content paradigm of moving popular documents as close as possible to the users.

We define the *miss process* at a cache as the successive instants at which misses occur at this cache, namely, the times at which the data item is requested and is not found in the cache. Let us denote by $\mathcal{C}(n)$ the set of children of cache n. The (overall) *request process*, also called the *arrival process*, at cache n is the superposition of the miss processes of caches in $\mathcal{C}(n)$ and of the exogenous request process at cache n, if any.

If Λ_n is the arrival rate of requests at cache n, $H_{P,n}$ (resp. $M_{P,n} = 1 - H_{P,n}$) and $H_{R,n} = \Lambda_n H_{P,n}$ (resp. $M_{R,n} = \Lambda_n(1 - H_{P,n})$) denote the stationary hit (resp. miss) probability and

[2]We observe that our TTL-based policy could be used also in conjunction with other strategies that do not keep a copy at every cache along a path, like those in [67].

Table 4.1: Glossary of main notation for cache n in a Content-Centric Network

λ_n	Exogenous arrival rate at cache n
Λ_n	Total arrival rate at cache n
$1/\mu_n$	Expected TTL at cache n
$F_n(t)$	CDF exogenous arrivals at cache n
$H_n(t)$	CDF overall arrivals at cache n
$G_n(t)$	CDF inter-miss times at cache n
$T_n(t)$	CDF TTL duration at cache n
$H_{P,n}, M_{P,n}$	Hit, miss probability resp. at cache n
$H_{R,n}, M_{R,n}$	Hit, miss rate resp. at cache n
$O_{P,n}$	Occupancy of cache n (stationary probability content is in cache n
$\mathcal{C}(n)$	Set of children of cache n
$\chi^*(s)$	LST of CDF $\chi(t)$

the stationary hit (resp. miss) rate at cache n, respectively. We denote by $O_{P,n}$ the steady-state probability that the data item is in cache n and we call it the *occupancy* of cache n. Hence, we just have to calculate $H_{P,n}$ and Λ_n.

For $t \geq 0$, $s \geq 0$, and a non-negative random variable X with Cumulative Distribution Function (CDF) $\chi(t) = P(X < t)$, $\bar{\chi}(t) = 1 - \chi(t)$ is the Complementary Cumulative Distribution Function (CCDF), and $\chi^*(s) = E[e^{-sX}] = \int_0^\infty e^{-st} d\chi(t)$ denotes its Laplace-Stieltjes Transform (LST).

4.2.3 Analysis of a single cache

We consider a TTL-based cache n in isolation. Requests arrive at the cache according at instants $\{t_k, k \geq 0\}$. Without loss of generality, we assume that the first request arrives at time $t_0 = 0$ and finds an empty cache. We denote by $X^{(k)}$ the inter-arrival time of the k-th and $(k+1)$-th requests. We also denote by $T^{(k)}$ the TTL duration set at the request instant $t_k, k \geq 0$.

To state the next results we need to strengthen the statistical assumptions made on the sequences $\{X^{(k)}, k \geq 0\}$ and $\{T^{(k)}, k \geq 0\}$.

Assumption 4.1 (Renewal arrivals and TTLs) *Both sequences* $\{X^{(k)}, k \geq 0\}$ *and* $\{T^{(k)}, k \geq 0\}$ *are independent renewal sequences i.e. successive inter-request times and TTLs at each cache are independent and identically distributed (i.i.d.) random variables. Morever TTLs are independent of the exogenous arrivals.*

Assumption 4.1 is general enough to cover a broad range of applications. In his earlier

work, Whitt [95] developed two basic methods to approximate a point process with a renewal process, and he showed in a joint work with Feldmann [35] that the long-tailed distributions which are generally observed in network performance analysis can be fitted by a renewal process with a hyper-exponential inter-arrival distribution. Also, Jung et *al.* [59] used renewal processes with Weibull and Pareto CDFs to fit the traces of DNS servers at the MIT Computer Science and Artificial Intelligence Laboratory and the Korea Advanced Institute of Science and Technology. More recently, Nelson and Gerhardt [77] have surveyed different methods used to fit a general point process to a special Phase-Type renewal process via *Moment Matching* techniques. In contrast to many existing works where it is assumed that the arrival process obeys to the Independent Reference Model (IRM) (or equivalently [38] that the arrival process is a Poisson process), our renewal assumption 4.1 is less restrictive.

We denote by $F_n(t)$ and $T_n(t)$ the CDFs of generic inter-arrival time $X^{(k)}$ and TTL duration $T^{(k)}$ at cache n. Define $L_n(t) := P(X^{(k)} < t, X^{(k)} < T^{(k)})$ the stationary probability that the inter-arrival time between two successive requests is smaller than t and smaller than the TTL associated with the former request. Because arrivals and TTLs are independent we have

$$L_n(t) = \int_0^t (1 - T_n(x))dF_n(x),\ t \geq 0. \tag{4.1}$$

Using notation in Table 4.1, the performance metrics at cache n are given as follows.

Proposition 4.1 (CCN Cache Performance with Renewal requests) *Under Assumption 4.1 the hit probability, hit rate and miss rate denoted by* $H_{P,n}$, $H_{R,n}$ *and* $M_{R,n}$, *respectively, are given by*

$$H_{P,n} = \int_0^\infty (1 - T_n(t))dF_n(t) = L_n(\infty), \tag{4.2}$$

$$H_{R,n} = \Lambda_n H_{P,n}, \quad M_{R,n} = \Lambda_n(1 - H_{P,n}). \tag{4.3}$$

Moreover, the occupancy probability $O_{P,n}$ *is derived as follows*

$$O_{P,n} = \Lambda_n E\left[\int_0^{X^{(0)}} (1 - T_n(t))dt\right]. \tag{4.4}$$

Proof The stationary hit probability $H_{P,n}$ is defined as the probability that an arriving request finds the data item in the cache, i.e. the TTL has not expired yet, namely,

$$H_{P,n} = P(X^{(k)} \leq T^{(k)}) = \int_0^\infty P(x \leq T^{(k)})dF_n(x) = \int_0^\infty (1 - T_n(x))dF_n(x).$$

The stationary miss probability is $M_{P,n} = 1 - H_{P,n}$ so that the miss rate is given by (4.3).

Let V be the time during which the document is in the cache between two consecutive request arrivals. We have $O_{P,n} = E[V]/E[X^{(0)}] = \Lambda_n E[V]$ by renewal theory. Let us find $E[V]$. Define the binary random variable $U(t)$ to be one if the document is in the cache at time t and

zero otherwise. Without loss of generality consider the interval $[0, X^{(0)}]$ corresponding to the inter-arrival time between the first and the second request. We have

$$E[V] = E\left[\int_0^{X^{(0)}} U(t)dt\right] = E\left[\int_0^{X^{(0)}} E[U(t)|X^{(0)}]dt\right] = E\left[\int_0^{X^{(0)}} (1 - T_n(t))dt\right]$$

where the last equality follows from $E[U(t)|X^{(0)}] = E[U(t)] = P(U(t) = 1) = P(T^{(0)} > t)$. ∎

Remark 4.1 (Stationary arrivals) *Proposition 4.1 is still valid even if the arrival process is a general stationary request process. This result is established in Chapter 2.*

We now evaluate the CDF $G_n(t)$ of the miss process which is needed to extend the analysis to a network of caches since a cache may receive requests due to misses at lower-level caches.

Proposition 4.2 (CCN Cache Miss Stream with Renewal requests) *Under Assumption 4.1 the miss process of at cache n taken in isolation is a renewal process. The CDF $G_n(t)$ of the inter-miss times is the solution of the integral equation*

$$G_n(t) = F_n(t) - L_n(t) + \int_0^t G_n(t-x)dL_n(x) \tag{4.5}$$

or, in compact form, $G_n = F_n - L_n + L_n \star G_n$ with $\star$ denoting the convolution operator. The LST $G^*(s)$ of the inter-miss times is given by

$$G_n^*(s) = \frac{F_n^*(s) - L_n^*(s)}{1 - L_n^*(s)}. \tag{4.6}$$

Proof Without loss of generality, assume that the first request arrives at time $t_0 = 0$ and finds an empty cache. Since a miss triggers a new TTL, miss times are regeneration points for the state of the cache under Assumption 4.1. This implies that miss instants form a renewal process which is fully characterized by the CDF $G_n(t)$ of the generic inter-miss time denoted by Y .

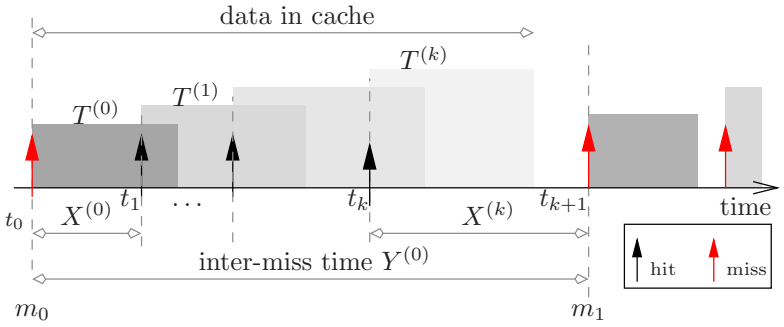


Figure 4.1: TTL-based model of content-router in CCN

The rest of the proof is an adaptation of a classical argument in renewal theory (see [22, Chapter 9]). Recall that $X^{(0)}$ (resp. $Y^{(0)}$, $T^{(0)}$) denotes the first inter-arrival time (resp. inter-miss time, TTL value) after $t_0 = 0$ as shown in Figure 4.1. Since $Y^{(0)} \geq X^{(0)}$ the event $\{Y^{(0)} < t\}$ may only occur if $X^{(0)} < t$. Therefore,

$$\begin{aligned} G_n(t) &= P(Y^{(0)} < t, T^{(0)} < X^{(0)} < t) + P(Y^{(0)} < t, T^{(0)} > X^{(0)}, X^{(0)} < t) \\ &= P(T^{(0)} < X^{(0)} < t) + E\left[G_n(t - X^{(0)})\mathbf{1}(T^{(0)} > X^{(0)}, X^{(0)} < t)\right] \\ &= P(X^{(0)} < t) - P(X^{(0)} < t, X^{(0)} < T^{(0)}) + \int_0^t G_n(t-x)P(T^{(0)} > x)dF_n(x) \quad (4.7) \end{aligned}$$

where we have used the independence of $X^{(0)}$ and $T^{(0)}$ to establish (4.7). Then it follows from equation (4.1) that

$$\begin{aligned} G_n(t) &= F_n(t) - L_n(t) + \int_0^t G_n(t-x)(1 - T_n(x))dF_n(x) \\ &= F_n(t) - L_n(t) + \int_0^t G_n(t-x)dL_n(x) \quad (4.8) \end{aligned}$$

The renewal equation (4.8) may be written $G_n = F_n - L_n + L_n \star G_n$. It is well known that its solution exists and is unique and is given by $G_n = R \star (F_n - L_n)$ [22, Theorem 2.3, p. 294] where $R = \sum_{k\geq 0} L_n^{(k)}$ and $L_n^{(k)}$ denotes the k^{th}-fold convolution of the function L_n with itself (by convention $L_n^{(0)} \equiv 1$).
From the identity $G_n = F_n - L_n + L_n \star G_n$ we readily get (4.6), which concludes the proof. ∎

Approximation for LRU caches

Che et *al.* [23] have experimentally shown that LRU caches fed with requests described by Poisson processes can be accurately modeled as deterministic TTL-based cache in isolation. In a 2013 paper, Martina et *al.* [72] have extended these experimental results [23] to the case of renewal request processes. The constant TTL value D is referred to in [23, 72] as the *characteristic time* of the LRU cache, and it is obtained by solving a fixed-point equation. Their results are theoretically established in Chapter 2. A similar fixed-point equation is derived in the case of finite cache capacity as shown in Section 4.2.8. Given a deterministic TTL value D_n at cache n, we have $T_n(t) = \mathbf{1}(t > D_n)$ and Equations (4.2), (4.3), (4.4) and (4.5) become

Corollary 4.1 (Deterministic TLLs)

$$H_{P,n} = F_n(D_n)\ ,\ M_{R,n} = \Lambda_n(1 - F_n(D_n))\ ,\ O_{P,n} = \Lambda_n \int_0^{D_n} (1 - F_n(x))dx \quad (4.9)$$

$$G_n(t) = \mathbf{1}(t > D_n)\left(F_n(t) - F_n(D_n) + \int_0^{D_n} G(t-x)dF_n(x)\right) \quad (4.10)$$

Remark 4.2 (Counters of Particles) *A single cache with a deterministic TTL is called a Geiger counter of Type II in [22, Example (1.34), p. 292].*

Approximation for RND caches

It was experimentally shown in [72] that RND caches can be studied as *memoryless* TTL-based caches with exponentially distributed TTLs. Also in this case, the expected TTL value μ^{-1} is solution of a fixed point equation as derived in Section 4.2.8 and proved in Chapter 2. In this case if $T_n(t) = 1 - e^{-\mu_n t}$ is the exponential TTL distribution at cache n then $L_n^*(s) = F_n^*(s + \mu)$, the metrics of interest and the miss process characterization follow directly from Equations (4.2), (4.3), (4.4) and (4.6).

Corollary 4.2 (Exponential TLLs)

$$H_{P,n} = F_n^*(\mu_n)\ ,\ M_{R,n} = \Lambda_n(1 - F_n^*(\mu_n))\ ,\ O_{P,n} = \frac{\Lambda_n(1 - F_n^*(\mu_n))}{\mu_n} \tag{4.11}$$

$$G_n^*(s) = \frac{F_n^*(s) - F_n^*(s + \mu_n)}{1 - F_n^*(s + \mu_n)}. \tag{4.12}$$

Optimality of a deterministic TTL-cache

Applying standard results about convex ordering and the formulas (4.2–4.4), we obtain the following interesting property for a deterministic TTL cache.

Proposition 4.3 *Given the expected TTL value* $D = E[T]$ *and the CDF* $F_n(t)$ *of inter-arrival times, the occupancy* $O_{P,n}$ *is maximized when the TTL is deterministic and equal to* D*. Moreover, if* $F_n(t)$ *is a concave function then the hit probability* $H_{P,n}$ *is maximized too.*

Proof We will show that the deterministic TTL distribution maximizes/minimizes our metrics of interest (i.e. the hit probability $H_{P,n}$ and the occupancy probability $O_{P,n}$) when the mean TTL value $D = E[T]$ is known. First, we prove the following lemma.

Lemma 4.1 (Convex ordering) *If* D *and* T *are respectively constant and random TTLs such that* $E[T] = D$*, then the following relation holds*

$$D \leq_{cx} T \tag{4.13}$$

where $\leq_{cx}$ *is the convex ordering.*

Proof [Lemma] The definition of convex ordering of random variables T_1 and T_2 says $T_1 \leq_{cx} T_2$ if and only if $E[\phi(T_1)] \leq E[\phi(T_2)]$ where $\phi(.)$ is a convex function. We shall show that this

convex ordering holds for any random TTL T and constant TTL D such that $E[T] = D$ in order to prove the lemma. For any random TTL $T \geq 0$ and any convex function $\phi(.)$, we have thank to Jensen's inequality :

$$E[\phi(T)] \geq \phi(E[T]) = \phi(D) = E[\phi(D)]$$

The last equality follows from the fact that $\phi(D)$ is a constant ∎

Now, we resume the proof of our proposition. We assume that the TTLs $\{T^{(k)}\}_{k\geq 0}$ are sampled from a general distribution $T(t)$ such that $E[T] = D$. Observe that the occupancy $O_{P,n}(T)$ and hit probability $H_{P,n}(T)$ are functions of the timer T which can be written as following:

$$O_{P,n}(T) = \Lambda_n E\left[\phi(T)\right] \ , \ H_{P,n}(T) = E[F_n(T)]$$

where $F_n(x)$ is the CDF of X and $\phi(t) = \int_0^t (1 - F_n(x))dx$. The second derivative of $\phi(t)$ is $\phi''(t) = -F_n'(t) \leq 0$ because $F_n'(t)$ is a probability density function; hence, $\phi(t)$ is a concave function for any $F_n(x)$. Then by applying Lemma 4.1, it follows that $O_{P,n}(T) \leq O_{P,n}(D)$ for any timer T such that $E[T] = D$. Meanwhile, if $F_n(x)$ is concave, Lemma 4.1 states that $H_{P,n}(D) \geq H_{P,n}(T)$. ∎

We note that if the request process is a Poisson process, the occupancy $O_{P,n}$ and the hit probability $H_{P,n}$ are equals and theses metrics are maximized when the TTL is deterministic.

Proposition 4.3 theoretically explains the superiority, in terms of hit and occupancy probabilities, of LRU caches over RND caches (given that $D = \mu^{-1}$) when they are fed by IRM traffic (or Poisson processes [38]) or traces in [35, 59] fitted by renewal processes. We can easily check that the CDFs $F_n(t)$ of inter-arrival times in these experiments are concave functions. Since a cache in isolation is completely analyzed, we can now investigate inter-connected caches.

4.2.4 Exact analysis on hierarchical TTL-based cache networks

In this section we present a class of cache networks for which an exact analysis can be derived from a simple extension of arguments introduced for single caches. Then we analyze TTL-based cache networks fed by exogenous renewal request processes. We assume each cache has an infinite buffer; also processing and transmission delays in the network are negligible. Hence, files at each cache are decoupled and can be separately studied.

Therefore, we focus on a single file whose requests are propagated along a tree, but it is easy to generalize to a polytree. In fact, the case where requests are routed on a polytree is derived from the tree analysis by applying the results on *dependent thinning of a point process* by Isham [52] at each position in the network where requests are forwarded between two or more destinations. We draw the attention of the reader to the fact the cache network topology does not have to be a tree (or a polytree); in fact, it can be general because files are decoupled

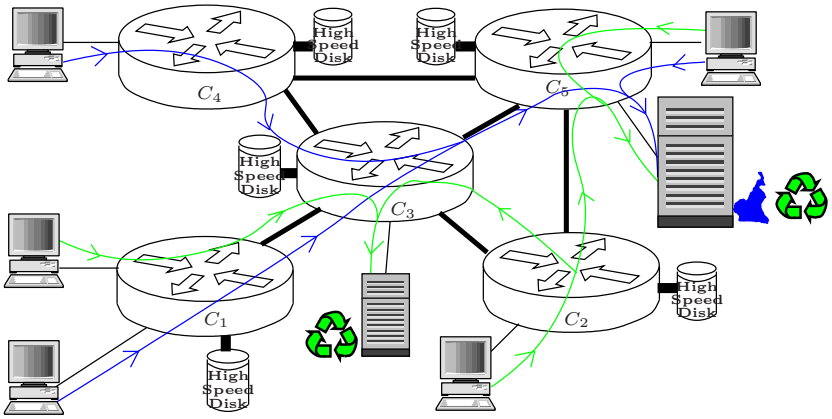


Figure 4.2: Five caches network with planar graph topology.

and requests can be independently routed (see the example in Figure 4.2 where requests for the *green* file are routed on as polytree and that of the *blue* file are routed as tree).

We consider then a tree of caches: the root is connected to the server, each other cache has a parent cache to which it forwards all the requests which cannot be satisfied locally. Once located, the data item is routed on the reverse-path and a copy is placed in each cache. Each cache operates exactly as described in the previous section, by setting a TTL for each new data item stored and redrawing the TTL at each cache hit. We recall that $\mathcal{C}(n)$ is the set of children of cache n.
The following assumption holds:

Assumption 4.2 (TTL values) *TTL values extracted at each cache are i.i.d. values. We recall that* $T_n(t)$ *is the CDF of TTL values at cache* n*. TTL values at different caches are independent and they are also independent from the request arrival processes.*

The ICN content-routers [3] are examples of caches that behave independently of other caches and of the requests they receive. They also decide locally what content to store or discard at least as described in the paper of Van Jacobson et *al.* [53].

Each cache, say cache n, receives two flows of requests: users' requests arriving directly at a cache are called *exogenous* and form the *exogenous* arrival process, misses at the children caches in $\mathcal{C}(n)$ form the *endogenous* arrival process. We generalize Assumption 4.1 as follows:

Assumption 4.3 (exogenous request streams are renewal processes) *The exogenous arrival processes are independent renewal processes. We denote by* $F_n(t)$ *and* λ_n *the CDF of the inter-arrival times and the rate of exogenous requests at cache* n *respectively.*

The superposition of the exogenous and endogenous arrival processes at cache n forms the *aggregated* arrival process. We introduce $H_n(t)$ and Λ_n to denote respectively its inter-arrival

time CDF and its rate. The notation is summarized in Table 4.1. The exact analysis we carried out on caches in isolation can be extended immediately to the case of *linear-star networks*, that we study below.

Linear network A linear cache network is a tandem of caches and one server/disk, where exogenous requests arrive only to the cache farthest from the server/disk as illustrated in Figure 4.3. Since the arrival process at the first cache is a renewal process (Assumption 4.1) the miss process of the first cache is also a renewal process (Proposition 4.2). Therefore, the second cache is fed by a renewal process. Reasoning in this way iteratively, we can then show that all the caches are fed by a renewal process. Moreover, all the metrics of interest can be derived successively at each cache from the results on a single cache analysis in Section 4.2.3.

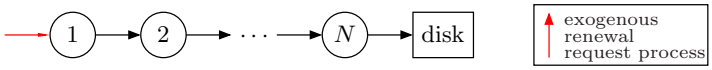


Figure 4.3: Linear cache network.

Example: Tandem of exponential TTL-based caches.

Consider the linear cache network in Figure 4.3 composed of N TTL caches labeled $n = 1, 2, \ldots, N$. Requests arrive only at cache 1 according to a renewal process with arrival rate λ_1 and CDF of inter-request times $F_1(t)$. We can then apply recursively the results obtained for a single cache. In particular, if we denote by $G^*_n(s)$ the LST of the inter-miss times at cache n, we may apply formula (4.12) where the LST of the inter-arrival times is $G^*_{n-1}(s)$. We obtain

$$G^*_n(s) = \frac{G^*_{n-1}(s) - G^*_{n-1}(s+\mu_n)}{1 - G^*_{n-1}(s+\mu_n)} \tag{4.14}$$

for $n = 1, \ldots, N$, where $G^*_0(s) = F^*_1(s)$. At the cache n the hit probability $H_{P,n} = G^*_{n-1}(\mu_n)$, the miss rate $M_{R,n} = M_{R,n-1}(1 - H_{P,n})$ and hit rate $H_{R,n} = M_{R,n-1}H_{P,n}$ are derived by using (4.14)

$$M_{R,n} = \lambda_1 \prod_{i=0}^{n-1} (1 - G^*_i(\mu_{i+1})), \tag{4.15}$$

$$H_{R,n} = \lambda_1 \prod_{i=0}^{n-2} (1 - G^*_i(\mu_{i+1}))\; G^*_{n-1}(\mu_n), \tag{4.16}$$

with $M_{R,0} := \lambda_1$. The occupancy of cache n is given by applying (4.4) with $F_1(t) = G_{n-1}(t)$ and $\lambda_1 = M_{R,n-1}$, hence

$$O_{P,n} = M_{R,n-1} \left(\frac{1 - G^*_{n-1}(\mu_n)}{\mu_n} \right). \tag{4.17}$$

Star network A star cache network is a tree with one internal node i.e. the root and leaves (see Figure 4.4). For this two-level tree, the metrics of interest and the miss process are easily found at each leaf from the single cache analysis in Section 4.2.3. The miss processes are also renewal processes since the request processes at the leaves are renewal processes. Hence, the root is fed by the aggregated request process resulting from the superposition of the (renewal) miss processes and its exogenous renewal process. It is possible to calculate exactly the CDF of the first inter-arrival time in the aggregated arrival process (see Theorem 4.1 and following remarks), then the metrics of interest are obtained from Proposition 4.1 since the aggregated process is a stationary process (See Remark 4.1). Our analysis provides exact results on star networks of caches.

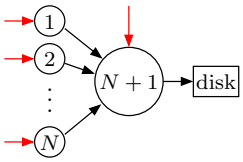


Figure 4.4: Star cache network.

Example: Two-level tree of exponential TTL-based cache.

Consider the tree network in Figure 4.4 with one root (labeled $N+1$) and N children (leaves) labeled $n = 1, 2, \ldots, N$. The exogenous requests arrive at the n-th node according to a Poisson process with rate λ_n. While the TTL is exponentially distributed with rate μ_n at the n-th leave, we assume an arbitrary CDF $T_{N+1}(t)$, with LST $T^*_{N+1}(s)$ for the TTL at the root. Using the results in Section 4.2.3 to study caches $n = 1, 2, \ldots N$ in isolation, and Theorem 4.1 to calculate the CDF of the overall request inter-arrival time at cache $N+1$, we can derive all metrics of interest [26, Sect. 3.3].

Linear-star network We generalize our exact approach on the two previous networks topologies by defining a class of networks called *Linear-star cache network* illustrated in Figure 4.5. We can characterize the exact performance metrics on any network that belongs to this class as follows: we start from the leaves and apply Propositions 4.1 and 4.2 (as was done for the linear network), until we reach the root where we apply Theorem 4.1 and Proposition 4.1 (as it was described for the star network).

4.2.5 Approximated methodology for general tree networks

The approach we described in Section 4.2.4 cannot be extended to arbitrary hierarchical networks. The problem arises from the fact that the aggregated arrival process is not in general a renewal process (this is not the case even if the exogenous and endogenous arrival processes

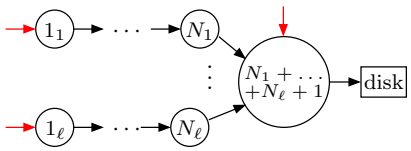


Figure 4.5: Linear-star cache network.

are both renewal ones). Hence, we cannot apply Proposition 4.2 that allows us to characterize the exact miss process. Indeed in the linear-star cache network we cannot extend our analysis beyond the cache with more than one child. Nevertheless, for our analysis we will suppose that the aggregated request process is a renewal process and we make the following approximation:

Approximation 4.1 (Overall Process) *The overall (aggregated) arrival process at node* n *is approximated by a renewal process with inter-arrival time CDF* $H_n(t)$.

Note that the statement above has a different status than the other assumptions in this chapter. While the assumptions can be considered approximations for actual cache networks, they are internally consistent. On the contrary the statement in Approximation 4.1 is in general false, even in the framework of our model. Nevertheless, it makes the analysis possible and leads to excellent approximations as we are going to show later.

We have shown in Section 4.2.3 that the miss process of a TTL cache fed by a renewal process is itself a renewal process, then a corollary of Approximation 4.1 is that each miss process can be considered a renewal process. In this case, the aggregated arrival process is the superposition of independent (due to the tree topology of the network) renewal processes and the CDF $H_n(t)$ of inter-arrival times has been calculated by Lawrence [69, Formula (4.1)].

Theorem 4.1 *The CDF* $A(t)$ *of the first inter-event time of the point process resulting from the superposition of* K *independent renewal processes is given by*

$$A(t) = 1 - \sum_{k=1}^{K} \frac{\alpha_k}{\sum_{l=1}^{K} \alpha_l} (1 - A_k(t)) \prod_{j=1, j\neq k}^{K} \alpha_j \int_t^{\infty} (1 - A_j(u))\, du,$$

where $A_k(t)$ *and* $\alpha_k > 0$ *are respectively the inter-event time CDF and the arrival rate of the* k^{th} *process.*

Theorem 4.1 actually holds for the superposition of independent stationary point processes [10, Formula (1.4.6), p. 35]. Note in passing that such a superposition is itself a stationary process, which allows us to use Proposition 4.1 to compute the hit probability, miss rate and cache occupancy, respectively, of a cache fed by the superposition of independent stationary processes (see Remark 4.1), and then in particular of renewal ones.

Thanks to Approximation 4.1 and Theorem 4.1, we are ready to study any cache tree network. The total request rate Λ_n at a node n, is

$$\Lambda_n = \lambda_n + \sum_{i \in \mathcal{C}(n)} M_{R,i} \tag{4.18}$$

where $\mathcal{C}(n)$ is the set of children of node n. Since the exogenous process (with CDF $F_n(t)$) and the miss process at the i-th child of node n (with CDF $G_i(t)$) are renewal processes, we invoke Theorem 4.1 to compute the approximate inter-arrival times CDF $H_n(t)$ of the overall arrival process. We get

$$\begin{aligned} H_n(t) &= 1 - \frac{\lambda_n}{\Lambda_n}\bar{F}_n(t) \prod_{i \in \mathcal{C}(n)} M_{R,i} \int_t^\infty \bar{G}_i(u)du \\ &\quad - \sum_{i \in \mathcal{C}(n)} \frac{M_{R,i}}{\Lambda_n}\bar{G}_i(t)\lambda_n \int_t^\infty \bar{F}_n(u)du \times \prod_{\substack{j \in \mathcal{C}(n) \\ j \neq i}} M_{R,j} \int_t^\infty \bar{G}_j(u)du. \end{aligned} \tag{4.19}$$

The approximate inter-miss times CDF $G_n(t)$ at cache n is obtained from Proposition 4.2 since we approximate the overall request process by a renewal process with CDF $H_n(t)$

$$G_n(t) = H_n(t) - L_n(t) + \int_0^t G_n(t-x)dL_n(x) \tag{4.20}$$

where $L_n(t) = \int_0^t (1 - T_n(x))dH_n(x)$ and $T_n(t)$ is the CDF of the TTL duration at cache n.

Equations (4.18), (4.19) and (4.20) provide a recursive procedure for calculating, at least numerically, the request rate Λ_n and the approximate CDFs $G_n(t)$ and $H_n(t)$ at each cache n of an arbitrary hierarchical network starting from the leaves. The approximate metrics of interest

are obtained from Proposition 4.1. Our approach is summarized in the following algorithm:

Algorithm 4: General procedure on tree fed by renewal request processes

input : TreeDepth d, CDFs $F_n(t)$, $\{G_i(t), i \in \mathcal{C}(n)\}$ and $\{T_n(t), n \geq 1\}$

output: Metrics of interest Λ_n, $H_{P,n}$, $O_{P,n}$ and CDF $G_n(t)$

while $d \neq 0$ **do** ; // Caches are different from the server

  foreach n in the set of caches at depth d **do** ; // Start from Leaves

    $\Lambda_n \xleftarrow{\text{Eq(4.18)}} \{\lambda_n, M_{R,i}, i \in \mathcal{C}(n)\}$;

    if $\mathcal{C}(n) = \emptyset$ **then**

      $H_n(t) \leftarrow F_n(t)$;

    else

      $H_n(t) \xleftarrow{\text{Eq(4.19)}} \{F_n(t),\ G_i(t), i \in \mathcal{C}(n)\}$;

    end

    $H_{P,n},\ M_{R,n},\ O_{P,n} \xleftarrow{\text{Prop.(4.1)}} \{H_n(t),\ T_n(t)\}$;

    $G_n(t) \xleftarrow{\text{Eq(4.20)}} \{H_n(t),\ T_n(t)\}$;

  end

  $d \leftarrow d - 1$;

end

While this algorithm allows us to study any cache tree under any possible exogenous arrival processes and TTL distributions, its numerical complexity can be very high as it requires to evaluate some integrals over infinite ranges as in (4.19) and to solve an integral equation as in (4.20). As we are going to show, simpler algorithms exist for more specific distributions.

Hierarchical networks of with exponential TTL-based caches

In order to show that this complexity can be significantly reduced, we focus on a particular class of tree networks, class $\mathcal{N}$ having a *caterpillar tree* topology (see Figure 4.6) and we give explicit results. The TTL at each node n is exponentially distributed with rate μ_n. A network belongs to class $\mathcal{N}$ if, in addition to **A1**, the following approximation holds:

Approximation A2: The node n is fed by the superposition of two independent request arrival processes: one (*stream 1*) is the miss rate of a child of cache n and is a generic renewal process and the other one (*stream 2*) is an exogenous renewal process with CDF of the form

$$K_n(t) = 1 - \sum_{m=0}^{M_n} \alpha_{n,m} e^{-\beta_{n,m} t} \tag{4.21}$$

where $0 \leq M_n < \infty$ and $\{\beta_{n,m}\}_m$ is a set of non negative numbers.

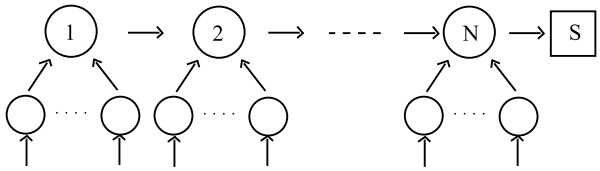


Figure 4.6: Caterpillar tree networks

In what follows we assume without loss of generality that stream 1 originates from a cache child labeled $n-1$. Then we denote the CDF of the inter-miss times in stream 1 as $G_{n-1}(t)$ and the miss rate as $M_{R,n-1}$. From (4.21) the arrival rate of stream 2 is $\lambda_n := \sum_{m=0}^{M_n} \alpha_{n,m}\beta_{n,m}$, the total arrival rate at node n is $\Lambda_n = M_{R,n-1} + \lambda_n$. Approximations **A1** and **A2** together yield the following procedure for approximating $G_n^*(s)$ and $H_n^*(s)$.

Proposition 4.4 (Approximation for class $\mathcal{N}$) *Under Approximations* **A1** *and* **A2**, *the LST of inter-arrival times of the aggregated request process at each node n is given by,*

$$H_n^*(s) = 1 - s\frac{\lambda_n}{\Lambda_n}\sum_{m=0}^{M_n}\frac{\alpha_{n,m}}{s+\beta_{n,m}} - s^2\frac{\lambda_n M_{R,n-1}}{\Lambda_n}\sum_{m=0}^{M_n}\frac{\alpha_{n,m}(1-G_{n-1}^*(s+\beta_{n,m}))}{(s+\beta_{n,m})^2\beta_{n,m}} \tag{4.22}$$

and the LST of inter-miss times is

$$G_n^*(s) = \frac{H_n^*(s) - H_n^*(s+\mu_n)}{1 - H_n^*(s+\mu_n)}. \tag{4.23}$$

Proof Applying Theorem 4.1 to $G_{n-1}(t)$ and $K_n(t)$ at cache n, we can approximate the CDF of the inter-arrival times by (thanks to Approximation 4.1)

$$H_n(t) = 1 - \frac{\eta_n M_{R,n-1}}{\Lambda_n}\sum_{m=0}^{M_n}\alpha_{n,m} \times \left(\bar{G}_{n-1}(t)\int_t^\infty e^{-\beta_{n,m}u}du + e^{-\beta_{n,m}t}\int_t^\infty \bar{G}_{n-1}(u)du\right)$$

from which we deduce (4.22). (4.23) is obtained from (4.12). ∎

The hit probability is simply $H_{P,n} = H_n^*(\mu_n)$, the total arrival rate $\Lambda_n = M_{R,n-1} + \lambda_n$ and the miss rate $M_{R,n} = \Lambda_n(1 - H_n^*(\mu_n))$ are

$$M_{R,n} = \sum_{i=1}^{n}\lambda_i\prod_{j=i}^{n}(1 - H_j^*(\mu_j)), \tag{4.24}$$

$$\Lambda_n = \sum_{i=1}^{n-1}\lambda_i\prod_{j=i}^{n-1}(1 - H_j^*(\mu_j)) + \lambda_n, \tag{4.25}$$

Relations (4.22)-(4.23) and (4.24)-(4.25) provide a recursive procedure for calculating Λ_n and $H_n^*(\mu_n)$ for each n, from which we obtain approximations for the hit probability, hit rate,

miss rate and stationary occupancy at node n:

$$H_{P,n} = H_n^*(\mu_n), \;\; H_{R,n} = \Lambda_n H_n^*(\mu_n), \;\; M_{R,n} = \Lambda_n(1 - H_n^*(\mu_n)), \;\; O_{P,n} = \Lambda_n \left(\frac{1 - H_n^*(\mu_n)}{\mu_n} \right).$$

The latter result follows from (4.11). In [26], we show that Approximation **A2** is verified on the network in Figure 4.6 when TTLs are exponentially distributed and exogenous request processes are Poisson.

Hierarchical networks with Matrix-Exponential request inter-arrival times and TTL

We consider a hierarchical cache network where the inter-arrival times of exogenous requests and the TTL values are described by Matrix-Exponential (ME) distributions, i.e. whose CDFs and PDFs are defined by

$$\Psi(t) = 1 - \boldsymbol{\alpha}\, e^{St}\, \mathbf{1}_R \,, \; \psi(t) = \boldsymbol{\alpha}\, e^{St}\, (-S\mathbf{1}) \quad t \geq 0 \tag{4.26}$$

respectively [46], where $\boldsymbol{\alpha}$ is a 1-by-R vector, called the starting vector, S is an R-by-R matrix, called the progress rate matrix, and $\mathbf{1}_R$ is an R-by-1 vector whose elements are all equal to 1. In general different pairs $(\boldsymbol{\alpha}, S)$ can lead to the same CDF $\Psi(t)$.

Here, we consider a representation with minimal order R. In [46, Theorem 3.1], He and Zhang established under which conditions R is minimal and they showed that in this case the matrix S is a Jordan matrix and Ψ can be written as follows (Karlin's representation [61]):

$$\Psi(t) \;\; = \;\; 1 - \sum_{k=1}^{K} Q_k(t) e^{\sigma_k t}, \; t \geq 0 \tag{4.27}$$

where $\{\sigma_k\}_k$ are the eigenvalues of S, $\sigma_i \neq \sigma_j$ if $i \neq j$, $Q_k(t) = \sum_{j=0}^{r_k - 1} q_{k,j} t^j$ is a polynomial of degree $r_k - 1$ and $\sum_{k=1}^{K} r_k = R$. The relations between $\boldsymbol{\alpha}$ and $\{q_{k,j}, 1 \leq k \leq K, 0 \leq j \leq r_k - 1\}$ can be found in [46]. In what follows we will usually consider Karlin's representation (4.27). The class of ME distributions is equivalent to the class of distributions having a rational LST [46], it includes then also all the phase-type distributions.

In what follows we are going to call a request renewal process with ME distributed inter-arrival times simply an ME renewal process. Similarly, we are going to use the expression ME TTL to indicate TTLs that are ME distributed. The following result guarantees us that if the request arrival process at a cache is an ME renewal process and TTL are ME distributed, then the miss process is also a ME renewal process with a known representation.

Proposition 4.5 (ME miss process) *If the TTLs and the inter-arrival times of the request renewal process at cache n are ME distributed, then the miss inter-arrival times are ME distributed.*

Proof We consider a cache n where the inter-arrival times and the TTLs are characterized by the ME CDFs $F_n(t)$ and $T_n(t)$. Both $F_n(t)$ and $T_n(t)$ admit a Karlin representation and a rational LST. From the definition (4.1) also $L(t)$ has a Karlin representation and its LST is rational. Thus, the solution $G_n(t)$ in (4.5) is a CDF with a rational LST $G_n^*(s)$ given by

$$G_n^*(s) \quad = \quad 1 - \frac{1 - F_n^*(s)}{1 - L_n^*(s)} = 1 - \frac{N(s)}{D(s)}$$

where $N(s)$ and $D(s)$ are the numerator and the denominator of the fraction $1 - G_n^*(s)$ after factorization and simplification of common terms. The CDF $G_n(t)$ is a ME distribution by the equivalence between ME distributions and CDFs having rational LST. Moreover, $G_n(t)$ admits a Karlin representation:

$$G_n(t) = 1 - \sum_{k=1}^{K} \sum_{j=0}^{r_k - 1} q_{k,j} t^j e^{\sigma_k t}, \; t > 0 \tag{4.28}$$

where the exponents $\{\sigma_k\}_k$ are the zeros of $D(s)$ and the coefficients $\{r_k, q_{k,j}\}_{k,j}$ are given by the relations

$$q_{k,r_k - i} \quad = \quad \frac{1}{i!} \left\{ \frac{d^i}{ds^i} \left[(1 - G^*(s))(s - \sigma_k)^{r_k} \right] \right\}_{s=\sigma_k}. \tag{4.29}$$

■

In general the aggregated request arrival process at cache n is the superposition of the miss processes at the cache in $\mathcal{C}(n)$ and the exogenous arrival process. If each of these processes is a ME renewal process, Approximation 4.1 and Theorem 4.1 allows us to conclude that also the inter-arrival times of the aggregated request arrival process are ME distributed. Under the Approximation 4.1 and Proposition 4.5, all the miss processes in the network are ME renewal processes. We can characterize them iteratively starting from the leaves as for the general case.

Hierarchical networks with Diagonal Matrix-Exponential distributions

The calculations become even simpler when the progress rates matrices (S) of the ME distributions are diagonal or diagonalizable. In this case, the distribution is said to be *Diagonal Matrix-Exponential* and we note diag.ME. Without loss of generality, if $\Psi(t)$ is a diag.ME distribution, then its Karlin representation is:

$$\Psi(t) = 1 - \sum_{j=1}^{K} q_j e^{\sigma_j t}.$$

If the request arrival process at cache n is a diag.ME renewal process and the TTLs are diag.ME distributed, respectively with CDFs

$$F_n(t) = 1 - \sum_{k=1}^{K_n} a_{n,k} e^{-\sigma_{n,k} t} \quad \text{and} \quad T_n(t) = 1 - \sum_{j=1}^{J_n} b_{n,j} e^{-\mu_{n,j} t},$$

then the metrics of interests at cache n are obtained from a straightforward calculation by applying Proposition 4.1.

Corollary 4.3 (Metrics of interests at a diag.ME TTL cache) *The request rate λ_n, the hit probability $H_{P,n}$, the miss rate $M_{R,n}$ and the occupancy $O_{P,n}$ at cache n are calculated with the formulas*

$$\lambda_n = \sum_{k=1}^{K_n} a_{n,k}\sigma_{n,k}, \qquad H_{P,n} = \textstyle\sum_{j=1}^{J_n} b_{n,j}F_n^*(\mu_{n,j}) \tag{4.30}$$

$$M_{R,n} = \lambda_n \left(1 - \sum_{j=1}^{J_n} b_{n,j}F_n^*(\mu_{n,j})\right), \quad O_{P,n} = \lambda_n \textstyle\sum_{j=1}^{J_n} b_{n,j}\left(\frac{1-F_n^*(\mu_{n,j})}{\mu_{n,j}}\right). \tag{4.31}$$

Similarly, the miss process is characterized by applying Propositions 4.2 and 4.5.

Corollary 4.4 (Miss process at a diag.ME TTL cache) *The LST of inter-miss times CDF is*

$$G_n^*(s) = 1 - (1 - F_n^*(s)) \left(1 - \sum_{j=1}^{J_n} b_{n,j}F_n^*(s + \mu_{n,j})\right)^{-1} \tag{4.32}$$

which can be inverted as

$$G_n(t) = 1 - \left[\sum_{k=1}^{K_n} a_{n,k}\left(1 + \sum_{i=1}^{K_n\times J_n} \frac{\gamma_i}{\theta_i - \sigma_{n,k}}\right) e^{-\sigma_{n,k}t} + \sum_{i=1}^{K_n\times J_n} \frac{-\gamma_i}{\theta_i}\left(1 + \sum_{k=1}^{K_n} \frac{a_{n,k}\sigma_{n,k}}{\theta_i - \sigma_{n,k}}\right) e^{-\theta_i t}\right] \tag{4.33}$$

where $(\theta_i)_{1\le i\le K_n\times J_n}$ are solutions of the algebraic equation in z

$$0 = 1 - \sum_{i=1}^{K_n\times J_n} \frac{\delta_i}{\eta_i - z}, \tag{4.34}$$

$(\gamma_i)_{1\le i\le K_n\times J_n}$ is the vector solution of the linear system of algebraic equations:

$$\left\{0 = 1 + \sum_{i=1}^{K_n\times J_n} \frac{\gamma_i}{\theta_i - \eta_l}\right\}_{l=1}^{K_n\times J_n}, \tag{4.35}$$

the constants δ_i and η_i are given by

$$\delta_i = a_{n,k}b_{n,j}\sigma_{n,k} \quad , \quad \eta_i = \sigma_{n,k} + \mu_{n,j} \tag{4.36}$$

and (k,j) is the i^{th} couple according to some ordering of the product set $\{1,\dots,K_n\}\times\{1,\dots,J_n\}$. Note also that the CDF $G_n(t)$ is a diag.ME distribution.

When we superpose diag.ME renewal processes, the inter-arrival times of the superposed process are still diag.ME distributed. In particular if G_i is the diag.ME miss process at cache $i \in \mathcal{C}(n)$, with CDF

$$G_i(t) = 1 - \sum_{k=1}^{K_i} a_{i,k} e^{-\sigma_{i,k} t}.$$

the overall arrival process is then characterized in the following corollary of Proposition 4.5:

Corollary 4.5 (Overall Request Process at diag.ME TTL cache) *Under Assumptions 4.2 and 4.1, the CDF of inter-arrival times $H_n(t)$ in the overall request process at cache n is given by*

$$H_n(t) = 1 - \sum_{i \in \mathcal{C}(n) \cup \{n\}} \sum_{k=1}^{K_i} \frac{a_{i,k}}{\Lambda_n} \left(\lambda_n \times \prod_{j \in \mathcal{C}(n)} M_{R,j} \right) e^{-\sigma_{i,k} t} \prod_{\substack{j \in \mathcal{C}(n) \cup \{n\} \\ j \neq i}} \left(\sum_{k=1}^{K_j} \frac{a_{j,k}}{\sigma_{j,k}} e^{-\sigma_{j,k} t} \right) \quad (4.37)$$

where $M_{R,i}$ is the miss rate of the i^{th} child node, λ_n is the rate of exogenous requests, and Λ_n is the total request rate at the cache n.

The Algorithm 4 simplifies when all the TTLs and the exogenous request arrival processes are diag.ME and becomes:

Algorithm 5: Efficient Procedure on diag.MED cache tree fed by diag.ME renewal processes

input : `TreeDepth` d, `CDFs` $F_n(t)$, $\{G_i(t), i \in \mathcal{C}(n)\}$ `and` $\{T_n(t), n \geq 1\}$
output: `Metrics of interest` Λ_n, $H_{P,n}$, $O_{P,n}$ `and CDF` $G_n(t)$

while $d \neq 0$ **do** ; `// Caches are different from the server`
  foreach n `in the set of caches at depth` d **do** ; `// Start from Leaves`
    $\Lambda_n \xleftarrow{\text{Eq.(4.18)}} \{\lambda_n, M_{R,i}, i \in \mathcal{C}(n)\}$;
    if $\mathcal{C}(n) = \emptyset$ **then**
      $H_n(t) \leftarrow F_n(t)$;
    else
      $H_n(t) \xleftarrow{\text{Eq.(4.37)}} \{F_n(t),\ G_i(t), i \in \mathcal{C}(n)\}$;
    end
    $H_{P,n},\ M_{R,n},\ O_{P,n} \xleftarrow{\text{Cor.(4.3)}} \{H_n(t),\ T_n(t)\}$;
    $G_n(t) \xleftarrow{\text{Cor.(4.4)}} \{H_n(t),\ T_n(t)\}$;
  end
  $d \leftarrow d - 1$;
end

Before concluding our theoretical analysis and moving to the validation of our approximations, we observe that it is possible to adapt the formulas above for diag.ME exogenous processes and TTLs to consider a slightly larger class of networks—previously introduced as class $\mathcal{N}$ networks—that extends hierarchical diag.ME cache networks as follows: a single exogenous request process is allowed to be a renewal process with a general CDF (not necessarily a diag.ME distribution) for inter-arrival times. We do not develop the general procedure for class $\mathcal{N}$, but we show how the formulas change for a specific case.

Consider that the miss request process of cache i, a child node of cache n, is a general renewal process with CDF $G_i(t)$ and rate $M_{R,i}$, while the exogenous request process at cache n has diag.ME CDF $F_n(t) = 1 - \mathbf{a}_n e^{-A_n t}\mathbf{1}_{K_n}$ and arrival rate $\lambda_n = (\mathbf{a}_n(-A_n)^{-1}\mathbf{1}_{K_n})^{-1}$. The total request rate at node n is $\Lambda_n = M_{R,i} + \lambda_n$. The following proposition generalizes the CDF of inter-arrival times of the aggregated request process $H_n(t)$.

Corollary 4.6 (Generalization of class $\mathcal{N}$ networks) *The performance metrics at cache n of a class $\mathcal{N}$ network are obtained from Corollary 4.3, and the overall request process is characterized by the CDF $H_n(t)$ in (4.19) whose LST $H_n^*(s)$ is given by Proposition 4.4.*

Proof Applying the (4.19) to the CDFs $G_i(t)$ and $F_n(t) = 1 - \sum_{k=1}^{K_n} a_{n,k} e^{-\sigma_{n,k} t}$, we obtain

$$H_n(t) \;=\; 1 - \frac{\lambda_n M_{R,i}}{\Lambda_n} \sum_{k=1}^{K_n} a_{n,k} \times \left(\bar{G}_i(t) \int_t^\infty e^{-\sigma_{n,k} u} du + e^{-\sigma_{n,k} t} \int_t^\infty \bar{G}_i(u) du \right).$$

Taking the LST of the latter equation, the metrics of interest and the LST $G_n^*(s)$ are obtained by replacing the LST $F_n^*(s)$ by $H_n^*(s)$ in Corollary 4.3 and Corollary 4.4. ∎

4.2.6 Validation and numerical results

In this section, we investigate the accuracy of Approximation 4.1 and then of the approximate results obtained through Algorithms 4 and 5. We recall that Approximation 4.1 consists in considering that all aggregated request processes are renewal processes.

First, we show that in a tandem of two caches the first autocorrelation lag ($\mathtt{ACF}_1$) of the aggregated process at node 2 is quite small. We calculate it using using the formula in [69, Eq.(6.4)]. This autocorrelation lag $\mathtt{ACF}_1$ depends on the arrival rates λ_1 and λ_2 and the timer μ_1. We find that for any possible choice of these parameters $0 > \mathtt{ACF}_1 > -0.015$. Simulation results show that the autocorrelation is even less significant at larger lags. Therefore, inter-arrival times are weakly coupled and Approximation 4.1 is indeed accurate in this small network scenario.

Second, we evaluate the approximation quality by simulations in more complex configurations. We focus on networks of exponentially distributed TTL-based caches fed by requests generated according to Poisson processes for which it is possible to carry on an exact analysis;

then we look at a tree of deterministic TTL-based caches also fed by Poisson requests; and finally we investigate the situation where requests are described by more general renewal processes and TTL distributions.

Poisson traffic and exponential timers

We start by observing that when the exogenous request processes are Poisson processes, it is possible to model a tree network of N caches as an irreducible continuous time Markov process, with state $\mathbf{x}(t) = (x_1(t), \dots, x_N(t)) \in \mathcal{E} = \{0,1\}^N$, where $x_n(t) = 1$ (resp. $x_n(t) = 0$) if the data item is present (resp. missing) at time t at node n. Once the steady-state probabilities $(\pi(\mathbf{x}),\ \mathbf{x} \in \mathcal{E})$ have been calculated, exact values of the performance metrics of interest are obtained. For example, the stationary occupancy probability of cache n is $O^{\mathcal{M}}_{P,n} = \sum_{\mathbf{x}\in\mathcal{E}|x_n=1} \pi(\mathbf{x})$ (the superscript "$\mathcal{M}$" stands for "Markov"). For a line of caches, the hit probability and the miss rate at cache 1 are respectively $H^{\mathcal{M}}_{P,1}(1) = \pi(1,*)$ and $M^{\mathcal{M}}_{R,1} = \lambda_1\pi(0,*)$, while for cache 2 we have

$$H^{\mathcal{M}}_{P,2} = \frac{\lambda_1\pi(0,1,*) + \lambda_2(\pi(0,1,*) + \pi(1,1,*))}{\lambda_1(\pi(0,0,*) + \pi(0,1,*)) + \lambda_2}\ ,\ M^{\mathcal{M}}_{R,2} = \lambda_1\pi(0,0,*) + \lambda_2(\pi(0,0,*) + \pi(1,0,*))$$

where $\pi(i,*) = \sum_{x_2,\dots,x_N\in\{0,1\}} \pi(i,x_2,\dots,x_N)$ and $\pi(i,j,*) := \sum_{x_3,\dots,x_N\in\{0,1\}} \pi(i,j,x_3,\dots,x_N)$ are the stationary probabilities that cache 1 is in state $i \in \{0,1\}$ and caches $(1,2)$ are in state $(i,j) \in \{0,1\}^2$, respectively. Due to space constraints we omit the general expressions for these quantities for a generic tree of caches. Throughout Section 4.2.6, we compare the results of our models against the exact ones obtained by studying the Markov process.

Nine caches linear network This network architecture is chosen for its depth and its small number of leaves. We aim at evaluating the quality of Approximation 4.1 when the depth of the network is large. We consider the tandem of $N = 9$ caches in Figure 4.7.

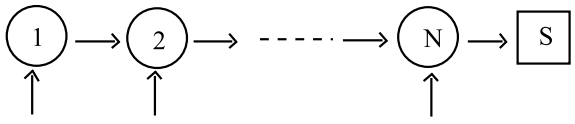


Figure 4.7: Linear network with exogenous request arrivals

At cache n, exogenous requests arrive according to a Poisson process with rate λ_n and TTL is exponentially distributed with mean μ_n^{-1}. We apply Algorithm 5 described in Section 4.2.5 and compare its prediction to the exact metrics obtained through the analysis of the Markov process $\{\mathbf{x}(t), t \geq 0\}$ introduced in the previous paragraph. We calculate the absolute relative errors at cache n for the hit probability ($E_{HP,n} := |H^{\mathcal{M}}_{P,n} - H_{P,n}|/H^{\mathcal{M}}_{P,n}$, where $H^{\mathcal{M}}_{P,n}$ is the hit probability

obtained from the Markov process analysis), the miss rate ($E_{MR,n}$), and the occupancy probability ($E_{OP,n}$). One thousand different samples for the exogenous request arrival rates and the TTL ones $\{(\lambda_n, \mu_n), n = 1, \ldots, 9\}$ have been selected from the intervals $[0.001, 10]$ and $[0.1, 2]$ respectively. We use the Fourier Amplitude Sensitivity Test (FAST) method [68] to explore the space $[0.001, 10] \times [0.1, 2]$. Figure 4.8 shows the CCDFs of the relative errors for cache 9. We observe that Approximation 4.1 is very accurate; in 90% of the different parameter settings the relative errors on all metrics of interest are smaller than 10^{-4}.

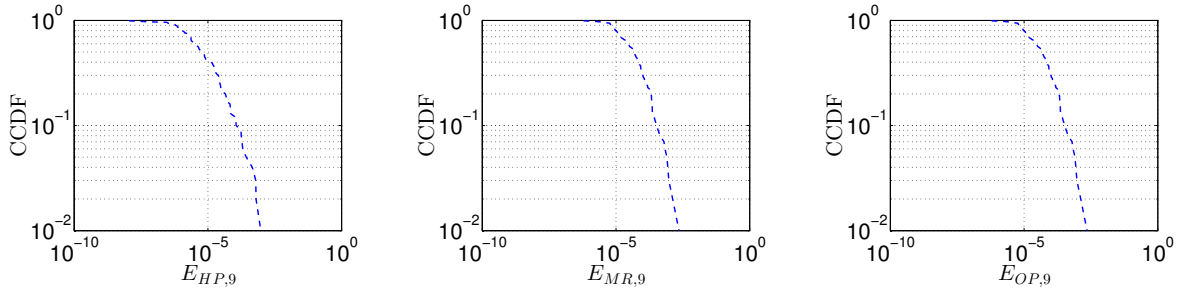


Figure 4.8: CCDFs of $E_{HP,9}$, $E_{MR,9}$, $E_{OP,9}$ for network in Fig. 4.7

We then considered a homogeneous scenario where all caches have identical TTL and exogenous arrival rates, i.e. $\mu_n = \mu$ and $\lambda_n = \lambda$, $\forall n$. The relative errors are shown in Figure 4.9 as a function of the normalized load $\rho = \lambda/\mu$ for $\mu = 0.2$. We observe that the largest error (about 2×10^{-4}) is obtained when arrival and timer rates are comparable (i.e. $\rho \approx 1$).

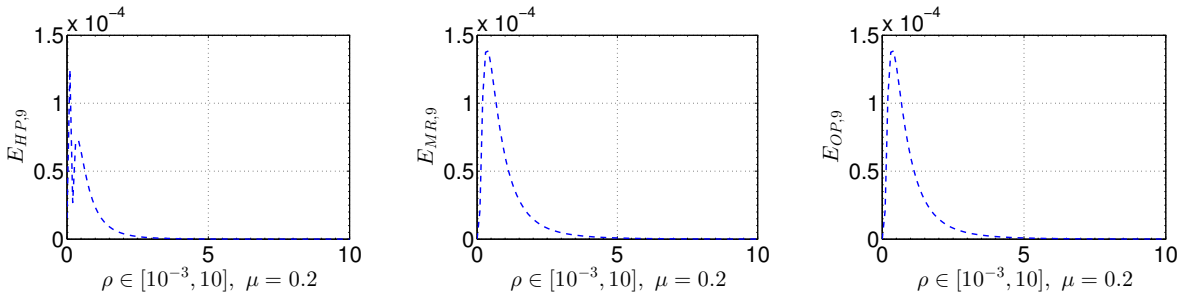


Figure 4.9: $E_{HP,9}$, $E_{MR,9}$, $E_{OP,9}$ for homogeneous network in Fig. 4.7 ($\lambda_n = \lambda = \rho\mu = \rho\mu_n$)

Twelve caches caterpillar tree This network consists of three trees (star networks), each with 4 caches, whose roots are connected as in Figure 4.10.

We choose this network architecture for its large number of leaves and its relative small

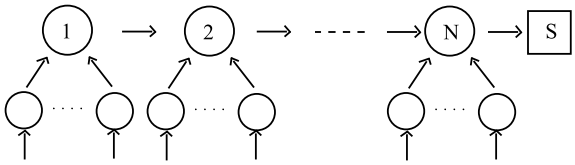


Figure 4.10: Caterpillar network

depth in comparison to the previous linear network. We consider the leaves of each root are identical i.e. they have the same average TTL value and they are fed with Poisson request processes with an identical rate. Again, Algorithm 5 produces exact results for all leaves. As previously, exact results are obtained by considering the Markov process $\{\mathbf{x}(t), t \geq 0\}$ associated to this network. Different request and TTL rates have been selected according to FAST method respectively in the intervals $[0.001, 10]$ and $[0.1, 2]$. We used 4921 samples for each rate. The empirical CCDFs of the relative errors $E_{HP,3}$, $E_{MR,3}$, and $E_{OP,3}$ at the higher level cache are shown in Figure 4.11.

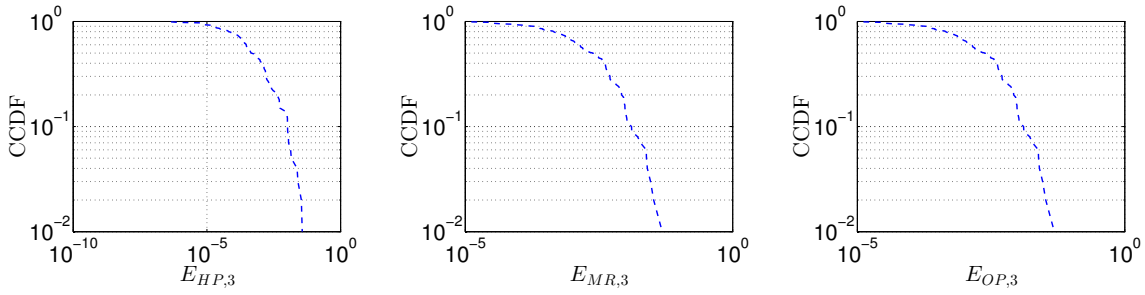


Figure 4.11: CCDFs of $E_{HP,3}$, $E_{MR,3}$, $E_{OP,3}$ for network in Fig. 4.10

The results obtained are analogous to those of the linear cache network in previous paragraph. The relative errors can be larger in this scenario, but they are probably negligible for most of the applications (10^{-2} in 90% of the cases). In this case too, we have also considered the homogeneous scenario where the TTLs at the leaves have the same expected value as the ones at the internal nodes. We observed that the relative errors have the same order of magnitude i.e. less than 10^{-2}.

Nine caches tree network We consider the tree network of nine caches illustrated in Figure 4.12 that combines the properties of the previous network samples (i.e. with both a relative large depth and number of leaves).

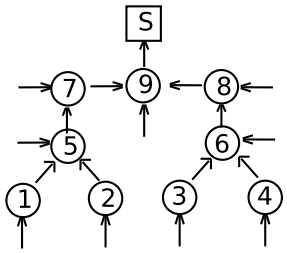


Figure 4.12: Balanced tree network

Also in this case, we consider caches are fed by exogenous requests described by Poisson processes and TTLs are exponentially distributed. The request and TTL rates are selected (6649 different samples in total) from the intervals $[0.05, 10]$ and $[0.1, 2]$ respectively using FAST method. Figure 4.13 shows the CCDFs of the relative errors at the higher level cache and in 90% of cases they are smaller than 10^{-2}. Thus, Approximation 4.1 is still accurate.

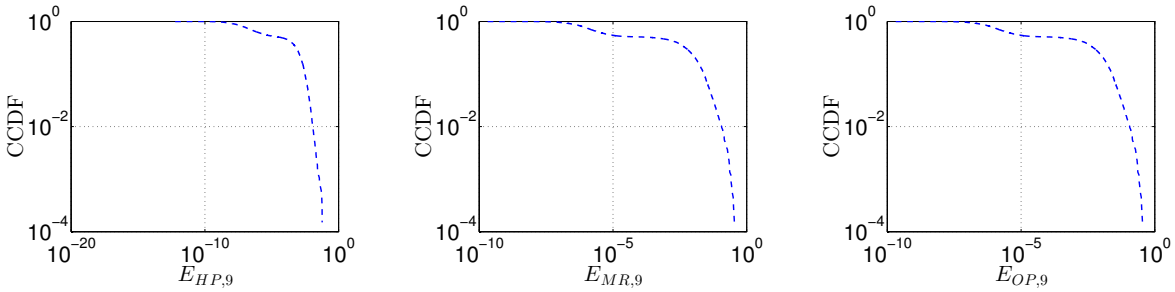


Figure 4.13: CCDFs of $E_{HP,9}$, $E_{MR,9}$, $E_{OP,9}$ for network in Fig.4.12

Poisson traffic and deterministic timers

When timers are deterministic, we resort to the general procedure in Algorithm 4 presented in Section 4.2.5. As term of comparison we consider simulation results, given that the network is no longer 'Markovian'.

Figure 4.14 shows the settings (topology, request rates and TTL values) of the network. Algorithm 4 introduces two sources of errors. First, the aggregated request process at a cache is not a renewal process; however, we use Approximation 4.1 and apply the renewal equation (4.20). Second, (4.19) and (4.20) introduce some numerical errors since we need to compute the integrals therein on a finite support. Two parameters determine the size of the numerical error: 1) the time interval (τ) from which the CDF samples are taken, and 2) the time interval between two consecutive samples (Δ). Clearly the larger τ and the smaller Δ are, the smaller is the numerical error and the larger is the computational cost.

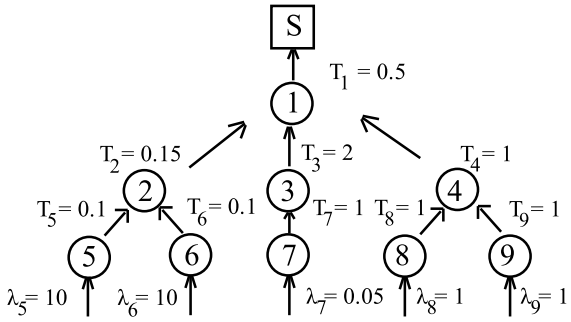


Figure 4.14: Tree network

We implemented a MATLAB numerical solver that iteratively determines the CDFs of inter-arrival times at each cache together with the metrics of interest. The integrals appearing in (4.19) and (4.20) are approximated by simple sums and for simplicity the same values τ and Δ have been considered for all the CDFs numerical integrations. These parameters are selected as follows: we set the parameter τ to five times the largest expected inter-arrival time in the network; while the parameter Δ is set to one thousandth of the minimum of the TTL values and the expected inter-arrival times of the exogenous request processes.
The absolute relative error of the hit probability is evaluated as $|H_{P,n} - H^{\mathcal{S}}_{P,n}|/H^{\mathcal{S}}_{P,n}$ where $H_{P,n}$ is our estimate and $H^{\mathcal{S}}_{P,n}$ is obtained through simulation. The duration of the simulation is set so that there is a small incertitude on the performance metrics: the 99% confidence interval $[H^{\mathcal{S}}_{P,n} - \epsilon, H^{\mathcal{S}}_{P,n} + \epsilon]$ is such that the ratio $(2\epsilon/H^{\mathcal{S}}_{P,n})$ is at most 0.6×10^{-4}. For all the performance metrics at all caches, the relative error of our approach is less than 10^{-2}.

Renewal/non-Poisson traffic

In this scenario, we consider that requests for each data item are generated according to Interrupted Poisson Processes (IPP). IPPs are Renewal processes whose inter-arrival times have a two stage hyper-exponential distribution [39] (then it is a particular diag.ME distribution).

We evaluate the accuracy of our approach on binary tree networks (like the one in Figure 4.15) where leaves are fed by request traffic described before and TTLs values are deterministic or drawn from the following diag.ME TTL distributions: exponential, hypo-exponential and hyper-exponential distributions. Also in this case we consider simulation results as term of comparison. Our model predictions are provided by Algorithms 4 and 5, respectively for deterministic and (hypo-, hyper-) exponentially distributed TTLs.

Small binary tree. We consider the seven caches binary tree in Figure 4.15. Relative errors at the higher level cache are displayed in Figure 4.16. For all performance metrics at all caches

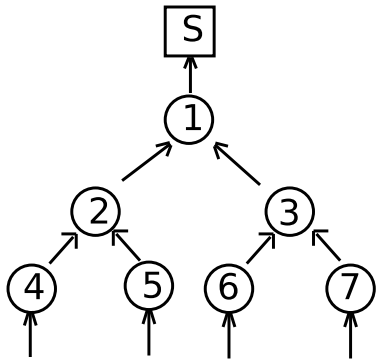


Figure 4.15: Binary tree network

of this tree, the relative errors of our approach are less than 2×10^{-3}. This result validates Assumption 4.1 and thus our model in the context of general networks i.e. with non-Poisson arrivals and different TTL distributions.

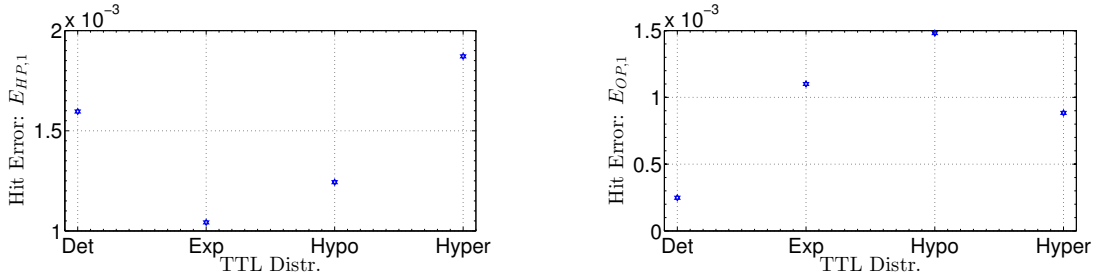


Figure 4.16: Relative error $E_{HP,1}$ and $E_{OP,1}$ under IPP traffic.

As theoretically proved in Proposition 4.3, Figure 4.17 confirms that the deterministic TTL is the optimal TTL configuration at the leaves (caches $4-7$) i.e. which maximizes the hit and occupancy probabilities. This observation is not surprising since IPPs are renewal processes with hyper-exponentially distributed inter-arrival times; in fact, it can be easily checked that the hyper-exponential CDF is concave and the observed results follows from Proposition 4.3.

Large binary tree. We also investigate the quality of our approximation on larger tree networks (up to 40 caches) where TTLs are constants drawn uniformly at random in the interval $[0.5; 1.5]$, and the exogenous requests at each cache are described by an IPP. The expected value and the squared coefficient of variation of inter-arrival times are uniformly chosen at random in $[0.05; 2]$ and $[1.5; 2]$ respectively. As shown in Table 4.2, the relative errors between the event-driven simulations and our analytic approach are of order of 1%. This result provides good insights on the robustness and accuracy of our approach when dealing with large networks.

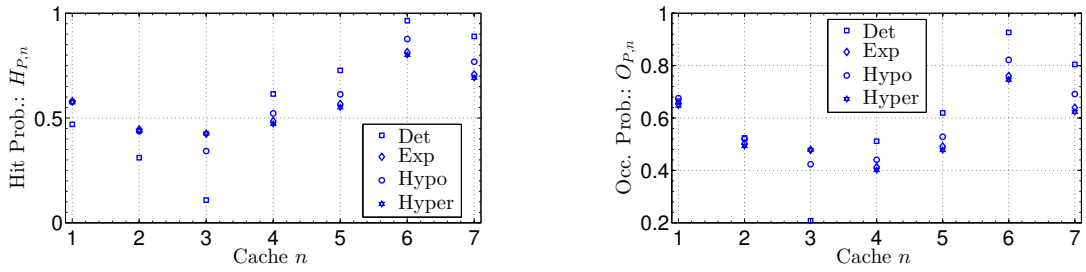


Figure 4.17: Optimality of the Deterministic TTL at leaves fed by IPP arrivals

Type (Degree, Depth, # Caches)	Level l, Cache n	$E_{HP,n}(\%)$	$E_{MR,n}(\%)$	$E_{OP,n}(\%)$
Binary Tree	1, 1	1.059	0.929	0.021
(2, 5, 31)	2, 3	0.406	0.042	0.117
	5, 31	0.075	0.018	0.061
Ternary Tree	1, 1	0.127	0.085	0.134
(3, 4, 40)	2, 3	0.061	0.278	0.124
	4, 40	0.006	0.283	0.759

Table 4.2: Relative Errors on Performance metrics for large trees

We have shown that Approximation 4.1 leads to very accurate results when exogenous requests are described by renewal process (Poisson and Interrupted Poisson processes) and TTLs have some matrix-exponential distributions or deterministic ones. This lets us think that the superposition of the request arrival processes at every cache is very 'close' to a renewal process at least for all the cases we tested.

4.2.7 Computational cost and time

In this section we perform a preliminary analysis of the computational cost and time of our approach, and we compare it to other solutions presented in the previous section such as solving a Markov chain (Section 4.2.6) and event-driven simulations (Sections 4.2.6 and 4.2.6).

TTLs with diag.ME distribution

We first address the case of a hierarchical tree of diag.ME TTL caches introduced in Section 15. We consider a tree of N nodes and M internal nodes (i.e. $N - M$ leaves). Since the computational cost for all the metrics is roughly the same, we focus here on the hit probability. In order to calculate the hit probability at one of the nodes labeled $n \in \{1, \ldots, N\}$, say at cache n, we need to:

- calculate the CDF $H_n(t)$ of inter-arrival times of the aggregated request process in (4.37). This requires a number of operations proportional to

$$(1+C_n)\prod_{i\in\mathcal{C}(n)\cup\{0\}} K_{i,n} = O\left((1+C_n)\times \tilde{K}_n^{1+C_n}\right),\ \tilde{K}_n = \max_{i\in\mathcal{C}(n)\cup\{0\}} K_{i,n}$$

  where $K_{i,n}$ is the minimal order of the i-th child miss process, $K_{0,n}$ is the minimal order of the exogenous request process, and $C_n = |\mathcal{C}(n)|$ is the number of children of cache n.

- evaluate the LST $H_n^*(\mu_{n,j})$ in the expression of the hit probability in (4.30) which requires $K_n \times J_n$ operations where K_n is the minimal order of the aggregated request process (and it is at most equal to $\tilde{K}_n^{1+C_n}$) and J_n is the minimal order of the TTL distribution.

Then, the total cost is

$$\mathcal{K}_{\texttt{diag.ME}} = O\left(\sum_{n=1}^{N}(1+C_n+J_n)\times \tilde{K}_n^{1+C_n}\right). \tag{4.38}$$

For linear networks in Figure 4.3 (case of small maximum degree), the number of children per cache is $C_n = 1$ and there are no exogenous requests at cache $n > 1$. Hence, the total cost is

$$\mathcal{K}_{\texttt{linear}} = O\left(NJ\times (K_{0,1}(J+1))^N\right),\ J = \max_{n=1,\dots,N} J_n \tag{4.39}$$

For star networks in Figure 4.4 (case of large maximum degree), the number of children at the root is $N-1$ and the total cost is

$$\mathcal{K}_{\texttt{star}} = O\left(NJK + J\,(K(J+1))^{N-1}\right),\ J = \max_{n=1,\dots,N} J_n,\ K = \max_{n=1,\dots,N} K_{0,n} \tag{4.40}$$

TTLs with exponential distribution

The exponential distribution has the minimal order which is one. Hence, if we consider exponential timers and exogenous requests are described by Poisson processes, we have $K_{0,n} = J_n = 1$ at each cache n. Therefore the costs $\mathcal{K}_{\texttt{linear}}$ and $\mathcal{K}_{\texttt{star}}$ are respectively equal to $O(N\times 2^N)$ and $O(N+2^N)$.

We showed in Section 4.2.6 that alternative approaches like the Markov chain analysis can provide exact results when the tree is fed by Poisson traffic and the TTLs are exponentially distributed. The size of the state space of the Markov process $\{\mathbf{x}(t), t\geq 0\}$ is 2^N where N is the number of nodes. The cost of determining the steady-state distribution by solving the linear equation system is $O(2^{3N})$. This is much larger than the cost of our method $O(N2^N)$.
A different approach is to obtain an approximate steady-state distribution of the Markov process using an iterative method. This approach takes advantage of the fact that most of the transition rates are zero. In fact, a state change is triggered by an exogenous request arrival at a cache

that does not have the data item or by a timer expiration at a cache with the data item, i.e. from a given state we can only reach other N states. Then the number of non-zero rates is $N \times 2^N$ and each iteration of the method requires $O(N \times 2^N)$ operations. The total cost of the iterative method is then $O(I \times N \times 2^N)$, where I is the number of iterations until termination. The quantity I depends on the spectral gap of the matrix used at each iteration, and also on the required precision. In general, we can expect that $O(I \times N \times 2^N) \ll O(2^{3N})$. Having this inequality, we can say that our method, even in the worst case, is still more convenient than solving the Markov process on linear/star networks, because $O(N2^N) < O(I \times N \times 2^N)$.

TTLs with deterministic distribution

Let us now consider the case of a general tree network with constant TTLs (equal to T). In this case there is no exact solution to compare our approach with, so we consider simulations as an alternative approach. We perform an asymptotic analysis. A meaningful comparison of the computational costs needs to take also into account the incertitude of the solution: both the simulations and our method can produce a better result if one is willing to afford a higher number of operations. In order to combine these two aspects in our analysis, we consider as metric the product precision times number of operations. Intuitively the larger this product the more expensive is to get a given precision.

For the simulations the computational cost is at least proportional to the number of events that are generated, let us denote it by n_E. The incertitude on the final result can be estimated by the amplitude of the confidence interval, that decreases as $1/\sqrt{n_E}$, then the product precision times number of operations is proportional to $\sqrt{n_E}$ for the simulations.

In the case of our approach, the most expensive operation is the solution of the renewal equation (4.5). If we adopt the same τ and Δ for all the integrals, we need to calculate the value of the CDF of the miss rate ($G_n(t)$) in $n_P = \tau/\Delta$ points and then we need to calculate n_P integrals. The integration interval is at most equal to the TTL duration T thanks to (4.48), then each integral requires a number of operations proportional to $n'_P = T/\Delta$. If the value of τ is selected proportionally to T, then the cost of our method is proportional to n_P^2. A naive implementation of the integral as a sum of the function values leads to an error proportional to the amplitude of the time step and inversely proportional to n'_P or n_P. In conclusion the product precision times the number of operations is proportional to n_P.

Then, for a given precision, our method would require a number of points much larger than the number of events to be considered in the corresponding simulation (at least asymptotically). The comparison would then lead to prefer the simulations at least when small incertitude is required (then large n_E and n_P). In reality integrals can be calculated in more sophisticated ways, for example if we adopt Romberg's method, with a slightly larger computation cost, we can get a precision proportional to n_P^{-2}. In this case the product precision times number of

operations is a constant for our method, that should be preferred.

Numerical experiments

Case of linear networks. We performed some experiments to validate our conclusion based on an asymptotic analysis. First, we consider linear networks of $N \in \{1, 2, \dots 9\}$ exponentially distributed TTL-based caches. We compare the running time of solving the corresponding Markov chain (see Section 4.2.6) against our Algorithm 5. Figure 4.18 shows the ratio of the computation times $T^{\mathcal{A}}$ and $T^{\mathcal{M}}$ respectively for our Algorithm 5 and for the Markov chain resolution. Both the solutions have been implemented in MATLAB, in particular the naive function *linsolve* has been used to determine the steady-state distribution of the Markov chain and the Algorithm 5 has been implemented with basic routines. Our algorithm performs faster than the Markov chain resolution specially when the depth of the linear network is large.

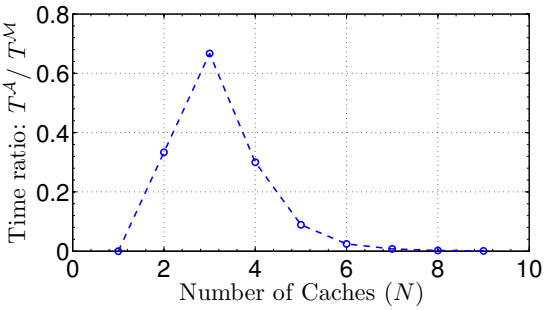


Figure 4.18: Computation time comparison on linear networks

Case of tree networks. Second, we evaluate the computational time of the event-driven simulation and our Algorithm 4 on the k-ary trees of Section 4.2.6 where the TTLs are constants and the request processes are IPPs. $T^{\mathcal{S}}$ and $T^{\mathcal{A}}$ are respectively the time to compute all performance metrics on these large tree networks via event-driven simulations and our analytic methodology in Algorithm 4; they are computed by using the MATLAB routines *tic* and *toc*. Table 4.3 shows that as the number of caches N increases, our analytic solution is clearly preferable since it is the least time consuming.

Type	Degree	Depth	# Caches, N	$T^{\mathcal{S}}$	$T^{\mathcal{A}}$
Binary Tree	2	5	31	53	88
Ternary Tree	3	4	40	197	129

Table 4.3: Comparison of computation time on large trees

4.2.8 TTL-based caches: implementation and other policies

We recall that TTL-based models we presented until now assume infinite cache capacity. In this section, we address issues and practical concerns related to finite capacity constraints.

Pra-TTL cache: a practical implementation of a TTL-based cache

While the TTL-based model allows an arbitrarily large number of contents in its memory, a real cache will have a finite capacity B. In this paragraph, we consider a possible practical implementation of our TTL-based model that we call *Pra-TTL*. The Pra-TTL cache uses a timer for each content item in the same way as the TTL-based model, but does not discard a content item whose timer has expired as long as some space is available in the memory. If a new content item needs to be stored and the cache is full, the content item to be erased is the one whose timer expired furthest in the past (if any) or the one whose timer will expire soonest.

We have compared the performance of the Pra-TTL cache with that of our TTL-based model on a linear network of $N = 5$ caches labeled $n = 1, \ldots, 5$ having the same capacities $B_n = 20$. The requests for each file $f = 1, \ldots, F = 200$ arrive only at the first cache at rate $\lambda_1 = 2.0$ i.e. there is no exogenous arrival at caches 2–5. We consider that requests over the set of files follow a Zipf distribution with parameter $\alpha = 1.2$: i.e. requests for file f are described by a Poisson process with rate $\lambda_{1,f} = \lambda_1 \times \left(f^{-\alpha} / \sum_g g^{-\alpha}\right)$. TTLs of file f at cache n are exponentially distributed with rate $\mu_{n,f} = \mu_n$ such that the total occupancy for the TTL-based model equals the corresponding cache capacity B_n in average. In other words, μ_n is chosen such that $\sum_{f=1}^{F} O_{P,n,f} = B_n$ where $O_{P,n,f}$ is the occupancy probability of file f at cache n calculted in Proposition 4.1 (i.e. predicted by the model of an infinite TTL-based cache).

The hit probability per file f at each cache n is denoted $H_{P,n,f}$ and the aggregate hit probability at cache n is denoted $H_{P,n,*}$. We compute theses performance metrics for both Pra-TTL and TTL-based caches by using the following expression for $H_{P,n,*}$:

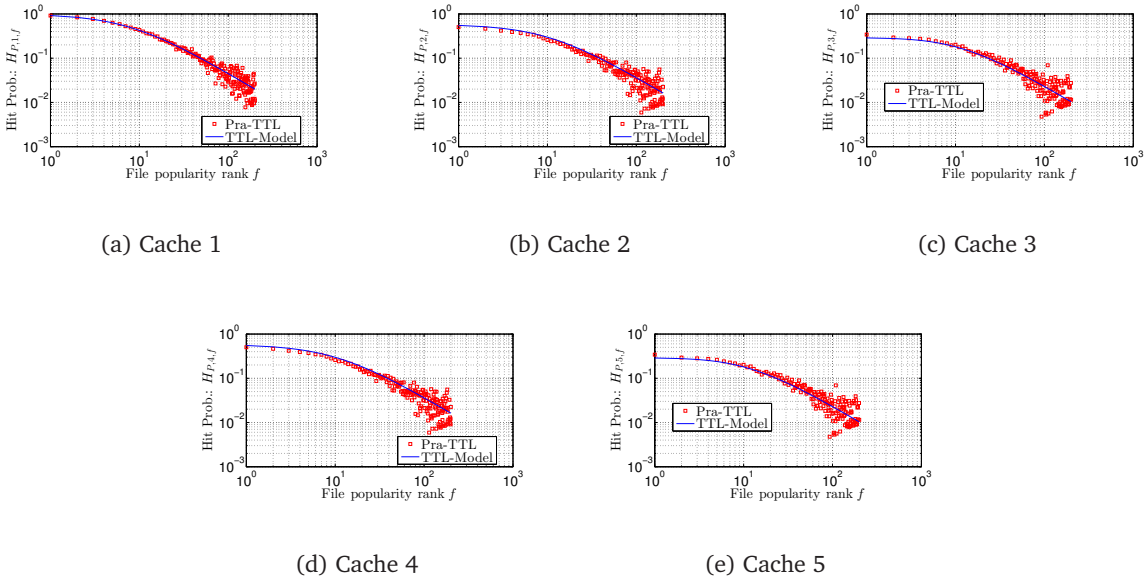
$$H_{P,n,*} = \left(\sum_f \Lambda_{n,f} H_{P,n,f} \right) / \Lambda_{n,*}$$

where $\Lambda_{n,f}$ is the total request rate of file f at cache n and $\Lambda_{n,*} = \sum_f \Lambda_{n,f}$. Then, $\Lambda_{n,f}$ is simply the miss rate of file f at cache $n-1$ since the network is linear and there is no exogenous request arrivals at cache n ($\forall\, n > 1$).

Table 4.4 and Figure 4.19 show that our model (that assume infinite cache size) well predict the performance metrics for Pra-TTL, both those of the aggregate at a cache and those of a specific file respectively. These preliminary results suggest that our analysis can be useful to study TTL-based policies under capacity constraints.

Table 4.4: Aggregated Hit probability at cache n, $H_{P,n,*}$

Caches n	1	2	3	4	5
Pra-TTL	0.5590	0.4216	0.3030	0.1941	0.1380
TTL-Model	0.5585	0.4658	0.2672	0.1670	0.1154



(a) Cache 1 (b) Cache 2 (c) Cache 3

(d) Cache 4 (e) Cache 5

Figure 4.19: Pra-TTL against TTL-Model, hit probability $H_{P,n,f}$ of file f at each cache n.

Relationship with other replacement policies

In this paragraph, we establish a link between our TTL-based model and other replacement policies at a single cache. We consider a single cache with capacity B serving F files, where requests are described by independent Poisson processes with rates λ_f for $f = 1, 2, \ldots, F$. We tune the expiration rate μ_f for each file f in order to obtain the same performance metrics of common replacement policies like LRU, FIFO or RND.

We detail the procedure for a single RND cache, but it can be extended to the other policies. Let us denote by π_f the stationary probability that file f is in the RND cache. This distribution has been calculated in [17, 62]. For the exponentially distributed TTL cache, the stationary occupancy probability of the f-th file is given by

$$O_{P,f} = \lambda_f \frac{1 - F_f^*(\mu_f)}{\mu_f},$$

where $F_f^*(s) = \frac{\lambda_f}{\lambda_f + s}$ is the LST of inter-arrival times. If we select $\mu_f = \lambda_f \left(\frac{1}{\pi_f} - 1\right)$, we have $O_{P,f} = \pi_f, \ \forall f$, i.e. the two policies have the same stationary cache occupancy for each file. If we select the same TTL rate μ for all the files it is possible to achieve the same average occupancy at the cache, i.e. $\sum_f \pi_f = \sum_f \frac{\lambda_f}{\lambda_f + \mu} = B$. For each file, the miss process obtained with the exponential TTL-based cache is an accurate description of its miss stream on the RND cache [72]. From the equality of the stationary cache occupancy probabilities, the equality of hit/miss probabilities and rates follows due to the PASTA property since requests are described by Poisson processes.

In this sense, the TTL policy is more general than RND or LRU since it can mimic their behavior and reproduce their performance metrics. While, the exponential TTL cache enables easy calculation we can select other distributions like the deterministic one (see Paragraph *Approximation for LRU caches* in Section 4.2.3) in order to better match the CDF of the inter-miss times of a LRU cache as well.

Remark 4.3 (Existing implementations of TTL-based models) *We showed in Chapter 2 that LRU, RND, and FIFO caches asymptotically behave as TTL-based caches. Thereby the former caches may be seen as possible practical and simple implementations of TTL-based caches.*

4.2.9 Perspectives on CCN work

In this section, we introduced a Time-To-Live (TTL) based policy as replacement algorithm for cache networks in general, and for the content-routers of CCNs in particular. We developed a set of building blocks for the performance evaluation of theses TTL-based cache networks through renewal arguments. We characterized a class of networks for which we provided the exact performance metrics: this class contains linear and star tree networks. We also provided a recursive and approximate procedure to study arbitrary hierarchical networks. We showed that our theoretic model predicts remarkably well the performance metrics with relative errors less than 1%. We formally proved that deterministic TTLs are optimal when the inter-arrival times have a concave CDF. Our approach is promising since it appears as a unifying framework to accurately analyze a richer class of networks also with heterogeneous policies deployed at different caches. We have also demonstrated that our TTL-based model can be used in practice under capacity constraints thanks to the new implementation Pra-TTL caches or existing ones as LRU and RND caches. A reader interested in how TTL-based models perform under finite capacity constraints may report to the next chapter where they are used to analyze general and heterogeneous networks where caches are running LRU and RND replacement policies. We also consider the case of correlated requests modeled by Markov-Arrival processes.

4.3 Modern DNS hierarchy

Caching is undoubtedly one of the most popular solution that easily scales up with a world-wide deployment of resources. Records in Domain Name System (DNS) caches are kept for a pre-set duration (time-to-live or TTL) to avoid becoming outdated. *Modern* DNS caches are those that set locally the TTL regardless of what authoritative servers say. In this section, we apply TTL-based cache models to study the modern DNS cache behavior relying on renewal arguments. For tree cache networks, we derive cache performance metrics, characterize at each cache the aggregate request and miss processes. We address the problem of the optimal caching duration and find that constant TTL is the best only if if inter-request times have a concave CDF. We validate our findings using real DNS traces (single cache case) and via event-driven simulations (network case). Our models are very robust as the relative error between empirical and analytic values stays within 1% in the former case and less than 5% in the latter case.

4.3.1 Introduction

In-network caching is a widely adopted technique to provide an efficient access to data or resources on a world-wide deployed system while ensuring scalability and availability. For instance, caches are integral components of the Domain Name System, the World Wide Web, Content Distribution Networks, or the recently proposed Information-Centric Network (ICN) architectures. Many of these systems are hierarchical. The content being cached is managed through the use of expiration-based policies using a time-to-live (TTL) or replacement algorithms such the Least Recently Used, First-In First-Out, Random, etc.

In this section, we focus on hierarchical systems that rely on expiration-based policies to manage their caches. These policies have the advantage of being fully configurable and provide parameters (i.e. timers) to optimize/control the network of caches. Each cache in the system maintains for each item a timer that indicates its duration of validity. This timer can be initially set by an external actor or by the cache itself.

The Domain Name System (DNS) is a valid application case. In short, the DNS maintains in a distributed database the mappings, called *resource records*, between names and addresses in the Internet. Servers in charge of managing a mapping are said to be *authoritative*. Caches—used to avoid generating traffic up in the DNS hierarchy—are found in both servers and clients (devices of end users). Caching is however limited in duration to avoid having stale records which may break the domains involved.

DNS cache updates are strongly related with how the DNS hierarchy works. When a requested resource record RR is not found at the client's cache, the client issues a request to a bottom level DNS server (usually that of the Internet server provider). If the RR cannot be

resolved locally and is not found in the cache, the latter server forwards the request to a server higher in the hierarchy. The process repeats itself until the RR is fetched at a cache or ultimately from the disk of an authoritative server. The server providing the RR is called the *answerer*. The RR is sent back to the client through the reverse path between the answerer and the client, and a copy of the RR is left at each cache on this path.

According to RFC 6195, all copies of the RR are marked by the answerer with a time-to-live (TTL) which indicates to caches the number of seconds that their copy of the RR may be cached. Consequently, all copies of a record along a path would be cached for the same duration. This RFC specification is called the *TTL rule* in the literature. Caches compliant with it are referred to as *traditional DNS caches*. Those overriding the advocated TTL with a locally chosen value (cf. [80, 13]) are called *modern DNS caches* [19].

In a tree of traditional DNS caches a request occurring anywhere just after the content expired in the local cache yields cache misses at all caches along the path to an authoritative server. Such a *miss synchronization effect* [51] is avoided with modern caches. Other cases of deviation from the TTL rule are reported in [74, Sect. 2.1]

Our objective is to assess the performance of these networks. Our contributions are:

(i) we are the first to provide analytic models to study both a single (modern) DNS cache and a tree of caches with general caching durations;

(ii) we characterize the distribution of the DNS traffic flowing upstream in the DNS hierarchy besides deriving the usual cache performance metrics;

(iii) for the case of a single cache we identify when deterministic caching duration is optimal and discuss how to choose the optimal deterministic value when this is the case;

(iv) for the case of a network of caches with diagonal matrix-exponential distributions, we compute the distribution of the inter-request and inter-miss times at each node;

(v) we check the robustness of our single cache model over DNS traces collected at Inria and

(vi) the robustness of our network of caches model through event-driven simulations.

The rest of the section is organized as follows. Section 4.3.2 reviews the works most relevant to this paper. Section 4.3.3 presents the scenario considered and some introductory material. Our single cache model is analyzed in Sect. 4.3.4 and the case of a tree of caches in Sect. 4.3.5. We validate our models in Sect. 4.3.6 and show some numerical results. Section 4.4 summarizes our findings.

4.3.2 Related work

Since the recent observation of the modern behavior of DNS caches [19, 80], only few results of the state of the art are applicable to modern DNS caches. Hou *et al.* consider in [50] a tree of traditional DNS caches fed by Poisson traffic. The performance metrics derived in [50] cannot characterize modern caches as these do not cause a miss synchronization effect—like traditional caches do—which is extensively used in their model.

Jung, Berger and Balakrishnan study in [59] a single traditional DNS cache fed by a renewal process. Their model assumes that each content is cached for a deterministic duration which would be either the value marked by an authoritative server or the maximum among all values received from intermediate caches. The hit/miss probabilities derived are approximate in traditional DNS caches receiving different TTLs from higher-level caches and exact in traditional DNS caches receiving always their responses from authoritative servers. It is interesting to note that the model of [59] is valid for a single modern DNS cache that overrides the given TTL with a fixed caching duration. Characterizing the traffic not served by the cache (the miss process), considering distributions of caching durations other than the deterministic one, and most challenging extending to the case of a network of caches are issues yet to be addressed.

The approach introduced in Section 4.2 is methodologically close to the analysis we shall derive here for modern DNS caches. The essential difference is that caching durations of TTL-based models in Section 4.2 are regenerated from the same distribution at each cache hit. As such, the model of Section 4.2 applies to modern DNS caches only if caching durations are exponentially distributed, thanks to the memoryless property of the exponential distribution. Observe that the context targeted in Section 4.2 is that of ICN achitectures.

It has been reported in [19, 60, 80]—and we have observed it in our collected DNS traces—that the sequence of TTLs received corresponding to a given resource record exhibits some randomness. We believe it is crucial to consider this randomness when modeling a modern DNS cache. Another key issue concerns the optimal distribution for the caching durations. Callahan, Allman and Rabinovich mention in [19] that no model or experiment characterizes the optimal (deterministic) TTL choice. We will address a more general problem in this paper, namely, finding the best distribution.

4.3.3 Definitions and assumptions

Considered scenario

In this section, caches are assumed to consist of infinite size buffers. This assumption derives naturally from the fact that the cached entities—the DNS records—have a negligible size when compared to the storage capacity available at a DNS server [59]. A nice consequence is that the management of different records can safely be decoupled, simplifying thereby the modeling of

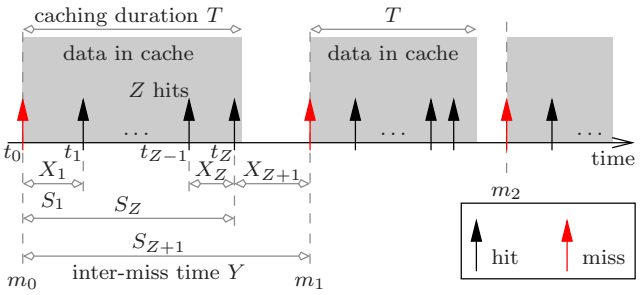


Figure 4.20: TTL-based model of modern DNS cache.

caches. Our analysis will focus on a *single* content/record, characterizing the processes relevant to it, keeping in mind that the *same* can be *repeated* for every single content requested by users.

Without loss of generality, consider that a *cache miss* occurred at time $m_0 = t_0 = 0$. In other words, the content was not in the cache at time when a request arrived. We will neglect the RR processing time at each server/client and the RR travel time between servers, as these times are typically insignificant in comparison with the request inter-arrival time. Consequently the content requested is cached and made available to the requester also at time t_0. More precisely, upstream requests and downstream responses are instantaneous.

A cache miss instantaneously makes the content available in the respective cache for a duration T as shown in Figure 4.20. Each cache samples this duration from its respective distribution. Caches along the path between the server/client receiving the original request and the server where the content was found all initiate a new duration T at the same time, but the durations initiated being different they will expire at different instants. Consequently, caches become asynchronous, something that would not occur should the caches follow the so-called TTL rule.

Any request arriving during T will find the content in the cache. This is a *cache hit*. The first request arriving after T has expired is a *cache miss*. It initiates a new duration during which the content will be cached.

Metrics and properties of a modern DNS cache

The performance of a cache policy can be assessed through the computation of several metrics. The *hit probability* $H_{P,n}$ is the probability that a request can be served by cache n. The *miss probability* $M_{P,n}$ is simply the complementary probability. The *hit/miss rate* ($H_{R,n}/M_{R,n}$) represents the rate at which cache hits/misses occur. The *occupancy* $O_{P,n}$ is the percentage of time during which the content is cached. We say "a cache policy is *efficient*" if its miss probability is low. This is relevant as long as cached contents are up-to-date.

In fact, by setting timers (or violating the TTL rule in the case of modern DNS), a server/client

takes a risk by caching a content for a longer period than it should, as the content may well have changed by the time the locally chosen duration T expires. The cache would then be providing an outdated content. Observe that a content item in the cache is updated only upon a cache miss. But it is only when the update originates from the authoritative server that one can absolutely be certain that the given update is correct. Therefore, a relevant performance metric is the *correctness* probability $c_{P,n}$ of cache n. Another property of a cache is its *freshness* $f_{R,n}$. It defines how fast a change in a record can propagate until this cache. High freshness is desirable with dynamic authoritative servers.

Processes at hand

To fully analyze a cache n taken in isolation one needs to consider:

• *The arrival process:* it may result from the superposition of multiple independent requests arrival processes. Let $X_k = t_k - t_{k-1}$ be the k-th inter-request time ($k > 0$). It is useful to define the kth jump time $S_k = X_1 + X_2 + \ldots + X_k$ with its cumulative distribution function (CDF) $F_n^{(k)}(t) = P(S_k < t)$ and its probability density function (PDF) $f_n^{(k)}(t) = \frac{dF_n^{(k)}(t)}{dt}$. Let $N(t) = \sup\{k : S_k \leq t\} = \sum_{k>0} \mathbf{1}(S_k \leq t)$. The arrival counting process is then $\{N(t), t > 0\}$.

• *The caching duration:* cache n draws the duration T from the same distribution, such that $\mu_n = 1/\mathbb{E}[T]$. The scenario analyzed here considers memoryless caches, i.e. all caching durations set by the same cache are independent and identically distributed. With a slight abuse of notation, let $T_n(t)$ be the CDF of the random variable (rv) T.

• *The outgoing miss process:* cache misses form a stochastic process whose inter-miss time is denoted by $Y_k = m_k - m_{k-1}$ for $k > 0$.

• *The number of hits between consecutive misses:* these hits occur within a single caching duration. Their number is a rv denoted by Z.

In the case of a tree of caches, a subscript referring to the cache label will be added to the rvs for disambiguation. Besides the "instantaneous transmission/processing" assumption that holds throughout this paper, the following holds:

Assumption 4.4 (Renewal arrivals) *Inter-request times* $\{X_k, k \geq 0\}$ *are independent and identically distributed rvs.*

Let X be the generic inter-request time, $F_n(t)$ be its CDF, $f_n(t) = \frac{dF_n(t)}{dt}$ be its PDF, and $\lambda_n = 1/\mathbb{E}[X]$ at cache n.

Assumption 4.5 (Independence) *At any cache, inter-request times and caching durations are independent.*

Assumption 4.6 (Independent arrivals) *Multiple arrivals at any high-level cache are independent.*

Assumption 4.7 (Independent caches) *Caching durations from any two different caches are independent.*

Assumption 4.4 is in agreement with the analysis in [59] and [95]. Feldmann and Whitt show in [95] that heavy-tailed processes can be well approximated by a renewal process with a hyper-exponential inter-arrival distribution. Jung, Berger and Balakrishnan show in [59] that the request process arriving at a DNS server's cache is heavy-tailed. Renewal processes with either Weibull or Pareto inter-event distributions are used in [59] to fit the collected inter-request times.

Assumptions 4.5 an 4.7 hold at modern DNS servers [80, 19] and Web browsers [13] as these use their own caching durations independently of the requests and other servers/browsers.

Assumption 4.6 holds if exogenous arrivals are independent, as long as requests for a given content "see" a *polytree* network (that is a directed graph without any undirected cycles, cf. Figure 4.2).

It is worth noting that the scenario and the set of assumptions considered here fit the case of a *single traditional* DNS server if the distribution of its caching durations fits the values marking the responses. Observe also that the popularity of a content is proportional to its request rate λ_n. Therefore, it should be clear that our models account for a content's popularity (which can be Zipfian, Uniform, Geometric, etc.) through the per-content request rate λ_n.

A word on the notation: for any function $\chi(t)$, its Laplace-Stieltjes Transform (LST) is $\chi^*(s) = \int_0^\infty e^{-st} d\chi(t)$ $(s \geq 0)$. Observe that the LST of a function is the Laplace transform of its derivative. The complementary cumulative distribution function (CCDF) of a CDF $\chi(t)$ is $\tilde{\chi}(t) = 1 - \chi(t)$. Table 4.5 summarizes the main notation used in the paper.

4.3.4 Analysis of a single cache

We are ready now to analyze a cache n taken in isolation. The results found here will be used in Sect. 4.3.5 when studying multiple caches in a tree network.

The model and its analysis

Our first goal is to characterize the miss process which is the same as the process going out from a server towards the higher-level server. The request process and the caching durations are as assumed in Sect. 4.3.3, i.e. $\{N(t), t > 0\}$ is a renewal process. The renewal function and the renewal density function associated to $\{N(t), t > 0\}$ are, respectively, $M_n(t) = E[N(t)] = \sum_{k>0} F_n^{(k)}(t)$ and $m_n(t) = \frac{dM_n(t)}{dt} = \sum_{k>0} f_n^{(k)}(t)$. It is well-known that the renewal function satisfies the so-called renewal equation [28]

$$M_n(t) = F_n(t) + \int_0^t F_n(t-x) dM_n(x). \tag{4.41}$$

Table 4.5: Glossary of main notation of modern DNS cache

$H_{P,n}$	hit probability at cache n	$F_n(t)$	CDF of X at cache n
$H_{R,n}$	hit rate at cache n	$f_n(t)$	PDF of X at cache n
$M_{P,n}$	miss probability at cache n	λ_n	arrival rate ($1/\mathbb{E}[X]$) at cache n
$M_{R,n}$	miss rate at cache n	$N(t)$	requests during t (rv)
$O_{P,n}$	occupancy at cache n	$M_n(t)$	renewal function at cache n
T	caching duration (rv)	$m_n(t)$	renewal density funct. at cache n
$T_n(t)$	CDF of T at cache n	Y	inter-miss time (rv)
$1/\mu_n$	expectation of T at cache n	$G_n(t)$	CDF of Y at cache n
X	inter-request time (rv)	Z	hits during T (rv)
S_k	kth jump time (rv)	$\chi^*(s)$	LST of $\chi(t)$
$L_n(t)$	expected number of hits until t within T at cache n		
$H_n(t)$	CDF of inter-request time at a higher-level cache n		

Since T is a rv and $N(t)$ the counting variable, $N(T)$ is a rv which represents the number of requests during a caching duration T. As all requests arriving during this period are necessarily hits, then following the definition of Sect. 4.3.3 we have that $Z = N(T)$ and its expectation is $E[Z] = E[N(T)] = E\left[E[N(T)|T]\right] = E[M_n(T)]$ ($M_n(.)$ is a deterministic function).

Proposition 4.6 (Miss process) *Under Assumptions 4.4 and 4.5 the miss process of a single cache is a renewal process.*

Proof Without loss of generality, we assume that the first request arrives at time $t_0 = 0$ while the content is not cached. This cache miss triggers a new caching period as shown in Figure 4.20. Consequently, miss instants are regeneration points of the state of the cache, implying that these form a renewal process. ∎

According to Proposition 4.6 inter-miss times $\{Y_k\}_{k>0}$ are independent and identically distributed. Let Y be the generic inter-miss time and $G_n(t)$ be its CDF. Deriving $G_n(t)$ completes the characterization of the miss process.

To this end we consider first the number of hits occurring in a renewal interval Y until time t, and more specifically its expectation $L_n(t)$. We can readily write for $t \geq 0$

$$L_n(t) = \sum_{k>0} P(S_k < t, T > S_k) = \int_0^t \bar{T}_n(x) dM_n(x). \tag{4.42}$$

Observe that $L_n(\infty)$ is nothing but the expected number of hits in a renewal interval and is equal to $E[Z]$.

Proposition 4.7 (Inter-miss times) *The CDF* $G_n(t)$ *of the generic inter-miss time* Y *and its LST are given by*

$$G_n(t) = F_n(t) - \int_0^t (1 - F_n(t-x))dL_n(x) \tag{4.43}$$

$$G_n^*(s) = 1 - (1 - F_n^*(s))(1 + L_n^*(s)). \tag{4.44}$$

Proof Let $m_0 = 0$ be the first miss time. The CDF $G_n(t)$ of the inter-miss time Y can be derived by noticing that $Y = S_{Z+1}$ where Z is the number of hits in a renewal interval ($Z \in \mathbb{N}$). As such, the $(Z+1)$st request occurs after T expires and it will initiate a new renewal interval as shown in Figure 4.20. By considering the possible values of Z, we can write

$$\begin{aligned} G_n(t) &= P(S_{Z+1} < t) = \sum_{k\geq 0} P(S_{Z+1} < t, Z = k) \\ &= \sum_{k\geq 0} P(S_k + X_{k+1} < t, S_k < T < S_k + X_{k+1}). \end{aligned}$$

By conditioning first on S_k and then on X_{k+1}, we get

$$\begin{aligned} G_n(t) &= \sum_{k\geq 0} \int_0^t \int_0^{t-u} (T_n(u+x) - T_n(u)) f_n(x)dx\, f_n^{(k)}(u)du \\ &= \sum_{k\geq 0} \int_0^t \int_0^{v} (T_n(v) - T_n(u)) f_n(v-u) f_n^{(k)}(u)\, du\, dv \end{aligned}$$

The last equality is obtained after letting $v = u + x$ in the inner integral and then exchanging the integrals. Observe now that, under Assumption 4.4, the density $f_n^{(k)}(t)$ of the jump time S_k is the k-fold convolution of $f_n(t)$ (the density of X). Also, the convolution of $f_n^{(k)}$ and f is nothing but $f_n^{(k+1)}$. Note that $S_0 = 0$ and $f_n^{(0)}(t) = \mathbf{1}(t = 0)$. A straightforward calculation yields

$$\begin{aligned} G_n(t) &= \sum_{k>0} \int_0^t (1 - F_n(t-x)) T_n(x) f_n^{(k)}(x)dx \\ &= \int_0^t (1 - F_n(t-x))(1 - \bar{T}_n(x))dM_n(x) \\ &= F_n(t) - \int_0^t (1 - F_n(t-x))\bar{T}_n(x)dM_n(x) \end{aligned} \tag{4.45}$$

where we have used (4.41) to write (4.45). By differentiating (4.42) and using $dL_n(x)$ in (4.45), we find (4.43). It suffices to differentiate (4.43) then apply the Laplace transform to get the LST given in (4.44). The proof is complete. ∎

Proposition 4.7 states that one needs to know the CDFs of the arrival process and the caching duration to derive the CDF of the miss process, or equivalently, the outgoing process. This proposition will be repeatedly used in Sect. 4.3.5 when analyzing networks of caches.

Performance metrics

Our next goal is to derive the performance metrics defined in Sect. 4.3.3 at a single cache. Note that these metrics have been defined with respect to a single content. Similar metrics for the a set of contents can also be defined as long as the contents popularity is known. The following proposition provides the cache performance metrics.

Proposition 4.8 (DNS Cache performance) *Under Assumption 4.4, the stationary hit probability* $H_{P,n}$*, the stationary miss probability* $M_{P,n}$*, the occupancy* $O_{P,n}$*, the stationary hit rate* $H_{R,n}$*, and the stationary miss rate* $M_{R,n}$ *are respectively given by*

$$H_{P,n} = \frac{E[Z]}{1+E[Z]}\,; \qquad M_{P,n} = \frac{1}{1+E[Z]}\,; \qquad O_{P,n} = \frac{\lambda_n/\mu_n}{1+E[Z]}\,;$$
$$H_{R,n} = \frac{\lambda_n E[Z]}{1+E[Z]}\,; \qquad M_{R,n} = \frac{\lambda_n}{1+E[Z]}; \qquad E[Z] = L_n(\infty).$$

Proof In the stationary regime, $E[Z]$ is the expected number of hits within a renewal interval and $E[Z]+1$ is the expected number of requests (including the single miss) in a renewal interval. Their ratio naturally gives the hit probability. We can readily find $M_{P,n} = 1 - H_{P,n}$, $H_{R,n} = \lambda_n H_{P,n}$ and $M_{R,n} = \lambda_n M_{P,n}$ since λ_n is the requests arrival rate. As Y is the inter-miss time, we have $E[Y] = 1/M_{R,n}$. Last, regarding the occupancy or the stationary probability that the content data item is in cache, we know that a content is cached for a duration T in a renewal interval Y. Then by renewal theory the occupancy $O_{P,n}$ is the ratio $E[T]/E[Y] = \mu_n^{-1} M_{R,n}$ which completes the proof. ∎

Proposition 4.8 states that it is enough to compute $E[Z]$ and estimate the request rate λ_n at a cache to derive all its metrics of interest (μ_n is locally known). It is worth noting that the hit probability $H_{P,n}$ and the occupancy $O_{P,n}$ are different in general and in particular under renewal arrival processes. The equality $H_{P,n} = O_{P,n}$ holds only if the arrival process is a Poisson process thanks to the PASTA (Poisson Arrivals See Time Average) property.

Refresh rate and correctness probability: case of a dynamic record A cached content may be refreshed only after the TTL T expires, upon a cache miss. Hence the refresh rate is nothing but the miss rate in the case of a cache directly connected to the authoritative server. In the presence of intermediate caches, the refresh rate of a cache is its miss rate times the product of miss probabilities at all intermediate caches.

The correctness probability of a cache is the probability that a request gets the correct content, whether it was cached or not. When a cache is directly connected to the authoritative server, a cache miss ensures that the delivered content is correct whereas a cache hit may or may not provide a correct content. This will depend on the distribution of the inter-change time, say W, at the authoritative server.

We denote by $c_{P,n}$ the correctness probability and $f_{R,n}$ the refresh rate at each cache $n \in \mathcal{C}(m)$ where is the set of children $\mathcal{C}(m)$ of cache m. $\mathcal{P}(n)$ is the set of caches (including cache n) on the path from cache n to the server. We recall that $H_{P,n}$ is the hit probability at cache n and we denote by $c_{P,0}$ the correctness probability at the server ($c_{P,0} = 1$). Given that cache n is fed by a renewal request process having a CDF $F_n(t)$ for inter-arrival times and a rate λ_n, the following result provides the second order metrics of interest of modern DNS caches when caching duration is drawn from a CDF $T_n(t)$.

Proposition 4.9 (Second order performance metrics of modern DNS cache) *The refresh rate $f_{R,n}$ at cache n is given by*

$$f_{R,n} = \lambda_n \times \prod_{i \in \mathcal{P}(n)} (1 - H_{P,i}). \tag{4.46}$$

Moreover, assuming that W is an exponentially distributed rv with parameter θ (i.e. the server is memoryless), the correctness probability $c_{P,n}$ at each cache $n \in \mathcal{C}(m)$ is given by the recursive formula

$$c_{P,n} = 1 - (1 - H_{P,n})(1 - c_{P,m}) - (H_{P,n} - F_n^*(\theta))(1 - T_n^*(\theta)) \tag{4.47}$$

Proof The refresh rate $f_{R,n}$ of cache n is by definition the product of the miss probabilities of caches on the path between cache n to the server times the request rate on cache i which is translated into (4.46).

The correctness probability $c_{P,n}$ at cache $n \in \mathcal{C}(m)$ is found as following

$$\begin{aligned} c_{P,n} &= P(S_Z > T; \quad \text{response from cache } m \text{ is correct}) + P(S_Z < T;\ S_Z < W) \\ &= M_{P,n} \times c_{P,m} + P(S_Z < \min(W, T)). \end{aligned}$$

Assuming that T, W and X are all independent, then conditioning the event $\{S_Z < \min(W, T)\}$ on whether $\{W < T\}$ or $\{W > T\}$ we note that $\min(W, T) = T$ with probability $P(T < W) = T_n^*(\theta)$ and

$$\begin{aligned} c_{P,n} &= (1 - H_{P,n}) \times c_{P,m} + H_{P,n} T_n^*(\theta) + F_n^*(\theta)(1 - T_n^*(\theta)) \\ &= (1 - H_{P,n}) \times c_{P,m} + H_{P,n} - H_{P,n}(1 - T_n^*(\theta)) + F_n^*(\theta) \times (1 - T_n^*(\theta)) \\ &= (1 - H_{P,n}) \times c_{P,m} + H_{P,n} + (F_n^*(\theta) - H_{P,n}) \times (1 - T_n^*(\theta)) \\ &= (1 - H_{P,n}) \times c_{P,m} + 1 - (1 - H_{P,n}) + (F_n^*(\theta) - H_{P,n}) \times (1 - T_n^*(\theta)) \\ &= 1 + (1 - H_{P,n}) \times (c_{P,m} - 1) + (F_n^*(\theta) - H_{P,n}) \times (1 - T_n^*(\theta)) \\ &= 1 - (1 - H_{P,n}) \times (1 - c_{P,m}) - (H_{P,n} - F_n^*(\theta)) \times (1 - T_n^*(\theta)) \end{aligned}$$

where W is exponentially distributed with mean θ^{-1}, $T_n^*(s)$ and $F_n^*(s)$ are the LST of the CDFs $T_n(t)$ and $F_n(t)$ respectively. ∎

Special TTL distributions

We will consider three particular cases for the distribution of the caching duration and derive the corresponding results.

Deterministic distribution We first look at the case when the caching duration at cache n is deterministic and equal to the constant D_n. This setup (single cache, deterministic TTL) is identical to the one in [59].

Result 4.1 (Deterministic TTL) *The expected number of hits in a renewal interval is* $E[Z] = M_n(D_n)$.

Combining Result 4.1 with Proposition 4.8 yields the performance metrics. These are exactly the ones found in [59, Thm 1]. The CDF $G_n(t)$ of inter-miss times, on the other hand, is a new result. Using $T_n(t) = \mathbf{1}(t > D_n)$, (4.43) becomes

$$G_n(t) = \mathbf{1}(t > D_n)\left(F_n(t) - \int_0^{D_n} (1 - F_n(t-x))dM_n(x)\right). \tag{4.48}$$

Approximation for FIFO caches. [72] showed that FIFO caches can be studied as deterministic TTL based caches where the expected TTL value D is solution of a fixed point equation obtained by summing the occupancy probabilities over all files and equalizing to the size of the FIFO cache. Hence, the TTL-based model of this section can be used to accurately describe the miss stream and compute the approximate performance metrics of FIFO caches.

Exponential distribution If caching durations follow an exponential distribution with rate μ_n, then $T_n(t) = 1 - e^{-\mu_n t}$ and the following holds.

Result 4.2 (Exponential TTL) *The expected number of hits in a renewal interval is* $E[Z] = \frac{F_n^*(\mu_n)}{1-F_n^*(\mu_n)}$, *and* (4.44) *giving the LST of* $G_n(t)$ *becomes*

$$G_n^*(s) = \frac{F_n^*(s) - F_n^*(s+\mu_n)}{1 - F_n^*(s+\mu_n)}. \tag{4.49}$$

The result above is identical to Corollary 4.11. We recall that the system considered in Section 4.2.3 consists of caches using expiration-based policies whose caching durations are reset at every cache hit. The DNS scenario considered in this section pre-sets the caching duration at each cache miss. However, when durations are drawn from an exponential distribution, both systems coincide thanks to the memoryless property of the exponential distribution. Therefore, RND caches can be also studied with the TTL-based model introduced in this section.

Diagonal Matrix-Exponential distribution The third particular case considered here is the one of a family of distributions, the so-called *diagonal matrix exponential* distribution (diag.ME for short). The CDF of an ME distribution can be written as $1 - \alpha \exp(\mathbf{S}t)\mathbf{u}$, where α and $\mathbf{u}$ are dimension-n vectors and $\mathbf{S}$ is an $n \times n$ matrix; the ME distribution is said to be of order n. If $\mathbf{S}$ is diagonalizable, i.e. there exist then an $n \times n$ matrix $\mathbf{P}$ and an $n \times n$ diagonal matrix $\mathbf{A}$ such that $\mathbf{S} = \mathbf{PAP}^{-1}$, then a diag.ME is obtained. The LST of its CDF is rational.

Our interest in the diag.ME is threefold. First, it covers a large set of distributions including the acyclic phase-type distributions like the generalized coxian distribution, the exponential distribution, the hypo-exponential distribution or generalized Erlang, the hyper-exponential distribution or mixture of exponentials. Second, as reported in [95], a general point process can be well fitted by a renewal process having a "phase-type distribution" such as the "mixture of exponentials". Third (and most attractively) it is analytically tractable as will become clear in Sect. 4.3.5. In brief, if inter-request times of exogenous arrivals and caching durations all follow this distribution, then any inter-miss time and any overall inter-request time in a network of caches will also follow this distribution (with other parameters), as long as an additional assumption is enforced.

The CDF of a caching duration following a diag.ME of order K can be written

$$T_n(t) = 1 - \sum_{k=1}^{K} b_k e^{-\mu_{n,k} t}, \quad \text{with} \quad \sum_{k=1}^{K} b_k = 1. \tag{4.50}$$

There is no restrictions on $\{\mu_{n,k}\}_{1 \le k \le K}$ except that $T_n(t)$ must be a CDF. The following then holds.

Result 4.3 (diag.ME TTL) *The expected caching duration and the expected number of hits are, respectively,*

$$\mu_n^{-1} = \sum_{k=1}^{K} b_k \mu_{n,k}^{-1}; \qquad E[Z] = \sum_{k=1}^{K} \frac{b_k F_n^*(\mu_{n,k})}{1 - F_n^*(\mu_{n,k})}, \tag{4.51}$$

and the LST of $G_n(t)$ *given in* (4.44) *can be rewritten*

$$G_n^*(s) = 1 - \sum_{k=1}^{K} b_k \frac{1 - F_n^*(s)}{1 - F_n^*(s + \mu_{n,k})}. \tag{4.52}$$

Using (4.51) in Proposition 4.8 yields the performance metrics.

Optimal TTL distribution per content

This section addresses the following challenging question: which distribution optimizes the performance of a content caching policy and under which conditions? A partial answer will be provided in the following.

There are conflicting objectives when optimizing a caching policy. Caching has been introduced to limit wide-area DNS traffic and to speed up DNS lookups at clients. An efficient cache is then one that has a small miss rate, a high hit probability and yet a small occupancy (data item is in cache only when needed). The counter effect is an increase in the probability for the user to obtain an outdated content. Indeed, as explained in Sect. 4.3.3, contents are refreshed only upon a cache miss. Having then a high miss rate is desirable when the content is likely to change often.

In this section, we will order distributions according to the achieved performance metrics, namely the miss rate $M_{R,n}$, the hit probability $H_{P,n}$ and the occupancy $O_{P,n}$. Consider two different policies. In one policy, a content is cached for a deterministic duration D; in the other, the caching duration T has a CDF $T_n(t)$ such that $E[T] = D$. The performance metrics vary with the distribution, the rv is then explicitely appended to the notation, e.g. $O_{P,n}(T)$.

Proposition 4.10 (Optimal policy) *If inter-arrival requests at a cache have a concave CDF then the deterministic caching duration yields the most efficient caching, i.e.*

$$M_{R,n}(D) \leq M_{R,n}(T)\ , \quad H_{P,n}(D) \geq H_{P,n}(T)\ , \quad O_{P,n}(D) \leq O_{P,n}(T).$$

Proof Define $\phi(t) = 1 + M_n(t)$. We therefore have (use $E[Z] = E[M_n(T)]$ in Proposition 4.8)

$$M_{R,n}(T) = \frac{\lambda_n}{E[\phi(T)]},\ H_{P,n}(T) = 1 - \frac{1}{E[\phi(T)]},\ O_{P,n}(T) = \frac{\lambda_n D}{E[\phi(T)]}.$$

We will now prove that ϕ is concave. Recall that $M_n(t)$ is the renewal function. Differentiating twice (4.41) yields

$$\phi''(t) = m_n'(t) = f_n'(t) + \int_0^t m_n(t-x) f_n'(x) dx. \tag{4.53}$$

Since $m_n(t)$ is a positive function (it is the renewal density function), it follows that $\phi(t)$ is a concave function if $F_n(t)$ is concave (i.e. if $f_n'(t) < 0$). Using now Jensen's inequality yields $E[\phi(T)] \geq \phi(E[T]) = \phi(D) = E[\phi(D)]$ which completes the proof. ∎

As F_n is a CDF, it may not be convex and the corollary of Proposition 4.10 never applies. Finding the optimal policy when F_n is not concave is an open problem. The simulations discussed in Sect. 4.3.6 suggest however that, in this latter case, the higher the coefficient of variation, the better the hit probability and the smaller the occupancy.

The concavity of the CDF $F_n(t)$ of the inter-request times is not a strong condition. Jung, Berger and Balakrishnan [59] use Pareto and Weibull (with shape less than 1) distributions to fit collected inter-request times (cf. discussion around Assumption 4.4 in Sect. 4.3.3). These distributions have concave CDFs. Also, it is known that long-tailed distributions having a decreasing failure rate can be well approximated by a mixture of exponentials [95], whose CDF

is concave. Last, a conceptual model often used in the analysis of caches (e.g. [31, 82, 42]) is the so-called *independent reference model* (IRM). This model is equivalent to assuming that requests for a single content form a Poisson process [38]. The CDF of (exponentially distributed) inter-arrival times is also concave.

Proposition 4.10 states that deterministic caching durations are the optimal when F_n is concave (Assumption 4.4 must hold).*This does not mean that all contents should use the same constant TTL value but rather to have a fixed value per content*. For each content which receives its own deterministic timer, the hit probability is maximized and yet the occupancy is minimized, suugesting that *the content is found in the cache mainly at requests arrivals*. The next obvious question is: which deterministic value is the optimal one? This question, already posed in [19], will be addressed now.

Since the deterministic policy is optimal only when F_n is concave, we will only consider this case in the discussion. Ideally, the optimal deterministic value, $D^\star$, should maximize the hit probability and minimize the occupancy.

For concave F_n, the renewal function $M_n(D)$ is also concave (and increasing) (cf. (4.53)). By combining Result 4.1 and Proposition 4.8, it becomes clear that the hit probability $H_{P,n}(D)$ is concave increasing (and the miss rate $M_{R,n}(D)$ convex decreasing).

Introduce now the function $g(D) = 1 + M_n(D) - Dm_n(D)$. The derivative of $O_{P,n}(D)$ w.r.t. D yields $O'_{P,n}(D) = \frac{\lambda_n\, g(D)}{(1+M_n(D))^2}$. Given that $g(0) = 1$ and $g'(D) = -Dm'_n(D) \geq 0$ for any $D \geq 0$ (recall that $m'_n(D) < 0$ for concave F_n), the function g is thus always positive and so is $O'_{P,n}$. Hence, the occupancy is an increasing function of the caching duration. It is therefore not possible to maximize $H_{P,n}(D)$ while minimizing $O_{P,n}(D)$, as both increase with the caching duration D.

We believe that having a high hit probability supersedes the desire of having a low occupancy. However, the miss rate should not be minimized (its minimum is 0 when $D \to \infty$) as it directly relates to the correctness of the cached content. Cache misses must occur in order to update the content and it is of great interest to keep an unpopular content for long duration.

The proper thing to do in such a case is to solve a constrained optimization problem, looking for instance to maximize the hit probability subject to a maximal occupancy $O_{P,\max}$ (for cache size issues) and/or a minimal miss rate $M_{R,\min}$ (for correctness issues)

$$\begin{aligned} \max_D \quad & H_{P,n}(D) \\ & O_{P,n}(D) \quad \leq O_{P,\max} \left(= \text{Cste} \times \frac{\lambda_n}{\Lambda_n} \right) \\ & M_{R,n}(D) \quad \geq M_{R,\min} \end{aligned}$$

where Λ_n is the total request rate over all records at cache n. Given the monotonicity of $H_{P,n}$,

$M_{R,n}$ and $O_{P,n}$ (for concave F_n), the solution is readily found as

$$D^\star = \min\{\arg O_{P,\max}, \arg M_{R,\min}\}.$$

The maximal occupancy $O_{P,\max}$ for a given content can be for instance the fraction of the cache size that is proportional to the content's popularity $\frac{\lambda_n}{\Lambda_n}$.

Applicability to traditional DNS caches

The modern DNS cache analyzed in Sect. 4.3.4 holds the content for a locally chosen duration. Instead, in a traditional DNS cache, the caching duration is the one advocated by the answerer. What matters in the analysis of a single cache is the distribution of the caching durations and not whether the distribution is set locally or it is imposed. Therefore, the findings of Sect. 4.3.4 apply in the case of a single traditional DNS cache, as long as Assumptions 4.4–4.5 hold. Note that the model developed in [59] provides *approximate* results for a single traditional DNS cache everytime the answerer is not an authoritative server, because the authors consider a deterministic caching duration (set to the maximum value among all those observed in the responses). Instead our model yields *exact* results for both traditional *and* modern caches, regardless of the distribution chosen for the whole range of caching durations.

4.3.5 Analysis of polytree cache networks

Section 4.3.4 focused on results for a single cache. In this section, we will extend these results for the case where we have caches at multiple nodes (e.g. client, ADSL modem, Internet server provider's DNS server, authoritative server). We say that we have a *network of caches*. To analyze it, one additionally needs to consider the network topology. Assumptions 4.4–4.7 are enforced throughout this section. Requests for a given content may only flow over a tree network and exogenous arrivals are independent so that Assumption 4.6 holds. In the following we consider the particular case of linear networks for which exact results can be derived (cf. Section 4.3.5). We will move next to the general tree network case for which approximate results can be derived by enforcing an additional assumption (cf. Section 4.3.5). Last, we focus on the particular case where caching durations and exogenous inter-request times follow a diag.ME distribution (cf. Section 4.3.5). Results for this last case are interesting as the diag.ME distribution will be preserved inside the network.

Linear networks: exact results

Consider the linear network depicted in Figure 4.3. There are N caches and the disk of the authoritative server (the rightmost cache is the one of the authoritative server). By Assumption

4.4, the overall request process at cache 1 is a renewal process. By Proposition 4.7, the miss process at cache 1 (which is nothing but the request process at cache 2) is also a renewal process. Hence, all processes in this linear network of caches are renewal processes. The performance metrics at each cache are derived using Proposition 4.8.

Tree networks: an iterative procedure

The aggregation of several renewal processes in *not* a renewal process. However, it is mandatory to have a renewal process for Proposition 4.6 to hold at any high-level cache inside the network. Similarly to Approximation 4.1, we overtake this limitation by proceeding as if we do have a renewal process, and then assess the robustness of the model against situations where this is not the case. The approximate results obtained are strikingly accurate as will be seen later in Section 4.3.6. In the rest of the section, the following assumption will be enforced.

Assumption 4.8 (Aggregation) *The overall request arrival process at each cache is a renewal process.*

A direct consequence of Assumption 4.8 is that the miss process at each cache is a renewal process thanks to Proposition 4.6. Propositions 4.7 and 4.8 are also valid at any cache. For the case of cache n in isolation, the CDF of the inter-miss time at a cache, namely $G_n(t)$, is expressed as a function of the CDF of the inter-request time, namely $F_n(t)$; see (4.43).

In the case of a network, one needs to consider the inter-request time of the *aggregate* process arriving at cache n. Let $H_n(t)$ be its CDF. Equation (4.43) provides the CDF of the inter-miss time at cache n, denoted by $G_n(t)$, after replacing $F_n(t)$ with $H_n(t)$ and by using the renewal function associated with the aggregate request process, say $M_n(t)$, in (4.42). To explicitly write this equation for the case of a network of caches, additional notation is needed.

The set of children of cache n is $\mathcal{C}(n)$ with $C = |\mathcal{C}(n)|$. The rate of *exogenous* requests (if any) at cache n is λ_n; the CDF of inter-exogenous request times is $F_n(t)$. There are $C+1$ request processes at cache n. Their aggregation has a rate

$$\Lambda_n = \lambda_n + \sum_{i \in \mathcal{C}(n)} M_{R,i}. \tag{4.54}$$

The n miss processes at the children of n and the exogenous request process at cache n are all independent. Thereby, the result derived by Lawrance in Theorem 4.1 or [69, Eq. (4.1)] applies. By Assumption 4.8, the aggregate request process at cache n is a renewal process and

the CCDF of the inter-request time is

$$\bar{H}_n(t) = \frac{\lambda_n}{\Lambda_n}\bar{F}_n(t)\prod_{i\in\mathcal{C}(n)} M_{R,i}\int_t^\infty \bar{G}_i(u)du \tag{4.55}$$
$$+\sum_{i\in\mathcal{C}(n)}\frac{M_{R,i}}{\Lambda_n}\bar{G}_i(t)\lambda_n\int_t^\infty \bar{F}_n(u)du\prod_{\substack{j\in\mathcal{C}(n)\\ j\neq i}} M_{R,j}\int_t^\infty \bar{G}_j(u)du.$$

Equation (4.45) becomes

$$G_n(t) = H_n(t) - \int_0^t (1-H_n(t-x))\bar{T}_n(x)dM_n(x) \tag{4.56}$$

with $\bar{T}_n(t)$ the CCDF of the caching duration at cache n and $M_n(t)$ the renewal function associated with the aggregate request process at the same cache. Equations (4.55)-(4.56) provide a recursive procedure for calculating the CDFs $H_n(t)$ and $G_n(t)$ at each cache n of a tree network. Numerical procedures such as Romberg's method or other techniques for computing (4.55)-(4.56) recursively can be found in [93]. We consider next a special case in which closed-form expressions for $H_n(t)$ and $G_n(t)$ can be found.

Closed-form results with diag.ME random variables

In this section, we consider a tree network where caching durations at any cache follow a diag.ME distribution. Also, we will consider that the *exogenous* request process at any cache is a renewal process whose inter-request time follows a diag.ME distribution. More precisely, at a cache n we have

$$F_n(t) = 1 - \sum_{j=1}^{J_n} a_{n,j}e^{-\lambda_{n,j}t},\ \bar{T}_n(t) = \sum_{k=1}^{K_n} b_{n,k}e^{-\mu_{n,k}t}, \tag{4.57}$$

for $t > 0$. J_n and K_n are the respective orders of the diag.ME distributions. We are now in position to prove an interesting property that is another contribution of this work. This property is the self-preservation of the diag.ME distribution across a tree network as stated in what follows.

Proposition 4.11 (diag.ME preservation) *Under Assumptions 4.4-4.8 and as long as* (4.57) *is verified at each cache n of a tree network, miss processes and aggregate requests are all renewal processes whose inter-event time follows a diag.ME distribution (parameters are in the proof).*

Proof The proof rests on three arguments: (i) the miss process at each of the lowest-level caches checks Proposition 4.11; (ii) the aggregate request process and (iii) the miss process at each of the next higher-level caches verify Proposition 4.11. Arguments (ii) and (iii) will be

used repeatedly until all caches in the network are covered. By Proposition 4.6 and Assumption 4.8, the processes at hand are renewal processes. We focus then on the distribution of the inter-event time.

Argument (i): the miss process at a lowest-level cache. Let n be such a lowest-level cache; it corresponds to a leaf in a tree. The CDF of the inter-request time is given by (4.57). The renewal equation (4.41) can be written as follows

$$M_n(t) = F_n(t) + \int_0^t \sum_{j=1}^{J_n} a_{n,j}\lambda_{n,j} e^{-\lambda_{n,j}(t-x)} M_n(x)dx. \tag{4.58}$$

The solution of (4.58) is given in [81, Section 2.2.1.19] which we can differentiate to find

$$dM_n(t) = \sum_{j=1}^{J_n} \gamma_{n,j} e^{-\theta_{n,j}t} dt \tag{4.59}$$

where $(\theta_{n,j})_{1\le j\le J_n}$ are the J_n roots of the algebraic equation

$$0 = 1 - \sum_{j=1}^{J_n} \frac{a_{n,j}\lambda_{n,j}}{\lambda_{n,j} - z}, \tag{4.60}$$

and $(\gamma_{n,j})_{1\le j\le J_n}$ are the solution of the linear system

$$\left\{ 0 = 1 + \sum_{j=1}^{J_n} \frac{\gamma_{n,j}}{\theta_{n,j} - \lambda_{n,j}}, \qquad 1 \le j \le J_n. \right. \tag{4.61}$$

Combining now (4.57) and (4.59), we can apply Proposition 4.7 to rewrite (4.45) as follows

$$\begin{aligned} G_n(t) = 1 &- \sum_{j=1}^{J_n} a_{n,j} \left(1 + \sum_{k=1}^{K_n} \sum_{i=1}^{J_n} \frac{b_{n,k}\gamma_{n,i}}{\theta_{n,i} + \mu_{n,k} - \lambda_{n,j}} \right) e^{-\lambda_{n,j}t} \\ &- \sum_{k=1}^{K_n} \sum_{i=1}^{J_n} \left(\sum_{j=1}^{J_n} \frac{(-a_{n,j})b_{n,k}\gamma_{n,i}}{\theta_{n,i} + \mu_{n,k} - \lambda_{n,j}} \right) e^{-(\theta_{n,i}+\mu_{n,k})t}. \end{aligned} \tag{4.62}$$

Clearly, the inter-miss time at a lowest-level cache follows a diag.ME distribution, whose order is $J_n(K_n + 1)$ which is the number of exponentials in (4.62).

Argument (ii): the aggregate request process at a next higher-level cache. The CCDF of the inter-request time at this intermediate cache n is given in (4.55), where $F_n(t)$ is relative to the exogenous request process and $G_i(t)$ is relative to the ith cache in $\mathcal{C}(n)$, the set of children of cache n. Recall that $C = |\mathcal{C}(n)|$. To simplify the derivation of $H_n(t)$, we rewrite $F_n(t)$ (4.57) and (4.62) with a new/modified notation ($t > 0$)

$$F_n(t) = 1 - \sum_{l_0=1}^{\mathcal{L}_0} a_{0,l_0} e^{-\lambda_{0,l_0}t}, \quad G_i(t) = 1 - \sum_{l_i=1}^{\mathcal{L}_i} a_{i,l_i} e^{-\lambda_{i,l_i}t}.$$

The exogenous request rate is denoted $\lambda_0 = \sum_{l_0=1}^{\mathcal{L}_0} a_{0,l_0}\lambda_{0,l_0}$. The miss rate at the ith cache in $\mathcal{C}(n)$ is denoted $M_{R,i}$. The overall request rate at cache n becomes $\Lambda_n = \lambda_0 + \sum_{i=1}^{C} M_{R,i}$ (see (4.54)). After tedious calculations, (4.55) can be rewritten

$$\bar{H}_n(t) = \frac{\prod_{i=0}^{C} M_{R,i}}{\Lambda_n} \sum_{l_0=1}^{\mathcal{L}_0} \sum_{l_1=1}^{\mathcal{L}_1} \cdots \sum_{l_C=1}^{\mathcal{L}_C} \sum_{i=0}^{C} \lambda_{i,l_i} \times \left(\prod_{j=0}^{C} \frac{a_{j,l_j}}{\lambda_{j,l_j}} \right) \exp\left(- \left(\sum_{j=0}^{C} \lambda_{j,l_j} \right) t \right). \tag{4.63}$$

The inter-request time at the intermediate cache n follows a diag.ME distribution of order at most $\prod_{i=0}^{C} \mathcal{L}_i$.

Argument (iii): the miss process at a next higher-level cache. Argument (i) can be repeated here by carefully replacing the exogenous request process with the aggregate request process discussed in Argument (ii). We can conclude that it is enough to have the caching duration at a cache and the inter-request time at the same cache follow a diag.ME distribution for the inter-miss process at this cache to follow a diag.ME distribution. This completes the proof. ∎

The performance metrics can be found at each cache by using Result 4.3 and Proposition 4.8. It is important to start the computation with the lowest-level caches as their miss rates will be used to derive $H_n(t)$ at a higher-level cache. It is also $H_n^*(s)$ that should be used instead of $F_n^*(s)$ in Result 4.3 at each higher-level cache.

Sections 4.3.5 and 4.3.5 provide approximate results as Assumption 4.8 is not true. The robustness of our model is tested in Section 4.3.6.

4.3.6 Validation and numerical results

The objective of this section is to test the robustness of our models against violations of the main assumptions. We first address the case of a single cache by comparing the analytic results of Sect. 4.3.4 to results obtained real DNS traces. The case of a network of caches is addressed next, where the objective is to validate Assumption 4.8.

Using a real trace (single cache)

In this paragraph, we use traces collected from a real DNS cache to assess the robustness of our analysis. Our home institution Inria at Sophia Antipolis manages two DNS servers in parallel to provide load balancing. The DNS traffic at one of these servers has been collected from 21 June to 1 July 2013. The trace contains information about 2313984 resource records requested by a total of 2147 users. Processing the trace provides, for each resource record (or content):

1. the request instants (from users to Inria's DNS server);

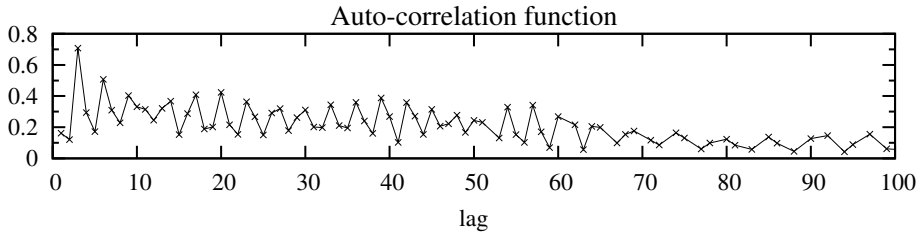
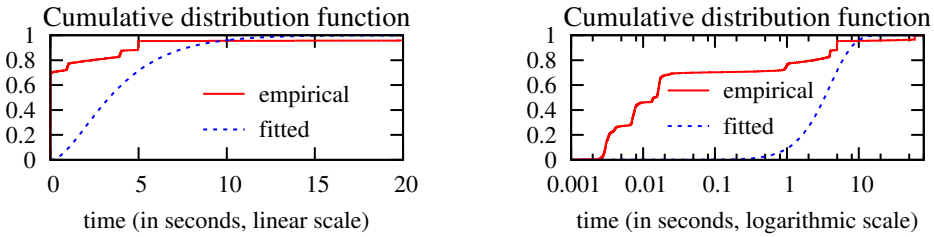


Figure 4.21: Correlated requests



(a) Inter-arrival times (s)

(b) Inter-arrival times (ms)

Figure 4.22: Arrival process fitting at second and millisecond time scales.

2. the cache miss instants (coinciding with the instants of requests from Inria's DNS server to Internet);
3. the response instants (from Internet to Inria's DNS server);
4. the final response instants (from Inria's DNS server to users);
5. the TTL values (in response packets).

A careful analysis of this trace reveals the following. First, requests instants and final responses instants do not differ much, thereby justifying our instantaneous transmission/processing assumption. Second, requests are time-varying (week day/week-end, day/night) and clearly dependent as illustrated in Fig. 4.21 for one of the contents (cf. lags 3 and 6). Therefore, Assumption 4.4 (renewal request process) is not met. *Testing our model using this trace will give insights on its robustness* since the main assumptions used in the single cache analysis are not met in this trace. Third, based on the TTLs recorded, Inria's DNS server behaves like a traditional DNS cache. For the same resource record, TTLs found in the final response packets vary from 1 to the initial TTL advocated by authoritative servers; this emphasizes the pertinence of our models as caches at the user side are given non-deterministic TTLs.

Our aim is to predict cache performance and most importantly the cache miss process as it represents the traffic that flows upstream in the DNS hierarchy (also needed for network

analysis). We selected one resource record out of the most requested among users. The caching duration of the chosen content (ranked 6th) turns out to be deterministic and equal to 2 hours (value provided directly by five authoritative servers). We used the KPC-Toolbox [21] to find the Markovian Arrival Process (MAP) that best fits the inter-request times X of the aggregated arrival process (generated by 145 different users). This tool matches with priority higher-order correlations and can convert any MAP into a renewal process having inter-arrival times identically distributed as arrivals in the MAP. The number of states of the fitted MAP is 128. The moments of the empirical inter-request time (as computed by the tool) are: mean $= 4.1614$, variance $= 4476.9$, skewness $= 83.8809$, kurtosis $= 7973.3$.

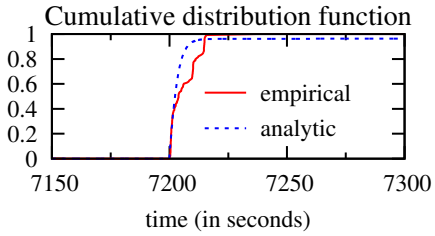


Figure 4.23: Miss process prediction.

Table 4.6: Performance metrics and relative errors (Rank 6)

Metric	Trace	Model	Rel. err. (%)
miss rate	0.00013876	0.00013749	0.920
hit probability	0.99943	0.99941	0.002
occupancy	0.99914	0.98995	0.920

Taking as input the fitted distribution and the TTL value, we use the findings of Sect. 4.3.4 to obtain the performance metrics of the cache relative to the content ranked 6th (cf. Table 4.6) and the CDF of the inter-miss times (cf. Fig. 4.23). To determine the CDF (4.48), we use a naive Riemann's sum for the integral computation. Two parameters must be set: (i) the upper bound of the integral τ, and (ii) the step length Δ. Clearly, the larger τ and the smaller Δ, the smaller the numerical error but also the larger the computational cost. We set $\tau = 720000$ (100 times the maximum between the mean inter-request time and the TTL) and $\Delta = 0.1$.

The analytic results are compared to those computed from the trace. Table 4.6 reports *negligible values of the relative errors on the performance metrics*. *Proposition 4.8 appears to be applicable even if Assumption 4.4 is not met*. In fact, we believe that it is enough to have *stationary and ergodic* point processes as requests for Proposition 4.8 to apply; cf. [10, Eq. (1.3.2), p. 21]. Lawrence's theorem [69, Eq. (4.1)] can then be replaced with [10, Eq. (1.4.6), p. 35].

Table 4.7: Analytic performance metrics and their relative errors (in percentage) at representative caches ($\lambda_1 = 1.57$ requests/s, $\lambda_2 = 0.87$ requests/s, $\lambda_3 = 1.37$ requests/s, $\lambda_4 = 0.68$ requests/s)

Cache	Performance metric	Distribution of caching durations								Trend
		deterministic		hypo-exponential		exponential		hyper-exponential		
		value	rel. err.	value	rel. err.	value	rel. err.	value	rel. err.	
1	miss rate	**0.49479**	0.00921	0.49906	0.00649	0.50039	0.08715	0.50235	0.07702	↗
	hit probability	**0.43275**	0.03832	0.42785	0.02724	0.42632	0.00660	0.42408	0.00065	↘
	occupancy	**0.35786**	0.04466	0.36094	0.04712	0.36191	0.03360	0.36333	0.02360	↗
5	miss rate	0.56708	1.1214	0.52673	0.08478	0.51681	0.10264	**0.51073**	0.00132	↘
	hit probability	0.41611	1.4561	0.46389	0.18679	0.47589	0.1514	**0.48412**	0.10321	↗
	occupancy	0.58169	1.146	0.54023	0.04850	0.53005	0.06307	**0.52379**	0.04179	↘
7	miss rate	0.52928	5.0614	0.48234	0.23668	0.46971	0.06873	**0.46045**	0.00650	↘
	hit probability	0.51789	4.536	0.52049	0.25253	0.52361	0.1067	**0.52731**	0.07069	↗
	occupancy	0.67667	5.0986	0.61667	0.19648	0.60051	0.02771	**0.58866**	0.03662	↘

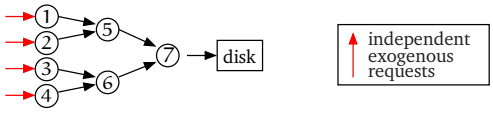


Figure 4.24: A binary tree with 7 caches.

As for the miss process, Fig. 4.23 is clear: *our model accurately estimates the CDF of the inter-miss time. Proposition 4.7 appears to be applicable even if Assumption 4.4 is not met*. This section suggests that *our single cache model is robust*.

Validating assumption 4.8

We now proceed to evaluate the robustness of our model of a network of caches. To this end, we resort to performing event-driven simulations. It is worth recalling that with exponentially distributed caching durations the model in this section coincides with the one developed in Section 4.2 to study caches that reset the caching durations at each hit. In Section 4.2, Assumption 4.8 is also used; we evaluate the robustness of the TTL-based model by comparing the approximate results it yields to exact analytic results that can be found when the conceptual IRM is used for requests. An excellent match is found which legitimates the use of Assumption 4.8. The same applies to our model when caching durations are exponentially distributed.

We consider a tree consisting of 7 caches as shown in Fig. 4.24. This tree represents well the hierarchy found in DNS: cache 7 is that of the authoritative server, caches 5 and 6 are typically those of ISP's DNS servers, and caches 1-4 are found at the client side (ADSL modem, laptop, etc.). To capture the fact that users have interleaving activity and inactivity periods, requests for all contents are assumed to form a Markov-Modulated Poisson Process (MMPP).

In other words, requests for a single content form an Interrupted Poisson Process (IPP). As a consequence, Assumption 4.8 is not satisfied at caches 5, 6 and 7 since each component (miss process) of their overall request process is not a Poisson process.

In each performed simulation, we consider a single content whose requests at each bottom-level cache form an IPP. The request rate at cache i is $\lambda_i \in [0.5, 20]$ for $i \in \{1, 2, 3, 4\}$. The caching durations at all caches follow the same distribution, with expectation in $[0.5, 1.5]$. Four distributions have been considered in the simulations: deterministic, hypo-exponential, exponential and hyper-exponential. Their respective coefficients of variation are 0, < 1, 1, and > 1.

The "exact" values of the performance metrics are those obtained after running long enough simulations. Our criterion for a long simulation is one that yields a relative incertitude on each metric less than 10^{-4}. For instance, the hit probability at cache n obtained through simulation is $H^S_{P,n}$ (the superscript S stands for "simulation"). We calculated the 99% confidence interval $[H^S_{P,n} - \epsilon, H^S_{P,n} + \epsilon]$, the relative incertitude on $H_{P,n}$ is then $2\epsilon/H^S_{P,n}$. At the end of a simulation run, the latter was at most 0.6×10^{-4}.

The approximate values of the performance metrics are those predicted by our model and are obtained by following the recursive procedure explained in Sect. 4.3.5. We have implemented a MATLAB numerical solver that determines the CDFs in the network (using (4.55)-(4.56)) and then the metrics of interest at each cache (using Proposition 4.8 where $E[Z_n] = L_n(\infty)$). The numerical error comes from the integral computation used in (4.55)-(4.56) (e.g., the integrals over infinite ranges). Again, we use Riemann's sum and, for simplicty, unique values for τ and Δ for all computations relative to a single simulation run. Consider all inter-request times and all caching durations within the network of caches. We set τ to one hundred-fold the *maximum* expectation among all these random variables, and Δ to one thousandth of the *minimum* expectation among the same random variables.

We have computed the relative error between the exact results obtained from simulations and the approximate results predicted by our model. The average relative error across all simulations on the miss rate, the hit probability and the occupancy at caches from different hierarchical levels are reported in Table 4.7 (columns 4, 6, 8, and 10). Our model is extremely accurate in predicting the performance metrics when caching durations are not deterministic as the relative error does not exceed 0.3%. For deterministic caching durations, an excellent prediction is available at bottom-level caches. The relative error increases as we consider caches at higher hierarchical levels, it reaches roughly 5% at the third level, which is nevertheless an affordable value. We conclude that *using Assumption 4.8 is not a limitation* and that *our model is very robust* to violations of this Assumption.

Optimal caching policy in a network

According to Sect. 4.3.4, if the CDF of inter-request times at a cache is concave, then the best caching policy is to cache a content for a deterministic duration. If exogenous request processes satisfy this condition, it will not be the case of the aggregate request process reaching a higher-level cache.

Consider again the simulations presented in Sect. 4.3.6. Table 4.7 reports in columns 3, 5, 7, and 9 the analytic values of the performance metrics obtained at caches 1, 5 and 7 (one cache at each level) of the synthetic network of Fig. 4.24. The trend observed on these metrics as the distribution changes from the least variable (i.e., the deterministic) to the most variable (i.e., the hyper-exponential) is shown in column 11.

The optimal values of the performance metrics are in bold fonts in Table 4.7. The best distribution at bottom-level caches (e.g., cache 1) is the deterministic one. This is predicted by Proposition 4.10 which applies here as the inter-request time of an IPP has a concave CDF. The trend on each of the metrics is inverted at higher-level caches. The deterministic policy achieves then the worst performance. The more variable a distribution, the better the performance metrics. The inter-request time at higher-level caches no longer has a concave CDF. Recall that these observations are for each content individually. The parameters of a given distribution will vary from a content to another according to the popularity.

The above trends are observed when all the caches in a tree use the same distribution. Since we have established that for concave CDF (the case of IPP requests) the deterministic distribution is the best, we repeated the simulations described earlier with the exception of having deterministic TTLs at all bottom-level caches. We observed the same trends for the same values of λ_i for $i \in \{1,2,3,4\}$ as in Table 4.7 and for another set of values that is $\lambda_1 = 0.052$ requests/s, $\lambda_2 = 0.061$ requests/s, $\lambda_3 = 0.091$ requests/s, $\lambda_4 = 0.078$ requests/s.

Our study suggests that for better performance, *deterministic caching durations should be used only at bottom-level caches*, i.e., at the client side. *Caches at servers should store contents for durations as variable as possible (large coefficient of variation)*.

4.4 Perspectives on DNS work

The TTL-based models used in this section proved to be very useful to study the modern DNS cache hierarchy. Our single cache model has been tested on real DNS traces that do not meet the renewal assumption. It predicts the performance metrics and the CDF of the miss process remarkably well. We have addressed the problem of the optimal caching duration and found that if inter-request times have a concave CDF, then the deterministic policy is the best. For non-concave CDF, our numerical analysis suggests that more variable distributions are better.

4.5 Conclusions

In this chapter, we studied two application cases: a proposal of TTL-based caches for content-routers of Content-Centric Networks and a TTL-based model for modeling modern DNS caches. We showed that our unified framework for TTL-based cache networks produces very accurate results under the assumption of renewal request processes. Several properties of TTL-based caches are established: deterministic TTL as the optimal distribution when the CDF of inter-request times is concave, closed-form characterization of processes involved at any point of the network when exogenous inter-request times and TTL are matrix-exponentially distributed, and simplified formulas for performance metrics. Moreover, we pointed out the usefulness of our model to approximately describe the metrics of interest and the miss streams of other caching systems running popular replacement policies such as LRU, RND and FIFO. Hence, our theoretic studies open the perspective to build an accurate methodology to analyze, dimension, predict performance of general and heterogeneous networks of LRU, RND and FIFO caches. This challenging research opportunity will be investigated in the next chapter.

5

Approximate analysis of general and heterogeneous networks of LRU, FIFO and Random caches

5.1 Summary

In this chapter, we propose an approximate methodology to assess the performance of general and heterogeneous cache networks. We consider that caches may run either the Least Recently Used (LRU), First-In First-Out (FIFO) and/or Random (RND) replacement algorithms. First, we address the case where requests are independently and identically distributed (i.i.d.) and streams of requests may be approximated by renewal processes [95]. Then, we investigate the case of correlated requests where statistical correlations are described by Markov-Arrival Processes (MAPs) [49]. In the former case, we provide a detailed approach to study networks with arbitrary topology based on: (1) the characteristic time approximation [23, 72] of LRU, FIFO, and RND caches, (2) the miss stream characterization of their corresponding TTL-based models [25, 73], and (3) the routing of requests as polytrees i.e. towards several destinations [52]. Our approximate model turns out to be very accurate in comparison to other existing models [82]; moreover, in presence of correlated requests correlations we describe how our modeling approach extends to account them.

Keyword 5.1 *Approximate analysis, heterogeneous cache networks, Least Recently Used, First-In First-Out, Random, characteristic time approximation, renewal and Markov-arrival processes.*

5.2 Introduction

Over the past years, popular cache management algorithms such as Least Recently Used (LRU), First-In First-Out (FIFO), and Random (RND) have received significant attention. Nowadays, the interest has shifted from caching systems in isolation to interconnected caches. The benefits of the latter approach come from storing contents in caches that are world-widely deployed or distributed across the network. This trend is mostly driven by the recent development of content-oriented technologies such as Content Distribution Networks, social networks, Video on Demand systems, and Information-Centric architectures [3, 53]. The purpose is of adapting the network architecture to the current content usage patterns, to the potential reduction of congestion, and the improvement of content delivery speed through deployment of caches in various places in the network.

Although considerable work has focused on studying isolated caches, it is still poorly understood how networks of caches operate. These cache networks have several properties such as arbitrary topology, heterogeneous nodes, complex routing schemes, and various statistical request correlations which make their analysis, performance evaluation, and design significantly challenging. Note that exact analysis of general cache networks is extremely difficult, and requires vast computational resources. Even for an isolated cache the complexity of an exact analysis grows exponentially with the cache size B and the number of files N. Also approximations [31] available for single cache systems have a complexity of order of $O(NB)$. This explains why only the 2010 paper by Rosensweig et *al.* [82] can be found as probably the most quoted modeling attempt of general (and homogeneous) network of LRU caches. Unfortunately, their approximate methodology suffers from inaccuracies with relative errors of 16%. The existing models for networks of caches are limited by one of the followings:

(i) the network topology, which is generally assumed to be hierarchical or tree-based [23, 72, 14, 42, 67];

(ii) the routing/forwarding schemes [23, 72, 20, 42], which are restricted to the case where requests flow in the same direction, e.g. from-children-to-parents forwarding schema of tree network;

(iii) the cache replacement policies [82], which are required to be identical across all caches;

(iv) the Independent Reference Model (IRM) or Poisson approximation [38], which is inaccurate [57] and commonly used to represent request streams [82]; and finally

(v) computational complexity due to exact calculation [25, 73], which can be significant especially for large networks.

Clearly, there is a lack of tools for performance evaluation of general and heterogeneous cache networks that can help engineers to gain insights into how interconnected caches perform, to test several network configurations, and to design their own network without investing too much effort and resources on test-beds. We address these five issues and derive approximate methodologies (and algorithms) by leveraging the notion of *cache characteristic time* for isolated caches, techniques of approximating general request processes by *renewal processes*, exact theoretic results on *performance metrics calculation* of TTL-based caches, and modeling approaches of complex *routing* schemes. Our modeling attempt is validated through extensive event-driven simulations.

The chapter is organized as follows. Section 5.3 presents relevant work under which our approach relies. It also shows the limitations of existing methodologies. We introduce our network model, assumptions and solution schema in Section 5.4. Section 5.5 describes the approximate models and algorithms for studying single LRU, FIFO, and RND caches. We also report some misconceptions on characteristic times found in the literature [72]. In Section 5.6, we focus on performance analysis of cache networks fed by independent and identically distributed (i.i.d.) requests and we present challenges in terms of network primitives that we address by translating the latter into well-known operations on request processes. We also describe how we can account for correlated request streams. Our modeling approach is evaluated in Section 5.7 and our findings are summarized in Section 5.8.

5.3 Related works

The closest work, methodologically speaking, to ours is the 2010 paper by Rosensweig, Kurose and Towsley [82]. The authors provide an in-depth analysis of sources of inaccuracies but also a simplified methodology for performance evaluation of general LRU cache networks. Their approach is built on top of the LRU cache approximation developed by Dan and Towsley [31] which has a complexity of order of $O(NB)$ where N is the number of files and B the cache size. Besides the complexity which can be significant for typical content-oriented networks (i.e. large values of N and B), [82] has identified the three potential sources of prediction error of their analysis of general and homogeneous networks of LRU caches: (e_1) the inaccurate performance metrics calculation of [31], (e_2) the violation of the IRM (or Poisson) assumption on the miss streams of LRU caches (already shown by Jelenkovic and Kang [57]), and (e_3) the inaccurate characterization of the aggregated request process that feed a cache within a network (this process may result from the superposition of miss streams other caches and exogenous requests).

Che et *al.* [23] proposed an alternative approach to that of Dan and Towsley [31] for the calculation of performance metrics of LRU caches. Their model is based on the *cache charac-*

teristic time and was experimentally proved to be very accurate on hierarchical (more precisely on two-level tree-based) cache networks. To some extent, the *characteristic time approximation* (CTA) on single LRU caches [23] addresses the source of prediction error (e_1) with a precision comparable to that of [82]. However, the complexity of the cache performance calculation using the CTA is not well established as the iterative procedure of [31, 82]. However, the network model in [23] still suffers from: (e_2) i.e. the characterization of the miss streams of LRU caches, the huge complexity of the exact characterization [69] of the aggregated request process at the root, the limitation of tree cache networks, and the strong dependence on Poisson request streams at leaves.

Martina et *al.* [72] experimentally extended the results of Che et *al.* [23] in three orthogonal directions. First their simulations showed that the *characteristic time approximation* can be applied to other cache replacement policies such as LRU, FIFO, RND and variants. Second their numerical results showed that the Poisson request streams assumption at leaves can be relaxed and non-IRM models such as renewal request processes can be safely considered. Third their analysis is carried out on arbitrary hierarchical (tree) cache network. However, their network model did not successfully address errors (e_2) and (e_3) since miss streams of requests and aggregated request processes are approximated by Poisson processes. Also, they stated that their study on a tandem of two caches "*with requests flowing in the same direction*" is enough to analyze general cache networks. We shall see in Section 5.6 that several operations such as *aggregating, splitting of request streams*, and also "*requests flowing in opposite direction*" cannot be handle or trivially derived from their tandem of two caches example.

In *Chapter 3* we studied TTL-based caches which have been previously shown in *Chapter 2* to describe and model the behavior of LRU, FIFO and RND caches under certain conditions (general stationary request processes, large number of files N and/or large cache capacity B). These latter conditions where sufficient to theoretically establish the validity and the accuracy of the *characteristic time approximation* (CTA) of [23, 72]. Networks of TTL-based caches are studied in *Chapter 4* under the assumption that all request processes involved in the network may be approximated by renewal processes. Simulations showed that performance metrics at each node can be obtained with errors less than 5%. Hence, combining the CTA and TTL-based network models, one can directly tackle the three sources of prediction errors (e_1), (e_2) and (e_3) on general and heterogeneous networks of LRU, FIFO and RND caches. However, the exact analysis carried out in *Chapter 4* is computationally expensive since it requires the evaluation of integrals over infinite supports and the resolution of Integral equations [25, 73]. This is definitely an handicap for the analysis of large and arbitrary cache networks.

In this chapter, we develop an approximate methodology for cache networks that addresses the numerical complexity observed in Chapter 4 while providing a similar level of accuracy. Our approach is presented in the next section and it is worth noting that our approach can be easily

extended to include other replacement policies such as q-LRU and Least Frequently Used (LFU) as introduced in [72].

5.4 Model and assumptions

In this section, we first describe the system of interest, and define the problem that we try to solve, and then explain the approach we take to solve the problem.

5.4.1 Problem statement and network model

The problem is stated by following previous effort of Rosensweig, Kurose and Towsley [82]. Let $\mathcal{G} = (V, E)$ represent a network of caches, $V = \{v_1, \ldots, v_N\}$ the set of caches, and $E \subset V \times V$ the set of connections between caches. Additionally, the file catalog is denoted by $\mathcal{F} = \{f_1, \ldots, f_K\}$. Each file is stored permanently at one or more public servers that are attached to a node in the network.

At some of the nodes $\{v_n \in V\}$ in this system, requests arrive exogenously and directly from users. When a request for file f_i arrives at a cache, it generates a cache hit if the file is located at the cache and a cache miss if not. In the latter case, the request is forwarded to other caches in the network based on the routing table at each cache, until the file is located in a cache or at the server storing the file. Then the file is forwarded along the reverse path taken by the request, and stored at each cache along the way; this network caching strategy is also known as *Leave Copy Everywhere* (LCE) by Laoutaris, Syntila and Stavrakakis [67].

If a buffer is full when a cache miss occurs, one of the files in the cache is selected based on an eviction policy to make room for the new file. In this work, we consider that caches may run one of the three most popular replacement policies, namely, LRU, FIFO, and RND. However, the model can be easily extended to include other variant replacement policies studied in [72]. Following common practice, we assume that all files have the same size (see [82] and references therein); otherwise, they can be divided in small chunks of identical size (see [41]). Hence we express the cache size in terms of the number of files/chunks it can hold at any given moment. As commonly agreed [31, 23, 82], we also assume that the request processing/forwarding times and the file download time after a cache miss are significantly smaller than the inter-request timescale. This assumption has been validated on real DNS traces in *Section 4.3.6 of Chapter 4*. Thus, once a cache miss occurs, the file is instantaneously available in the cache.

5.4.2 Processes at hand

We denote by $\mathcal{R}_{n,i} = \{t_k(n,i)\}_{k\geq 0}$ the overall request process of file f_i at cache n where $t_k(n,i)$ is the arrival instant of the $k+1$-st request, $\Lambda_{n,i}$ the intensity of the process $\mathcal{R}_{n,i}$, and

$\lambda_{n,i}$ the rate of exogenous arrival at cache v_n. In the remainder of this chapter, the following statement holds unless otherwise specified.

Assumption 5.1 (Renewal Request Process) *Exogenous requests and aggregated streams of requests may be approximated by renewal processes.*

We denote by $X_{n,i}$ the generic inter-arrival time of the process $\mathcal{R}_{n,i}$, $F_{n,i}(t) = P(X_{n,i} < t)$ its Cumulative Distribution Function (CDF), $\hat{F}_{n,i}(t)$ the CDF of its survival time (or forward recurrence time), $M_{n,i}(t)$ the Renewal Function (RF) associated to $F_{n,i}(t)$, and $F^*_{n,i}(s) = E[e^{-sX_{n,i}}]$ its Laplace-Stieltjes Transform (LST) for all $t \geq 0$ and $s \geq 0$. We refer to classical references on *renewal theory*, such as the book by Cox [28], for detailed definitions of theses quantities.

We aim at providing simple and accurate approximations of $\{\mathcal{R}_{n,i}\}$ based on *minimal available information* that engineers could easily measure or estimate at edge nodes of the network. We choose the *request rate* and *coefficient of variation* of exogenous inter-request times as inputs of our process model. This information is also translated into the *frequency* and the *burtiness* of the cache solicitations.

Whitt's approximation of general point processes [95] In this chapter, exogenous, miss and aggregated request processes will be approximated by renewal processes having hyper-exponential or shifted-exponential CDF for inter-arrival times introduced by Whitt [95]. The following characteristics of these approximate renewal processes will be intensively used.

1. If X has an Hyper-exponential CDF with parameters p_1, p_2, θ_1 and θ_2, then

   (a) the mean and the variance of X,

   $$E[X] = p_1\theta_1^{-1} + p_2\theta_2^{-1}, \qquad Var[X] = p_1\theta_1^{-2} + p_2\theta_2^{-2} - E[X]^2$$

   (b) CDF of inter-request time X,

   $$F(t) = 1 - (p_1 e^{-\theta_1 t} + p_2 e^{\theta_2 t}),\ p_1 + p_2 = 1,\ t \geq 0$$

   (c) CDF of forward recurrence time,

   $$\hat{F}(t) = 1 - \left(\frac{p_1\theta_1^{-1}}{E[X]} e^{-\theta_1 t} + \frac{p_2\theta_2^{-1}}{E[X]} e^{-\theta_2 t}\right)$$

   (d) Renewal Function associated to $F(t)$,

   $$M(t) = \frac{t}{E[X]} - p_1 p_2 \left(\frac{\theta_1 - \theta_2}{p_1\theta_2 + p_2\theta_1}\right)^2 \left(1 - e^{-(p_1\theta_2 + p_2\theta_1)t}\right)$$

(e) LST of $F(t)$

$$F^*(s) = \frac{p_1\theta_1}{\theta_1 + s} + \frac{p_2\theta_2}{\theta_2 + s}$$

2. If X has a Shifted-exponential CDF with parameters τ and θ, then

(a) the mean and the variance of X,

$$E[X] = \tau + \theta^{-1}, \qquad Var[X] = \theta^{-2}$$

(b) CDF of inter-request time X,

$$F(t) = 1 - e^{-\theta(t-\tau)},\ t \geq \tau$$

(c) CDF of forward recurrence time,

$$\hat{F}(t) = \left(1 - \frac{e^{-\theta(t-\tau)}}{1+\tau\theta}\right)\ \mathbf{1}(t \geq \tau) + \left(\frac{\theta t}{1+\theta\tau}\right)\ \mathbf{1}(t < \tau)$$

(d) Renewal Function associated to $F(t)$,

$$M(t) = \sum_{k=1}^{K} \frac{\gamma\left(k, \theta(t - k\tau)\right)}{\Gamma(k)},\ K = \left\lfloor \frac{t}{\tau} \right\rfloor,$$

where $\gamma(.)$ is the incomplete Gamma function.

(e) LST of $F(t)$

$$F^*(s) = \frac{\theta}{\theta + s} e^{-s\tau}$$

Thanks to Whitt's approximations, our task on characterization of processes reduces to determine the frequency and the burstiness of incoming and miss request streams at each node of the network. Our approach is detailed in the next section.

5.4.3 Solution schema and contributions

In this section, we describe our procedure for a single cache. The latter will be iterated on cache networks. Our approach relies on the following building blocks:

(b_1) The **Arrival Process Approximation**, in short *APA*.
This block is used to provide a simplistic description of request processes of each file at each cache of the network. It takes advantage of Assumption 5.1 and well-known approximation procedures [95, 5] to describe a general point process. The method used here is known as *moment matching* [77] since it requires knowledge of the first two moments of the inter-request times.

(b_2) The **Characteristic Time Approximation**, in short *CTA*.
The characteristic time T_i of file f_i of a single cache is defined as the maximum inter-request time of that file which leads to cache hit. This notion was initially introduced by Che et *al.* [23] on LRU caches and later extended by Martina et *al.* [72] to other replacement policies such as RND, FIFO and variants under Assumption 5.1. It was experimentally [23, 66, 72] verified that $\mathbb{E}[T_i] \approx T$, $\forall i$ especially when the rate Λ_i is negligible [41] in comparison to the total request rate $\Lambda = \sum_{i=1}^{K} \Lambda_i$. We proved in Chapter 2 and simulations by [23, 72] confirmed that LRU and FIFO caches have a deterministic characteristic time, while that of RND caches is exponentially distributed. Hence, this block allows us to study LRU, RND and FIFO caches through their corresponding TTL-based models where files are decoupled such that the analysis we derive for one file applies for the others in the same fashion.

(b_3) The **Performance metrics and Miss process Characterization**, in short *PMC*.
This block relies on exact and closed-form formulas of performance metrics of TTL-based caches derived in *Chapter 4* under Assumption 5.1. It is needed for the extension of our approximation procedure to cache network since it provides an exact characterization of the miss stream of TTL-based caches. We recall that this miss process was shown to be a renewal process when Assumption 5.1 holds. Hence, we only need to compute the exact two first moments of the cache miss process.

(b_4) The **Routing of Request Streams**, in short *RRS*.
This step is useful at any point in the network where several destinations of requests are possible. In other words, it implements routing table or routing policies (e.g. load balancing, shortest path, or Peering/Transit link of Autonomous Systems) at each node in the network. This operation is translated into *dependent splitting/thinning* request processes [52].

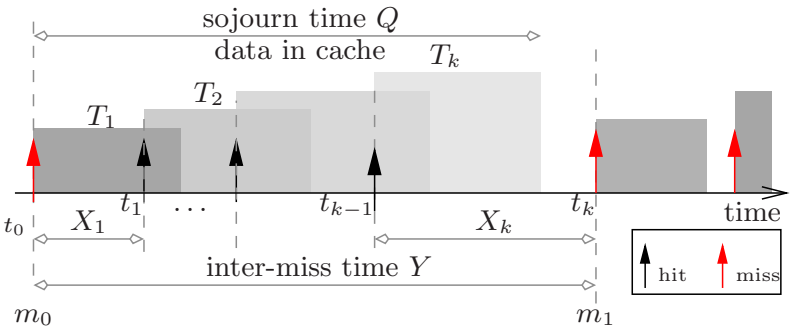


Figure 5.1: TTL-model of LRU caches

Combining blocks (b_2) and (b_3), the TTL-based model of LRU caches initializes the timer of a file when cache misses occur and later resets this timer at each cache hit as shown in Figure 5.1; while that of FIFO caches sets the timer only when a cache miss occurs as depicted in Figure 5.2. This is explained by the LRU (resp. FIFO) algorithm which inserts a requested file at the head of the cache (resp. if a cache miss happens on that file) given that the eviction is done at the tail. However, a cache hit does not change the state of a FIFO cache. According to these different TTL-models, LRU and FIFO caches can be analyzed via the class of *TTL-resetting* caches [25, 26] and *TTL-non-resetting* [73, 74] caches respectively. RND caches can be modeled by either of these classes of TTL-based policies. This is due to the memoryless property of the exponential TTL distribution of TTL-based models of RND caches. In the following, the terms *characteristic time* and *TTL* will be interchangeably used.

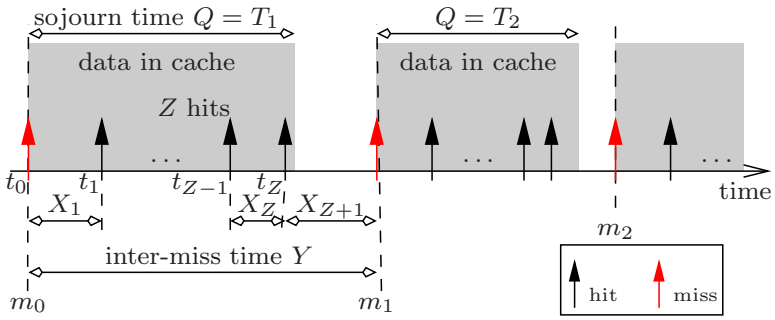


Figure 5.2: TTL-model of FIFO caches

5.4.4 General results on single cache under Assumption 5.1

In this section, we define the metrics of interest and we establish results which hold for any cache policy under Assumption 5.1. From now on we omit the subscript n that refers to the cache label in all the quantities we introduced or the ones we shall define later.

We consider the TTL T_i of a file f_i as defined in the previous section and we denote by Q_i the expected time that file f_i resides/sojourns in the cache (see the notation *data in the cache* in Figures 5.1 and 5.2). Also, Λ_i, $H_{P,i}$ and $O_{P,i}$ denote the request rate, the hit probability (i.e. probability that an arriving request finds file f_i in the cache), and the occupancy (i.e. stationary probability that the file f_i is in the cache at any time instant) for file i, respectively. For quick reference, notations of this chapter are summarized in Table 5.1.

The next result links metrics of interest and the sojourn time of a file in the cache.

Corollary 5.1 (Little's Law) *Under Assumption 5.1, the following relations hold:*

$$O_{P,i} = \Lambda_i \times (1 - H_{P,i}) \times Q_i, \ \forall\, i = 1, \ldots, K\,; \tag{5.1}$$

Table 5.1: Main notations for single cache and file f_i

Notation	Description
Λ_i	Arrival rate of file f_i (single cache)
$1/\mu_i$	Expected value of TTL T_i of file f_i (single cache)
$F_i(t)$	CDF inter-arrival times of file f_i (single cache)
$G_i(t)$	CDF inter-miss times of file f_i (single cache)
$T_i(t)$	CDF TTL duration of file f_i (single cache)
$Z^*(s)$	LST of CDF $Z(t)$
$H_{P,i}, M_{P,i}$	Hit, miss probability resp. of file f_i (single cache)
$H_{R,i}, M_{R,i}$	Hit, miss rate resp. of file f_i (single cache)
$O_{P,i}$	Occupancy probability of file f_i (single cache)

moreover, if C is the cache of size we have

$$C = \sum_{i=1}^{K} \Lambda_i \times (1 - H_{P,i}) \times Q_i \, . \tag{5.2}$$

Proof Equation (5.1) follows directly by applying Little's law since successive requests of file f_i are independent, the expected number of copies for file f_i in a cache is $O_{P,i}$, the rate at which file f_i enters the cache is the miss rate i.e. $\Lambda_i \times (1 - H_{P,i})$, and the expected time that file f_i spends in the cache is Q_i. Also, (5.2) follows from (5.1) by summing the two sides of the equality over all files and replacing $\sum_i O_{P,i}$ by the cache size. ∎

A similar result was established by Fricker et *al.* [41] for RND caches fed by Poisson request processes. Therefore, Corollary 5.1 extends their results to all caches (e.g. LRU, FIFO, RND, ...) fed by renewal request processes (i.e. under Assumption 5.1). Finally, this corollary has been proved to hold for general stationary request process in *Chapter 2, Section 2.6.2, Proposition 2.9*.

The next result is a corollary of Proposition 2.13, Section 2.6.3, Chapter 2 which provides a lower bound of the characteristic time T_i, under the CTA i.e. when we approximate $E[T_i]$, $\forall i$ by the same constant T.

Corollary 5.2 (General Inequalities under Assumption 5.1) *When* $E[T_i] \approx T$, $\forall i$*, the following inequalities hold*

$$T \leq \quad Q_i \quad , \forall\, i \tag{5.3}$$

$$\Lambda_i(1 - H_{P,i})T \leq \quad O_{P,i} \quad \leq \Lambda_i T, \ \forall\, i \tag{5.4}$$

$$T_{min} = \frac{C}{\Lambda} \leq \quad T \quad \leq T_{max} = \frac{C}{\Lambda \times \tilde{m}_p} \tag{5.5}$$

where $\bar{m}_p = \sum_{i=1}^{K} \frac{\Lambda_i}{\Lambda}(1 - H_{P,i})$ *denotes the average miss probability and* $\Lambda = \sum_{i=1}^{K} \Lambda_i$ *is the aggregate request rate.*

Now we are ready to present our models for single cache approximation.

5.5 Single cache approximation

In this section, we implement our building blocks (b_1) to (b_4) on FIFO, RND and LRU caches in isolation. Algorithms are provided and quickly evaluated.

5.5.1 FIFO cache

The hit probability $H_{P,i}$ on a file f_i in caches running the FIFO replacement policy is given by Proposition 4.8 in Chapter 4 as follows. $H_{P,i} = 1 - (1 + E[Z])^{-1}$ where $E[Z]$ is the expected number of hits during the sojourn time of the file in the cache.

Since FIFO is a TTL-non-resetting policy, the application of the CTA yields that $Q_i = E[T_i] \approx T$; and hence, $E[Z] = E[M_i(T_i)] \approx M_i(T)$. Therefore, it follows from Corollary 5.1 that

$$C = \sum_i O_{P,i} = \sum_i \lambda_i (1 + M_i(T_i))^{-1} Q_i \approx T \sum_i \lambda_i (1 + M_i(T))^{-1}.$$

We propose an iterative procedure shown in Algorithm 6 to calculate T by solving the latter fixed-point equation. We initialize the iteration as follows. Since $Q_i = T$ it follows from Corollary 5.2 that

$$\frac{C}{\Lambda} \leq T = \frac{C}{\Lambda \bar{m}_p}.$$

Using the lower bound C/Λ as the initial value of T provides fast convergence. As one can see in Figures 5.3 and 5.3, values of T are obtained after one iteration step.

Regarding the miss process of FIFO caches or more precisely the two first moment of inter-miss times (needed by (b_1) APA block), they are obtained deriving the LST $G_i^*(s)$ of the CDF $G_i(t)$ of inter-miss times Y_i derived in Proposition 4.7 of Chapter 4.

$$E[Y_i] = E[X_i](1 + L_i^*(0)) \tag{5.6}$$

$$E[Y_i^2] = E[X_i^2](1 + L_i^*(0)) - 2E[X_i] \frac{dL_i^*(s)}{ds}\Big|_{s=0} \tag{5.7}$$

where $L_i^*(0) = L_i(\infty) = M_i(T)$, $L_i^*(s)$ and its derivative are given as follows.

■ X_i has an Hyper-exponential CDF,

$$\begin{aligned}
L_i^*(s) &= \frac{1-e^{-sT}}{sE[X_i]} + \frac{p_1p_2(\theta_1-\theta_2)^2}{\delta}\frac{1-e^{-(s+\delta)T}}{s+\delta} \\
L_i^*(0) &= \frac{T}{E[X_i]} + p_1p_2\left(\frac{(\theta_1-\theta_2)}{\delta}\right)^2(1-e^{-\delta T}) \\
\frac{dL_i^*(s)}{ds}\Big|_{s=0} &= \frac{-T^2}{2E[X_i]} - \frac{p_1p_2(\theta_1-\theta_2)^2}{\delta}\frac{1-e^{-\delta T}-\delta Te^{-\delta T}}{\delta^2}, \quad \delta = p_1\theta_2+p_2\theta_1
\end{aligned}$$

■ X_i has an Shifted-exponential CDF,

$$\begin{aligned}
L_i^*(s) &= \sum_{k=1}^{K}\left(\frac{\theta e^{-s\tau}}{s+\theta}\right)^k\frac{\gamma(k,(s+\theta)(T-k\tau))}{\Gamma(k)} \\
L_i^*(0) &= \sum_{k=1}^{K}\frac{\gamma(k,\theta(T-k\tau))}{\Gamma(k)} \\
\frac{dL_i^*(s)}{ds}\Big|_{s=0} &= \sum_{k=1}^{K}\frac{(\theta(T-k\tau))^ke^{-\theta(T-k\tau)}}{\theta\Gamma(k)} - \frac{k\gamma(k,\theta(T-k\tau))}{\Gamma(k)E[X_i]}, \quad 1\le K=\left\lfloor\frac{T}{\tau}\right\rfloor.
\end{aligned}$$

Algorithm 6: Approximating characteristic time of a cache with FIFO or RND policy with a renewal process input

input : Cache Size C, cache policy P, number of files K, total arrival rate λ, precision ϵ
output: Characteristic time T

$n \leftarrow 1$
$T^{(1)} \leftarrow C/\lambda$
$\bar{m}_p^{(1)} \leftarrow 1$
do
  $n \leftarrow n+1$
  if $P =$ FIFO **then**
    $\bar{m}_p^{(n)} \leftarrow \sum_{i=1}^{K}\frac{\lambda_i}{\lambda}(1+M_i(T^{(n-1)}))^{-1}$
  else
    $\bar{m}_p^{(n)} \leftarrow \sum_{i=1}^{K}\frac{\lambda_i}{\lambda}(1-F_i^*(1/T^{(n-1)}))$
  $T^{(n)} \leftarrow C/(\lambda\times\bar{m}_p^{(n)})$
while $|T^{(n)}-T^{(n-1)}| > \epsilon$
$T \leftarrow T^{(n)}$

5.5.2 Random cache

The characteristic time of RND caches is an exponentially distributed random variable which allows us to model a RND cache as either a TTL-renewing or TTL-non-renewing cache. Considering RND as a TTL-non-resetting policy, the hit probability is given by $H_{P,i} = 1 - (1 + E[Z])^{-1}$ where $E[Z] = F_i^*(T_i^{-1})(1 - F_i^*(T_i^{-1}))^{-1} \approx F_i^*(T^{-1})(1 - F_i^*(T^{-1}))^{-1}$. Moreover, $Q_i = T_i \approx T$ and it follows from Corollary 5.1 that

$$C = \sum_i O_{P,i} = \sum_i \lambda_i (1 - F_i^*(T_i^{-1}))Q_i \approx \sum_i \lambda_i T(1 - F_i^*(T^{-1})).$$

Again, the above equality can be seen as a fixed point equation. Algorithm 6 implements an iterative scheme to compute the characteristic time T of RND caches with the initial point provided by Corollary 5.2

$$\frac{C}{\Lambda} \leq T = \frac{C}{\Lambda \bar{m}_p}.$$

As first observed in Remark 2.4 of Chapter 2 and latter confirmed by simulations in Figures 5.3 and 5.3, the characteristic times of RND and FIFO caches are approximately equal under identical traffic conditions. This implies that we can safely replace FIFO caches by RND ones for the calculation of the characteristic time T; thereby avoiding the evaluation of the incomplete gamma function several times.

Regarding the two first moment of inter-miss times (needed by (b_1) APA block), modeling RND caches as TTL-resetting caches significantly simplifies their calculation since they are obtained deriving the LST $G_i^*(s)$ of inter-miss times Y_i given in Proposition 4.2 of Chapter 4.

$$E[Y_i] = \frac{E[X_i]}{1 - F_i^*(\mu)} \tag{5.8}$$

$$E[Y_i^2] = \frac{E[X_i^2]}{1 - F_i^*(\mu)} - \frac{2 \times E[X_i]}{(1 - F_i^*(\mu))^2} \frac{dL_i^*(s)}{ds}\Big|_{s=0} \tag{5.9}$$

where $\mu = T^{-1}$; $L_i^*(s) = F_i^*(s + \mu)$, $L_i^*(0)$ and $\frac{dL_i^*(s)}{ds}\Big|_{s=0}$ are given as follows.

- X_i has an Hyper-exponential CDF,

$$\begin{aligned} L_i^*(0) &= F_i^*(\mu) = \frac{p_1\theta_1}{\theta_1 + \mu} + \frac{p_2\theta_2}{\theta_2 + \mu} \\ \frac{dL_i^*(s)}{ds}\Big|_{s=0} &= -\frac{p_1\theta_1}{(\theta_1 + \mu)^2} - \frac{p_1\theta_1}{(\theta_1 + \mu)^2} \end{aligned}$$

- X_i has an Shifted-exponential CDF,

$$\begin{aligned} L_i^*(0) &= F_i^*(\mu) = \frac{\theta e^{-\mu\tau}}{\theta + \mu} \\ \frac{dL_i^*(s)}{ds}\Big|_{s=0} &= -F_i^*(\mu)(\tau + (\theta + \mu)^{-1}) \end{aligned}$$

5.5.3 LRU cache

LRU is modeled as a TTL-renewing policy; thefore the hit $H_{P,i}$ probability of file f_i in LRU caches is $H_{P,i} = F_i(T_i) \approx F_i(T)$ thanks to the CTA. Applying Corollary 5.1 and summing over all files, we get

$$C = \sum_i O_{P,i} = \sum_i \hat{F}_i(T_i) \approx \sum_i \hat{F}_i(T).$$

The above equality is solved using the iterative scheme of Algorithm 7 by multiplying both members by T i.e.

$$T \approx \frac{C \times T}{\sum_i \hat{F}_i(T)},$$

and using the initial value C/Λ provided by Corollary 5.2. Similarly to RND caches, the first and second moments of the inter-miss times Y_i are obtained as following.

$$E[Y_i] = \frac{E[X_i]}{1 - F_i(T)} \tag{5.10}$$

$$E[Y_i^2] = \frac{E[X_i^2]}{1 - F_i(T)} - \frac{2 \times E[X_i]}{(1 - F_i(T))^2} \frac{dL_i^*(s)}{ds}\Big|_{s=0} \tag{5.11}$$

where $L_i^*(s)$ and derivatives are given as follows.

- X_i has an Hyper-exponential CDF,

$$\begin{aligned} L_i^*(0) &= F_i(T) = 1 - p_1 e^{-\theta_1 T} - p_2 e^{-\theta_2 T} \\ \frac{dL_i^*(s)}{ds}\Big|_{s=0} &= -\frac{p_1}{\theta_1}(1 - e^{-\theta_1 T} - \theta_1 T e^{-\theta_1 T}) - \frac{p_2}{\theta_2}(1 - e^{-\theta_2 T} - \theta_2 T e^{-\theta_2 T}) \end{aligned}$$

- X_i has an Shifted-exponential CDF, for $T \geq \tau$

$$\begin{aligned} L_i^*(0) &= F_i(T) = 1 - e^{-\theta(T-\tau)} \\ \frac{dL_i^*(s)}{ds}\Big|_{s=0} &= -E[X_i](1 - e^{-\theta(T-\tau)}) + (T - \tau)e^{-\theta(T-\tau)} \end{aligned}$$

  with $L^*(0) = 0$ and $\frac{dL_i^*(s)}{ds}\Big|_{s=0} = 0$ if $T < \tau$.

5.5.4 Preliminary results and remarks

Before discussing the model for a network of caches, we validate Algorithms 6 and 7 on single FIFO/RND and LRU caches in isolation. We consider caches with capacity $C = 100$ and a catalog of $K = 10^5$ files fed by two types of request processes:

- **Poisson Process:** Here, requests are generated by a Poisson process with total request rate $\lambda = 1$, and assume that file popularity follows a Zipf distribution with $\alpha = 0.7$.

Algorithm 7: Approximating characteristic time of a cache with LRU policy with a renewal process input

input : CacheSize C, number of files K, total arrival rate λ, precision ϵ
output: Characteristic time T

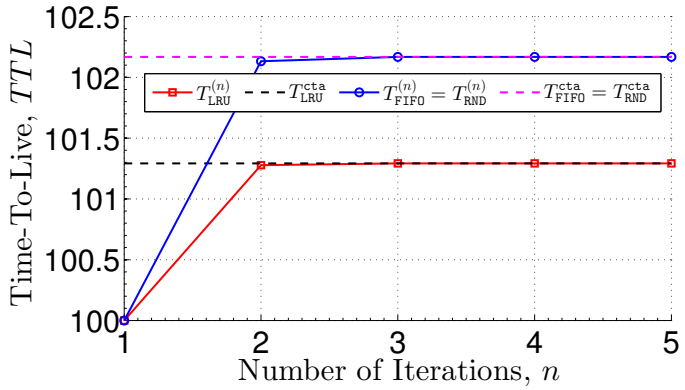
$n \leftarrow 1$
$C^{(1)} \leftarrow C$
$T^{(1)} \leftarrow C^{(1)}/\lambda$
$\bar{m}_p^{(1)} \leftarrow 1$
do
    $n \leftarrow n + 1$
    $C^{(n-1)} \leftarrow \sum_{i=1}^{K} \hat{F}_i(T^{(n-1)})$
    $T^{(n)} \leftarrow C \times T^{(n-1)}/C^{(n-1)}$
    $C^{(n)} \leftarrow \sum_{i=1}^{K} \hat{F}_i(T^{(n)})$
while $|T^{(n)} - T^{(n-1)}| > \epsilon$
$T \leftarrow T^{(n)}$

Figure 5.3(a) show that our algorithms are accurate and fast. This stresses the importance of the initialization point to $T^{(0)} = C/\Lambda$ which makes our algorithms converge after one iteration.

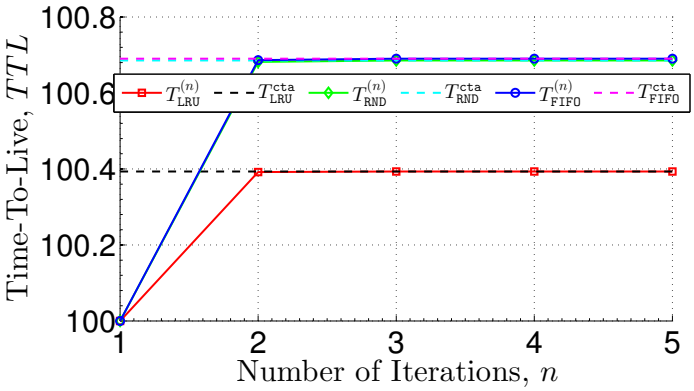
- **Interrupted Poisson Process:** In this case, request traffic is modeled by Interrupted Poisson Processes (IPP). An IPP is a renewal process having hyper-exponentially distributed inter-arrival times. The total request rate is set to $\lambda = 1$, and the content popularity follows a Zipf distribution with $\alpha = 0.7$. The squared coefficient of variation of each IPP is chosen to be $c_v^2 = 1.5$. Figure 5.3(b) shows that our algorithms converge very fast to the real value of the characteristic time.

We conclude this section with two remarks:

Remark 5.1 ($T_{LRU} \neq T_{FIFO}$) Unlike what was stated in [72], the characteristic time of a FIFO cache, T_{FIFO}, is not in general equal to that of a LRU cache, T_{LRU}; but, $T_{LRU} \leq T_{FIFO}$. This is because T_{LRU} measures the time required to observe C distinct requests, while T_{FIFO} is the time to observe C **cache misses**. For a FIFO cache in the steady state, a cache hit will not change the state of the cache. This means that files are not pushed down when a cache hit occurs (as the case for LRU caches) and hence the characteristic time of a FIFO cache is at least equal to that of a LRU cache with same size C, i.e. $T_{LRU} \leq T_{FIFO}$. Figure 5.3 confirms this result.



(a) IRM traffic/Poisson processes



(b) Renewal traffic/Interrupted Poisson Processes

Figure 5.3: Approximation of the Characteristic times of FIFO, RND and LRU caches.

Remark 5.2 (Poisson Assumption) When the arrival request for file i is assumed to follow a Poisson process of rate λ_i, the occupancy probability $O_{P,i}$ equals the hit probability $H_{P,i}$. In this case, Q_i and T_i are related through Corollary 5.1 as follows

$$Q_i = \lambda_i^{-1} \times \frac{H_{P,i}}{1 - H_{P,i}}, \ \forall\, i. \tag{5.12}$$

where $H_{P,i}$ is a function of T_i. By applying the CTA i.e. $\mathbb{E}[T_i] = T, \forall\, i$ we obtain:

$$C = \sum_i H_{P,i} \approx \begin{cases} \sum_i 1 - (1 + \lambda_i T)^{-1} & \text{, for FIFO caches} \\ \sum_i 1 - (1 + \lambda_i \mu^{-1})^{-1} & \text{, for RND caches} \\ \sum_i 1 - e^{-\lambda_i T} & \text{, for LRU caches} \end{cases} \tag{5.13}$$

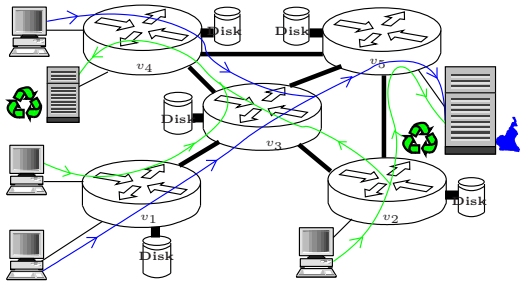


Figure 5.4: Request flows merge, split, move in opposite directions at some nodes.

We proved in *Chapter 2* that the expected characteristic times of RND and FIFO caches are asymptotically equal and our proof is supported by simulations in Figure 5.3. Having the expressions in (5.13) in hand, the equality of hit probabilities of RND and FIFO caches under IRM or Poisson request processes follows from the equality of their expected characteristic times. The former equality i.e. that of hit probabilities has been initially established by Gelenbe [43] via a Markov chain analysis, and by Martina et *al.* [72] using the insensitivity property of the $M/G/1/0$ queue. However, we bring to the attention of the reader that the approximate equality of these expected characteristic times is a new result which holds for general stationary request processes. Unfortunately, the applicability of Remark 5.2 on cache networks is limited since miss streams are no longer described by Poisson processes.

5.6 Network approximation

In this section, we extend our single cache approximation to a heterogeneous network of caches with a general topology. We recall that each file is permanently stored in at least one server connected to the network. Note that nodes also perform routing tasks; and hence, each of them maintains a routing table such that requests of each file are routed on a *tree* or *polytree* as shown in Figure 5.4.

In the case of Figure 5.4, we have two content items: one is blue and the other is green. The blue content item is located at the server connected to node v_5; requests for it are routed on the tree formed by nodes v_1 and v_4 (the leaf caches), v_3 (the intermediate cache), and v_5 (the root cache). Meanwhile the green content item is available at two different servers and its requests are routed on the polytree having nodes v_4 and v_5 as roots. We will in general refer to this tree or polytree based routing as the *routing topology*. Each node can independently choose to implement LRU, FIFO or RND for managing the cache content.

Finally, we assume that Assumption 5.1 holds i.e. exogenous and aggregated request streams may be approximated by renewal processes.

The main challenges when extending the analysis of single caches to cache networks are to describe the following network primitives in terms of operations on processes:

1. **Merging streams of requests**. This primitive is illustrated in Figure 5.5 where node 1 received requests for file 2 from both caches 2 and 3. These latter request streams are always independent thanks to our polytree-based routing topology; therefore, the aggregate request process of file 2 at cache 1 is the superposition of the exogenous request process with CDF $F_{0,1,2}(t)$ and the miss request processes with CDFs $G_{2,1,2}(t)$ and $G_{3,1,2}(t)$.

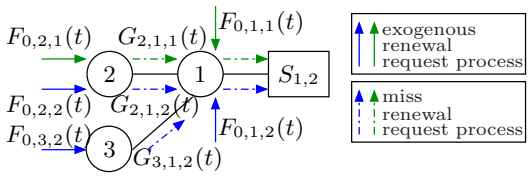


Figure 5.5: Requests are merged at cache 1: miss and exogenous processes

We rely on block (b_1) to describe this aggregate request process as a simple renewal process introduced in Section 5.4.

2. **Characteristic Time Approximation: Case of flows in "opposite" direction**. Describing the miss process of a cache requires the calculation of the characteristic time of that cache. However, the characteristic time of a cache depends on the arrival processes which could be miss processes of other caches as shown in Figure 5.6.

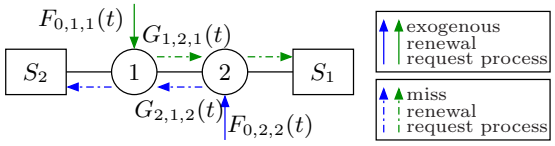


Figure 5.6: Two files requested on a Tandem of two caches

In this case, cache characteristic times are *dependent* i.e. coupled and cannot be obtained by just running Algorithms 6 or 7 in one shot as we did for cache in isolation.

3. **Splitting a stream of requests**. This network primitive should be take into account when several destinations are possible as illustrated by Cache 1 in Figure 5.7.

   This operation is known as *dependent thinning of point processes* [52, 71]. Thinning/routing can be simple (e.g. via a Bernoulli process) or complex (e.g. via Markov chains). In this work, we assume that the routing policy at each cache is simple i.e. implemented by a Bernoulli process.

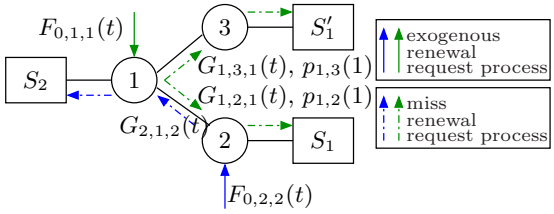


Figure 5.7: Bernoulli splitting of requests for *green* file at cache 1

Let $\mathcal{N}(i)$ denote the set of neighbor caches fed by miss requests of cache i. Assuming a Bernoulli routing scheme, we denote by $p_{i,j}(k)$ the probability that cache i forwards a missed request of file f_k to cache j. We also assume

$$\sum_{j \in \mathcal{N}(i)} p_{i,j}(k) = 1 \ , \ \forall (i,k).$$

Since exogenous, aggregated and miss processes involved in the network are renewal processes (under Assumption 5.1), Isham [52] proved that the processes resulting from Bernoulli thinning of a renewal process are also renewal processes. Taking the example of cache 1 in Figure 5.7, the LSTs of inter-request times of the thinned miss processes [52, Eq.(1.2)] are:

$$G^*_{1,j,1}(s) = \frac{p_{1,j}(1) G^*_{1,1}(s))}{1 - (1 - p_{1,j}(1)) G^*_{1,1}(s)}, \quad j = 2,3$$

where $G^*_{1,1}(s)$ is the LST of inter-miss times of file 1 at cache 1. These thinned processes may be approximated by simpler renewal processes via the moment matching technique of [95, Sect.3] by calculating the two first moments of inter-request times after sucessive derivation of the latter LSTs. One can easily show that if a renewal process with inter-request times X_i is thinned by a Bernoulli process with probability p, the resulting thinned process has inter-request times $X_{i,th}$ with the following two first moments:

$$\begin{aligned} E[X_{i,th}] &= \frac{E[X_i]}{p} \\ E[X^2_{i,th}] &= \frac{E[X^2_i]}{p} + 2\frac{1-p}{p^2}(E[X_i])^2 \end{aligned}$$

Before discussing the algorithms for the general network, we make two remarks regarding the routing schemes.

Remark 5.3 (Probabilistic Routing) In case of several possible paths, a Bernoulli routing scheme selects a path according to a probability distribution. When routing is done according to the shortest path, the probability mass distribution on the paths can be concentrated on the shortest path. This Bernoulli routing can be easily extended to more sophisticated routing schemes

that account for request history through l-dependent Markov chain thinning method described in [52, Sect.2, 3 & 4].

Remark 5.4 (Deterministic Routing) The general network of LRU caches studied in [82] was defined as follows. It is assumed that there is at least one server in the network for each content, one path from any cache to any server, and the routing is done by always selecting the shortest path, which is assumed to be unique or chosen uniformly at random in case of multiple shortest paths. This scheme is a special case of our probabilistic routing on polytrees. In fact, if $\mathcal{N}_s(i,k)$ is the set of different caches which correspond to the next hop on one of the shortest paths from cache i to a server of file f_k, it is enough to take $p_{i,j}(k) = 0$, $\forall j \in \mathcal{N}(i) \backslash \mathcal{N}_s(i,k)$ and $p_{i,j}(k) = 1/|\mathcal{N}_s(i,k)|$, $\forall j \in \mathcal{N}_s(i,k)$.

For the sake of simplicity, we will first consider the case of a cache network with polytree (physical) topology where file servers are connected to the root node(s). In this case, the routing topologies of files are polytrees made of caches of the physical network topology; and more important, *requests are assumed to flow in the same direction* e.g. from leaves/edges towards the root(s). Then, we will address the case of tandem of two caches where requests flow in opposite directions. And, finally we will extend the model to a network having a general physical topology.

5.6.1 Cache network with polytree topology

In this section, we consider a network with a polytree topology, where servers are located at root nodes. Requests are forwarded from children-to-parents and they flow in the same direction. Under this settings, we can analyze our network by starting from the leaf nodes and approximating the characteristic time and the miss processes. The latter processes are then aggregated with the exogenous request streams if any to form the arrival process at a higher level cache. We repeat this process until we reach all root nodes. The algorithm for approximating performance of cache networks with polytree topology is given in Algorithm 8.

5.6.2 Cache network with arbitrary topology

Case of two caches in tandem We consider a network of two caches connected in tandem, where the requests flow in opposite directions as shown in Figure 5.6. Moreover, we assume without loss of generality that these two caches runs the RND replacement policy to manage files in its cache. In fact, we can easily adapt our result to the case of LRU or FIFO policy.

Algorithm 8: Approximating performance of a cache network with polytree topology

input : Topology $\mathcal{G}(V, E)$, Depth d, Number of files K, Exo. processes $\{F^*_{0,i,k}(s),\ \nu_i \in V, f_k \in \mathcal{F}\}$, Size of caches $\{C_i, \nu_i \in V\}$

output: Characteristic times T_i, hit probs $H_{P,i,k}$, occupancy $O_{P,i,k}$, $\{\nu_i \in V, f_k \in \mathcal{F}\}$

while $d \neq 0$ **do** ; // Caches are different from the server

  $\mathcal{S}(d) \xleftarrow{\text{Select caches at depth } d} \mathcal{G}(V, E)$

  foreach $i \in \mathcal{S}(d)$ **do** ; // Start from Edges

    for $k = 1, \ldots,\ K$ **do** ; // On each file at a given cache

      $\{\Lambda_{i,k}, H^*_{\cdot,i,k}(s)\} \xleftarrow{\text{Whitt's Approx.}} \{F^*_{0,i,k}(s), \nu_{j,i,k}, G^*_{j,i,k}(s), j \in \mathcal{C}(i)\}$;

    end

    $\{T_i,\ \mu_i\} \xleftarrow{\text{Single Cache Approx. —— Algs. 6, 7}} \{C_i,\ H^*_{\cdot,i,k}(s),\ k = 1, \ldots, K\}$;

    for $k = 1, \ldots,\ K$ **do** ; // On each file at a given cache

      $\{H_{P,i,k}, O_{P,i,k}\} \xleftarrow{\text{Exact Metrics}} \{T_i, \mu_i, H^*_{\cdot,i,k}(s)\}$;

      $G^*_{i,\cdot,k}(s) \xleftarrow{\text{Exact Miss}} \{T_i, \mu_i, H^*_{\cdot,i,k}(s)\}$;

      $\{G^*_{i,j,k}(s),\ j \in \mathcal{N}(i)\} \xleftarrow{\text{Isham's Thinning. —— Eqs. (1.2, 2.5, 3.0), [52]}} G^*_{i,\cdot,k}(s)$;

      $\{G_{i,j,k}(t), G^*_{i,j,k}(s)\} \xleftarrow{\text{Whitt's Approx.}} G^*_{i,j,k}(s)$;

    end

  end

  $d \leftarrow d - 1$;

end

Tandem of caches We denote by $F_{0,j,k}(t)$ the CDF of inter-arrival times of exogenous requests for file f_k at cache j and by $G_{i,j,k}(t)$ the CDF of inter-miss times of requests for file f_k at cache i forwarded to cache j.

We apply Corollary 5.2 at each cache to obtain

$$C_1 = \frac{\lambda_{1,1}}{\mu_1}(1-F^*_{0,1,1}(\mu_1)) + \frac{\lambda_2(1-F^*_{0,2,2}(\mu_2))}{\mu_1}(1-G^*_{2,1,2}(\mu_1)) \quad (5.14)$$

$$C_2 = \frac{\lambda_{2,2}}{\mu_2}(1-F^*_{0,2,2}(\mu_2)) + \frac{\lambda_1(1-F^*_{0,1,1}(\mu_1))}{\mu_2}(1-G^*_{1,2,1}(\mu_2)) \quad (5.15)$$

where $G^*_{i,j,k}(s) = 1 - \frac{1-F^*_{0,i,k}(s)}{1-F^*_{0,i,k}(s+\mu_i)}$. We get the following coupled system of fixed-point equations:

$$\mu_1 C_1 = \lambda_{1,1}(1-F^*_{0,1,1}(\mu_1)) + \lambda_{2,2}(1-F^*_{0,2,2}(\mu_2)) \times \frac{1-F^*_{0,2,2}(\mu_1)}{1-F^*_{0,2,2}(\mu_1+\mu_2)} \quad (5.16)$$

$$\mu_2 C_2 = \lambda_{2,2}(1-F^*_{0,2,2}(\mu_2)) + \lambda_{1,1}(1-F^*_{0,1,1}(\mu_1)) \times \frac{1-F^*_{0,1,1}(\mu_2)}{1-F^*_{0,1,1}(\mu_2+\mu_1)} \quad (5.17)$$

These coupled equations can be solved using an iterative scheme. Given μ_j, the performance metrics at cache j for file f_k are obtained as follows:
$H_{P,j,k} = G^*_{i,j,k}(\mu_j)$ and $O_{P,j,k} = \lambda_{i,k}(1-H_{P,i,k})\mu_j^{-1}(1-H_{P,j,k})$.

Case of tandem of two caches with exogenous arrivals Let us consider now the tandem of two caches as illustrated in Figure 5.8 with exogenous request arrivals at both caches. $F_{0,j,k}(t)$ the CDF of inter-arrival times of exogenous requests for file f_k at cache j, $H_{\cdot,j,k}(t)$ the CDF of inter-arrival times of the aggregated requests for file f_k at cache j and $G_{i,j,k}(t)$ the CDF of inter-miss times of requests for file f_k at cache ν_i sent to cache j. We also denote by $\Lambda_{j,k}$ the

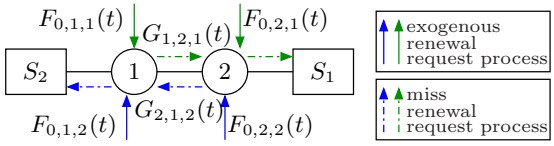


Figure 5.8: Two files exogenously requested on a Tandem of two caches

total request rate of file f_k at cache j, $\nu_{i,j,k}$ is the miss rate of requests of file f_k at cache j sent by cache i and $\mathcal{N}(j)$ is the set of caches in the next hop which are connected (might receive requests from) to cache j. The coupled fixed-point equations becomes

$$\mu_1 C_1 = \Lambda_{1,1}(1-F^*_{0,1,1}(\mu_1)) + \Lambda_{1,2}(1-H^*_{\cdot,1,2}(\mu_1)) \quad (5.18)$$

$$\mu_2 C_2 = \Lambda_{2,2}(1-F^*_{0,2,2}(\mu_2)) + \Lambda_{2,1}(1-H^*_{\cdot,2,1}(\mu_2)) \quad (5.19)$$

where $H^*_{.,i,k}(s)$ is approximated by the two moments matching techniques [5] of the superposition of the processes $\left\{F^*_{0,i,k}(s),\ G^*_{j,i,k}(s),\ i \in \mathcal{N}(j)\right\}$,

$$1 - G^*_{i,j,k}(s) = \frac{1 - H^*_{.,i,k}(s)}{1 - H^*_{.,i,k}(s + \mu_i)}, \quad \Lambda_{j,k} = \lambda_{j,k} + \sum_{\nu_i \in \mathcal{C}(j)} \nu_{i,j,k}, \text{ and } \nu_{i,j,k} = \Lambda_{i,k}(1 - H^*_{.,i,k}(\mu_i)).$$

If cache 2 is running LRU (resp. FIFO) policy, the second equation of the system (5.18) of coupled equations is replaced by

$$C_2 = \hat{F}_{0,2,2}(T_2) + \hat{G}_{1,2,1}(T_2), \qquad \left(\text{resp. } C_2 = \frac{\Lambda_{2,2}T_2}{1 + M_{0,2,2}(T_2)} + \frac{\Lambda_{2,1}T_2}{1 + M_{1,2,1}(T_2)}\right)$$

General case Thanks to these previous toy networks, we have clearly shown that calculating the performance of general cache networks requires solving a coupled system of fixed-point equations, specially when requests flows in opposite direction.

An intuitive approach would be to derive these coupled equations for each network and them solve them through an iterative method. However, this approach will be a per-case solution; instead, we develop a cache network iterative algorithm similar to Algorithms 6 and 7 for single cache approximation.

Our cache network approximation algorithm starts with an initialization step where all caches are assumed to have miss probabilities of one. The consequence is that the miss process of a node is initialized by its aggregated arrival processes. Then the TTLs at each cache are also initialized by the minimum TTL values provided by Corollary 5.2 i.e. the ratio of the cache capacity and the total request rate on the cache. After, this initialization step, our algorithm updates the miss processes of the caches using the initial value of the TTLs; and recalculates the new TTL values as described in Algorithms 6 and 7. Our cache network algorithm stops when all TTL values at all nodes of the network have converged. In this way we can iteralively solve any system of coupled equations and handle arbitrary cache network. This approach is presented in Algorithm 9.

5.6.3 General cache networks under correlated requests

In this section, we propose an implementation of the building blocks of our solution schema (see Section 5.4.3) which takes into account the correlation structure within request streams.

The (b_1) APA block is now implemented by a simple Switched Markov-Arrival Process (SMAP) as described by Horváth [49, Eqs(11-12)]. The *MAPmatch* [49, Figure 2] approximation technique requires very few inputs (i.e. the frequency, the burstiness, the skewness, the first auto-correlation lag, and possibly the second auto-correlation lag) of the original request process. We recommend to calculate input parameters of aggregated processes (which are the superposition of several MAPs in this case) before applying the Horváth's approximation. This

Algorithm 9: Approximation on arbitrary physical network of caches

input : `CacheSize` C_i, `CatalogSize` K, `ExoRates` $\lambda_{i,k}$, `Precision` ϵ

output: `CharacteristicTime` T_i, μ_i

$n := 1$;

$m^{(1)}_{p,i,k} := 1$;

$\Lambda^{(1)}_{i,k} := \lambda_{i,k} + \sum_{\nu_j \,:\, \nu_i \in \mathcal{N}(j)} \lambda_{j,k}$;

$H^{*(1)}_{\cdot,i,k}(s) \xleftarrow{\texttt{Aggr. Arr. --- Whitt's Approx.}} \{F^{*}_{0,i,k}(s),\ F^{*}_{0,j,k}(s),\ \nu_j : \nu_i \in \mathcal{N}(j)\}$;

$\mu^{(1)}_i := 1/T^{(1)}_i := (\sum_k \Lambda_{i,k})/C_i$;

$\{G^{*(1)}_{i,j,k}(s),\ \nu_j \in \mathcal{N}(i)\} \xleftarrow{\texttt{Exact Miss, Isham's Thinning \& Whitt's Approx.}} \{H^{*(1)}_{\cdot,i,k}(s), \mu^{(1)}_i, T^{(1)}_i\}$;

while `true` **do** ; `// Iterate until convergence`

  $n := n + 1$;

  $H^{*(n)}_{\cdot,i,k}(s) \xleftarrow{\texttt{Whitt's Approx.}} \{F^{*}_{0,i,k}(s),\ G^{*(n-1)}_{j,i,k}(s),\ \nu_j : \nu_i \in \mathcal{N}(j)\}$;

  $\{H^{(n)}_{P,i,k}, O^{(n)}_{P,i,k}, m^{(n)}_{p,i,k}, \Lambda^{(n)}_{i,k}\} \xleftarrow{\texttt{Exact Metrics}} \{H^{*(n)}_{\cdot,i,k}(s), \mu^{(n-1)}_i, T^{(n-1)}_i\}$;

  $m^{(n)}_{R,i} := \sum_{k=1}^{K} \Lambda^{(n)}_{i,k} m^{(n)}_{p,i,k}$;

  $C^{(n)}_i := \sum_{k=1}^{K} O^{(n)}_{P,i,k}$;

  if `RND` or `FIFO` **then**

    $\mu^{(n)}_i := 1/T^{(n)}_i := \frac{m^{(n)}_{R,i}}{C_i}$;

  else

    $T^{(n)}_i := \frac{C}{C^{(n)}_i} \times T^{(n-1)}_i$;

  end

  if $\mu^{(n-1)}_i - \mu^{(n)}_i < \epsilon$ or $T^{(n)}_i - T^{(n-1)}_i < \epsilon$ **then**

    $\mu_i, T_i :\approx \mu^{(n)}_i, T^{(n)}_i$;

    `break`;

  end

  $\{G^{*(1)}_{i,j,k}(s),\ \nu_j \in \mathcal{N}(i)\} \xleftarrow{\texttt{Exact Miss, Isham's Thinning \& Whitt's Approx.}} \{H^{*(1)}_{\cdot,i,k}(s), \mu^{(1)}_i, T^{(1)}_i\}$;

end

prevents generally a MAP representation with a large state space when applying the Kronecker sum. The (b_2) CTA, (b_3) PMC, and (b_4) RRS blocks are then adapted using our unified framework introduced in *Chapter 3*. Finally, similar procedures such as Algorithms 6, 7, and 9 are obtained.

5.7 Evaluation of the approximation under independent requests

In this section, we evaluate the accuracy of our model by computing performance metrics of cache networks: the hit probability of each file at each cache and the average miss probability of each node. We consider as "exact" the metrics of interest obtained via long event-driven simulations ($\approx$ 16.77 million of events generated). If the per-file hit probabilities are interesting from the user point of view, the per-cache average miss probabilities are other performance metrics of particular interest from the system point. In fact, the miss probability provides insights on the residual load on end-servers.

In order to appreciate the quality of our model, we also calculate the percentage of relative errors on the average hit probability and the miss probability ratio between simulation results and our predictions. A small percentage of the relative error on the average hit probability and miss probability ratios close to one are desirable and will be the global indicators of the quality of our model.

Our model assume that requests streams are described by renewal processes. We implemented our model using three classes of renewal processes where inter-requests times are drawn from hyper-exponential, shifted-exponential, and exponential distributions. As we shall see soon, the main difference among these implementations is how we calculate the characteristic of the aggregated request process at the input of a cache.

Unless otherwise specified, we consider that exogenous requests arrive at leaf/edge nodes of the network according to a Poisson process with rate $\lambda = 1$, and the file popularity follows a Zipf distribution with parameter $\alpha = 0.7$. Moreover, caches of tree networks are labeled starting from the root (which is the node with label 1 at level 1), then the most left children (is the node labeled 2) until the last child (is the node labeled k) at level 2 if the root has an in-degree equals to k, and so on. Finally, the cache capacity is set $C = 10^2$ to accommodate a total number of files $N = 10^3$ and the metrics of interest are reported for one cache per level of the tree.

5.7.1 Accuracy of the Whitt approximations

In this section, we consider that request streams at any point of the network are approximated by renewal processes having hyper-exponentially or shifted-exponentially distributed

inter-request times (See Section 5.4).

In order to characterize the superposition of independent request streams, where each of them are described by these hyper/shifted-exponential renewal processes, Whitt developed two basic techniques to compute the request rate and the square coefficient of variation of inter-request times needed to approximate the aggregated arrival process [95, Sect. 4]. These techniques are known as the *stationary-interval* and *asymptotic* methods. The former method calculates the two first moments of the inter-request times using the exact CDF of the first inter-request time (this CDF is given in [69, 10]); while the latter uses asymptotic results (over a large interval of time) on the cumulants or equivalently the moments of the random variable defined by the time-average renewal rate over a finite interval of time (see the elementary renewal theorem [28, 65] and [92, Theorems 5.1 and 5.2]).

Both techniques provide the same request rate of the aggregated process; however they return different values for the square coefficient of variation (or equivalently the variance) of inter-request times. Therefore, we will first evaluate the accuracy of our model while implementing each of Whitt's methods.

On linear networks with exogenous request at the leaf node only. In this case the overall request process at each cache is not the superposition of renewal processes but a simple renewal process. Hence, we can expect the asymptotic and the stationary interval methods to predict identical performance metrics. The interest of this case study is to evaluate the accuracy of renewal approximation itself (i.e. without the aggregating effect) which is based on the description of request streams by hyper/shifted-exponential renewal processes.

First, we consider the case where each node is running the RND replacement policy. Figure 5.9 shows our metrics of interest on a tandem network of five RND caches. As expected we can easily check that both implementations perform almost identically.

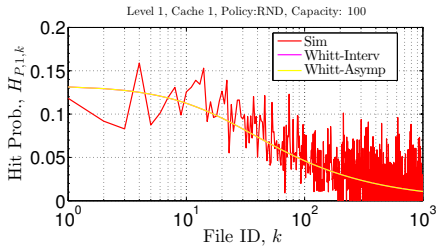
Figures 5.9(a–e) show the per-file hit probabilities at caches 1–5 where the model fairly approximates the simulations. This means that the hyper/shifted-exponential renewal processes introduced in Section 5.4 are appropriate models to describe the miss stream of a RND cache.

Although the curve of the miss probability ratio is relatively close to one (see Figure 5.9(g)), Figure 5.9(f) reveals that both implementations slightly underestimate the average hit probability with a maximum absolute relative error around 15% as depicted in Figure 5.9(h). This inaccuracy at least at cache 4 where the maximum error is observed can be explained as follows.

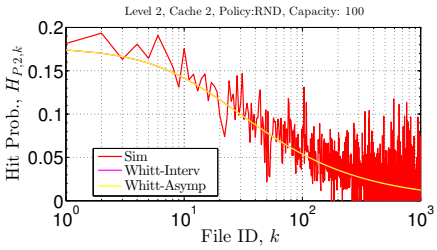
Under the characteristic time approximation (CTA), a RND cache may be analyzed as an exponential TTL-based cache. In this case, one can easily show that each inter-miss time at cache 5 may be described as the sum of two exponentially distributed random variables, namely the TTL and the inter-request time at cache 5, given that cache 5 is fed by Poisson request processes. Therefore, the LST of inter-arrival times of file i at cache 4 is $\frac{\lambda_i T^{-1}}{(\lambda_i+s)(T^{-1}+s)}$ where λ_i

is the request rate and T is the characteristic time at cache 5.

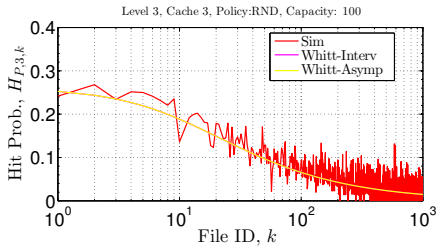
However, Whitt's approximation methods (and thus our model) cannot match exactly the latter LST since it approximates the arrival streams at cache 4 by shifted-exponential renewal processes instead of the hypo-exponential renewal processes suggested by the TTL-based caches analysis. Clearly, Whitt's approximation methods add a potential source of prediction errors at cache 4 as observed in Figures 5.9(e).



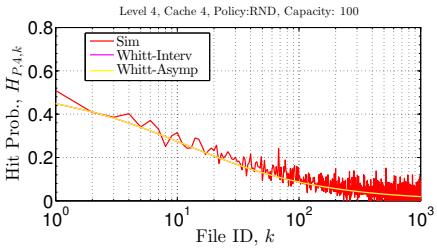
(a) Cache 1, root node



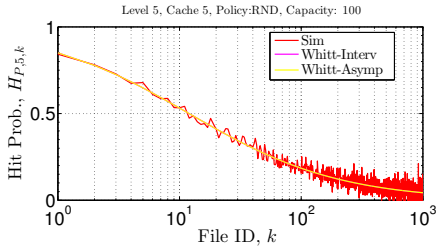
(b) Cache 2



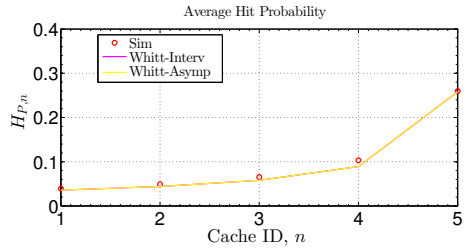
(c) Cache 3



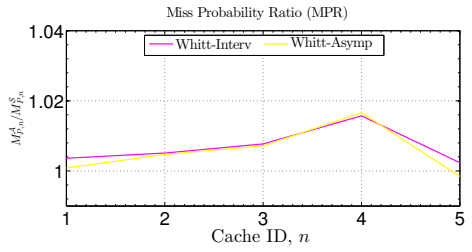
(d) Cache 4



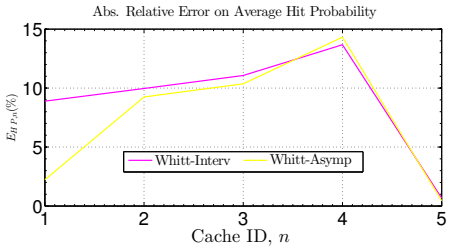
(e) Cache 5, leaf node



(f) Average Hit Probability



(g) Miss Probability Ratio



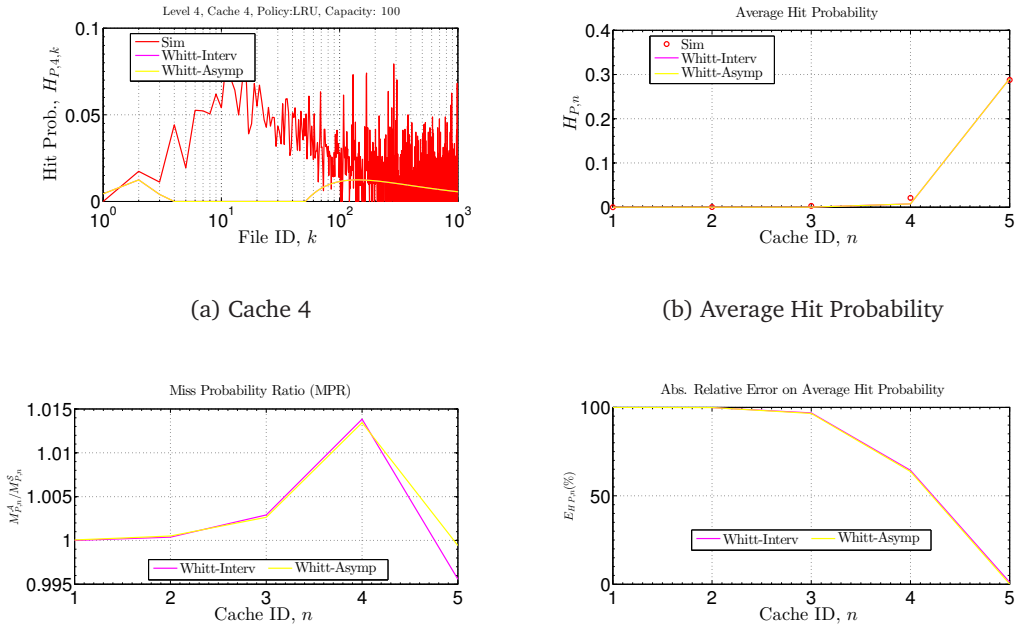
(h) Percentage of relative errors

Figure 5.9: Whitt approximations on a linear network of five RND caches, N $= 10^3$, C $= 10^2$.

We repeat the previous experiments on a tandem network of LRU caches. We only report in Figure 5.10 the per-file hit probabilities at cache 4 and the global metrics of interest at each node. In this scenario, Figure 5.10(a) reveals significant discrepancy on per-file hit probability at cache 4 between our model predictions and the simulations. Moreover, the percentage of the absolute relative errors at the same node reaches 70% as shown in Figure 5.10(d). Note also that Whitt's stationary interval method predicts a hit probability equal to zero.

As previously, this error may be explained by comparing the LST of the CDF of inter-request times at cache 4 predicted by our model and that suggested by the analysis of deterministic TTL-based caches.

In fact, a LRU cache may be modeled as a constant TTL-based cache under the CTA; therefore the LST of inter-arrival times of file i at cache 4 given by the TTL-based analysis is $\frac{\lambda_i e^{-\lambda_i T} \times e^{-sT}}{s+\lambda_i e^{-\lambda_i T} \times e^{-sT}}$ where λ_i is the request rate of file i and T is the cache characteristic time at cache 5. Clearly, the latter LST cannot be matched using the shifted-exponential renewal process of Whitt [95]; hence, this explains the error observed at cache 4.



(a) Cache 4

(b) Average Hit Probability

(c) Miss Probability Ratio

(d) Percentage of relative errors

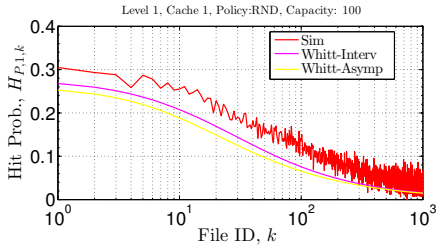
Figure 5.10: Whitt approximations on linear network of five LRU caches, $N = 10^3$, $C = 10^2$.

On binary trees of depth three. The purpose of this experiment is to assess the impact on the calculation of performance metrics when using the *stationary-interval* and *asymptotic* methods

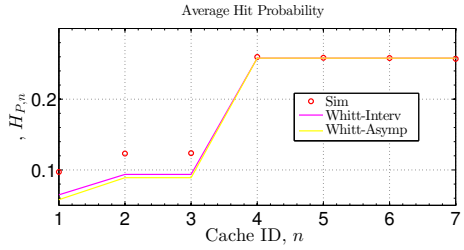
in the presence of superimposed request streams.

We consider two binary trees of depth three having seven caches; the first tree is made of RND caches only, while the second consists of LRU caches only. For both scenarios, we report the per-file hit probabilities at the root node and the per-cache average hit probabilities.

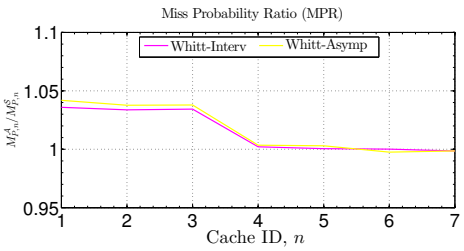
As we can see in Figures 5.11 and 5.12 respectively for the binary tree of RND and LRU caches both implementations underestimate the metrics of interest. However, the *stationary-interval* method clearly outperforms the *asymptotic* method in term of minimizing the prediction error. Therefore, we will refer to the implementation of our model using the *stationary-interval* method as the Whitt approximation in the rest of this chapter.



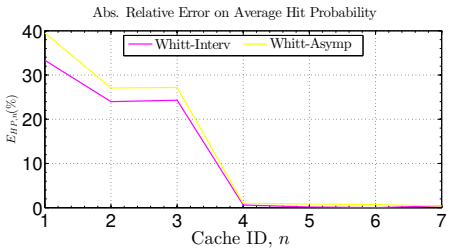
(a) Cache 1, root node



(b) Average Hit Probability



(c) Miss Probability Ratio



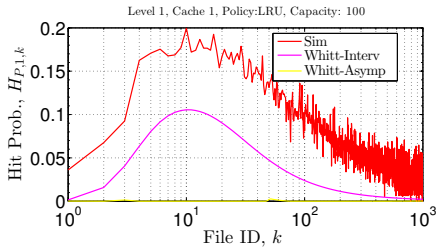
(d) Percentage of relative errors

Figure 5.11: Whitt approximations on a binary tree of RND caches, Depth 3, N $= 10^3$, C $= 10^2$.

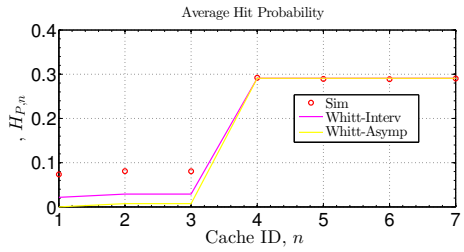
5.7.2 Accuracy of the Poisson approximation

In this section, we consider that request streams are described by renewal processes with exponentially distributed inter-request times (or equivalently Poisson request processes). This workload model is commonly used in the literature [82, 72] and it is also known as the Independent Reference Model (IRM). We refer to this implementation as the Poisson approximation.

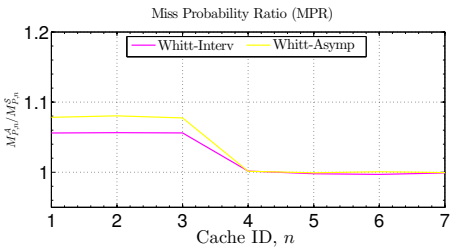
We report simulation results obtained on a tandem network and a binary tree of LRU caches of depth three respectively in Figures 5.13 and 5.14. We also test our Poisson approximation



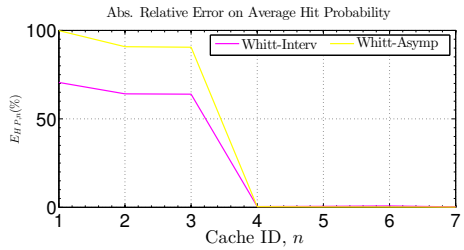
(a) Cache 1, root node



(b) Average Hit Probability



(c) Miss Probability Ratio

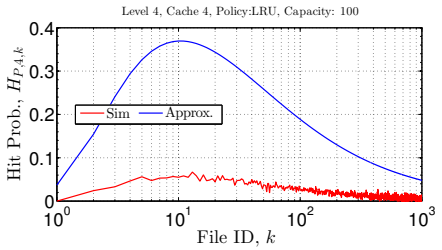


(d) Percentage of relative errors

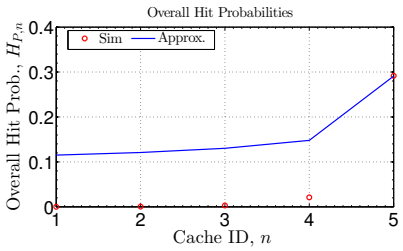
Figure 5.12: Whitt approximations on a binary tree of LRU caches, Depth 3, N $= 10^3$, C $= 10^2$.

on a tandem network and a binary tree of RND caches of depth three.

We note that the Poisson approximation overestimates the metrics of interest. As expected, the prediction errors decreases as the in-degree of the cache increases; for instance from 1 in the tandem network to 2 in the binary tree.

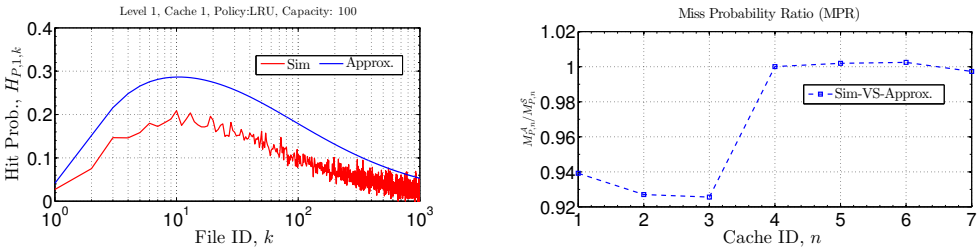


(a) Cache 4



(b) Average Hit Probability

Figure 5.13: Poisson approximation on a linear network of five LRU caches, N $= 10^3$, C $= 10^2$.



(a) Cache 1, root node

(b) Miss Probability Ratio

Figure 5.14: Poisson approximation on a binary tree network of LRU caches, Depth 3, $N = 10^3$, $C = 10^2$.

5.7.3 Accuracy of the Hybrid approximation

The previous experiments showed that the Whitt approximation underestimates our metric of interests; while the Poisson approximation overestimates the same performance metrics. As suggested in [95], we propose a method—called the Hybrid approximation—to approximate the superposition of independent request streams. Our approach is similar to the one developed by Albin in [5] and it consists of combining Whitt's *stationary-interval* method and Poisson assumption as follows. The aggregated request rate predicted by the Hybrid approximation is identical to that of Whitt and Poisson approximation; however, the coefficient of variation of inter-request times c_h^2 is computed as a *convex* combination of that obtained from the *stationary-interval* method of the Whitt approximation c_s^2 and from the Poisson approximation c_p^2 ($c_p^2 = 1$):

$$c_h^2 = w c_s^2 + (1 - w) c_p^2, \quad w \in [0, 1].$$

The parameter w is known from [5] as the weighting function; and we notice that w is close to zero (resp. one) when the in-degree of the cache is large (resp. small) i.e. the aggregating effect (resp. filtering effect) is more dominant in which case the Hybrid approximation is similar to the Poisson (resp. Whitt) approximation.

Using a small training set of k-ary trees of depth three (i.e. the root node at level 1, k intermediate nodes at level 2, and the leaf nodes at level 3) of LRU caches, we found numerical values of the weighting function w when the number of files is $N = 10^3$ and the cache capacity $C = 10^2$. These values are reported in Table 5.2 and they can be fitted using the following rational expressions for more general networks.

$$w = (1 + 0.4686 \times (k - 0.7130))^{-1}, \quad \text{at parents of leaf nodes} \tag{5.20}$$

$$w = (1 + 0.6425 \times (k - 0.9519))^{-1}, \quad \text{at ancestor nodes, i.e. parent of parents} \tag{5.21}$$

Table 5.2: Values of the weighting function w for the k-ary trees of depth $h = 3$ from our training set with settings $N = 10^3$, $C = 10^2$, LRU caches

Degree k	Depth $h = 3$	Weight. funct. $w \approx \frac{1}{1+a(k-b)}$
1	$level = 2 = h - 1$	0.875
	$level = 1 < h - 1$	0.97
2	$level = 2 = h - 1$	0.65
	$level = 1 < h - 1$	0.60
3	$level = 2 = h - 1$	0.45
	$level = 1 < h - 1$	0.40
4	$level = 2 = h - 1$	0.40
	$level = 1 < h - 1$	0.35
5	$level = 2 = h - 1$	0.35
	$level = 1 < h - 1$	0.30
6	$level = 2 = h - 1$	0.30
	$level = 1 < h - 1$	0.25
7	$level = 2 = h - 1$	0.25
	$level = 1 < h - 1$	0.20
8	$level = 2 = h - 1$	0.20
	$level = 1 < h - 1$	0.15

In the reminder of this section we use these values of weighting function w to compute the request rate and the square coefficient of variation c_h^2 of inter-request times of each arrival request process. Then, we approximate all request streams in the network by hyper/shifted-exponential renewal processes introduced in Section 5.4. Finally, we investigate the accuracy of the Hybrid approximation on large, heterogeneous, and hierarchical cache networks having up to 1093 nodes; and we also report results on a five LRU caches general network.

Unless otherwise specified, we recall that the cache capacity is set to $C = 10^2$, the total number of files is $N = 10^3$, exogenous request streams are described by Poisson processes with total rate $\lambda = 1$, and popularity of files follow a Zipf distribution of parameter $\alpha = 0.7$.

Homogeneous tree networks

In this section, we show simulation results obtained on homogeneous k-ary tree networks where caches are running the same replacement policy. These cache networks were not in the training set used to calculate the values of the weighting function w obtained on k-ary tree

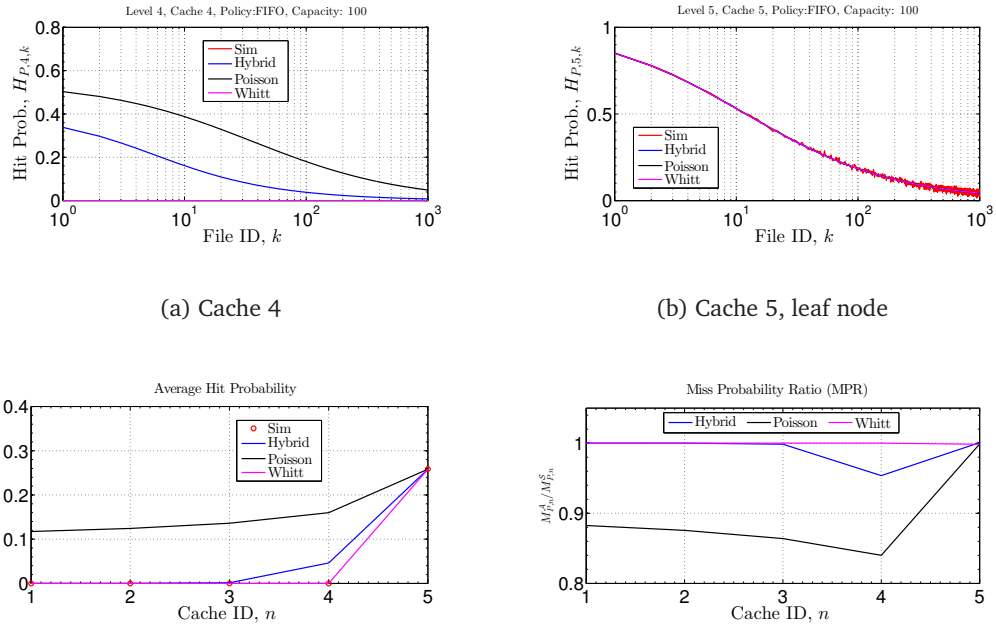
networks of LRU caches of depth three at most (see Table 5.2).

FIFO policy everywhere. First we consider the case of a tandem network of five FIFO caches. We report in Figure 5.15 the per-file hit probabilities at caches 4 and 5, and also the average hit probability at each node of the network.

As expected from the analysis of single FIFO cache, an accurate match on the per-file hit probabilities between our model and simulation results is found at cache 5 (see Figure 5.15(b)).

However, Figures 5.15(a), 5.15(c), and 5.15(c) show that the Whitt approximation provides a more accurate prediction of all metrics of interest than that obtained from the Hybrid approximation. This suggests that the weighting function w for FIFO caches with small in-degree (as in this tandem network) should be set to one ($w = 1$) at all caches of this tandem network.

This conclusion is not surprising if we look carefully at the LST of inter-arrival times of file i at cache 4 (or equivalently the inter-miss times at cache 5) for instance. The TTL-based model predicts that this LST is $\frac{\lambda_i e^{-sT}}{\lambda_i + s}$ where λ_i is the request rate of the Poisson process and T the cache characteristic time at cache 5. Meanwhile, the Whitt approximation can exactly match the latter LST since it corresponds to a shifted-exponential CDF.



(a) Cache 4

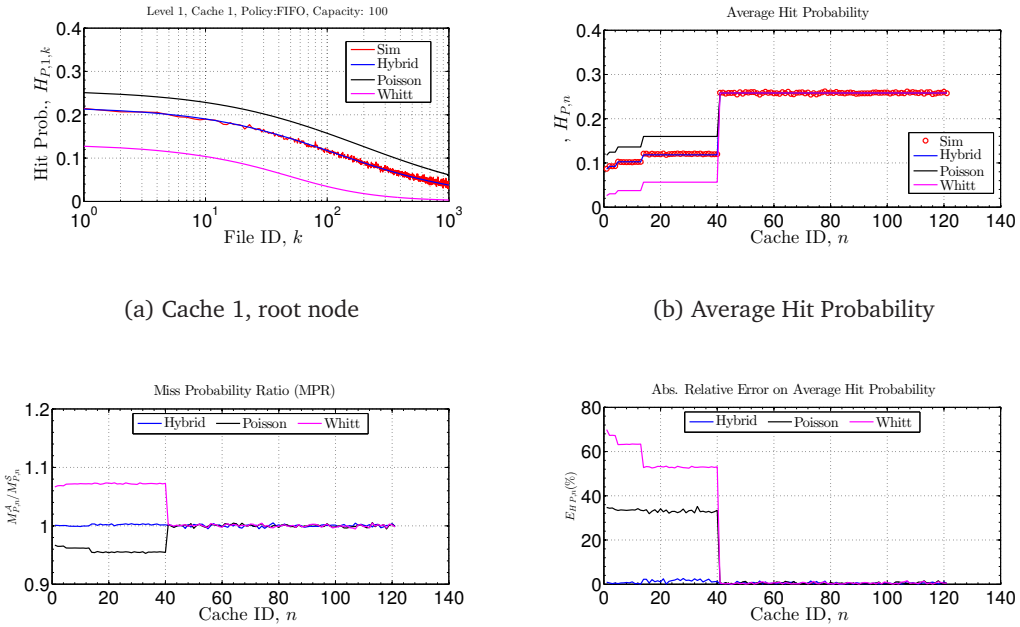
(b) Cache 5, leaf node

(c) Aggregate Hit Probability

(d) Miss Probability Ratio

Figure 5.15: Comparison of all approximations on linear network with five FIFO caches, $N = 10^3$, $C = 10^2$.

We investigate the quality of the Hybrid approximation on a ternary tree of depth five having 121 FIFO caches. This test case provides insights on the accuracy of our model when accounting for the aggregating effect. We report numerical results in Figure 5.16. For all metrics of interest the Hybrid approximation outperforms both the Whitt and Poisson approximations. Moreover its relative error and miss probability ratio are respectively less than 5% and close to one for all caches of the tree.



(a) Cache 1, root node

(b) Average Hit Probability

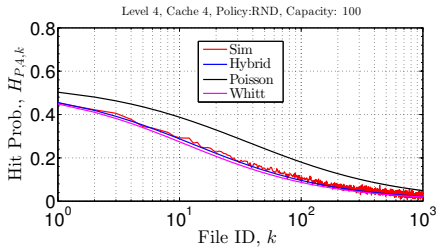
(c) Miss Probability Ratio

(d) Percentage of relative errors

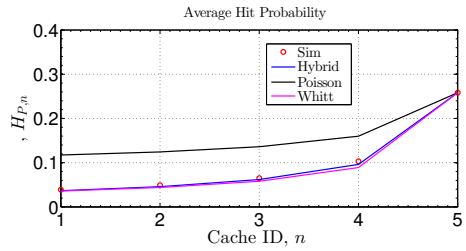
Figure 5.16: Comparison of all approximations on ternary tree of depth five with 121 FIFO caches, N $= 10^3$, C $= 10^2$.

RND policy everywhere. We also consider a tandem and, later, a ternary tree of depth five having 5 and 121 RND caches respectively. We observed in Figures 5.17 and 5.18 that the Hybrid approximation is very accurate in both cases with relative errors less than 5%.

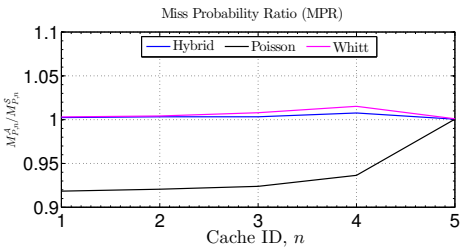
LRU policy everywhere. We consider a tandem network of five LRU caches and, then, two ternary tree networks of depth five and seven having 121 and 1093 LRU caches respectively. As shown in Figures 5.19, 5.20, and 5.21 our Hybrid approximation predicts all performance metrics with high accuracy.



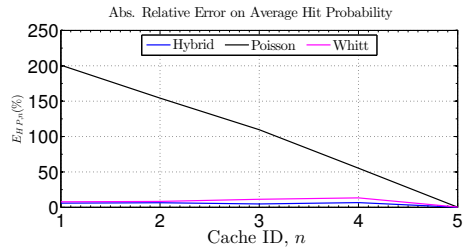
(a) Cache 4



(b) Average Hit Probability



(c) Miss Probability Ratio



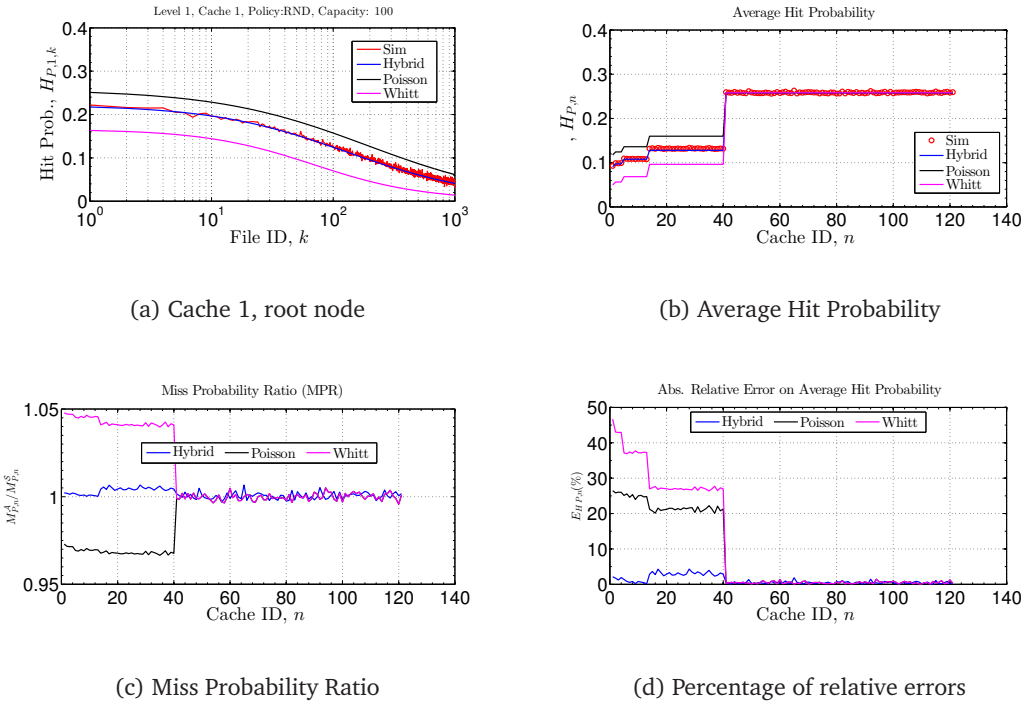
(d) Percentage of relative errors

Figure 5.17: Comparison of all approximations on linear network with five RND caches, $N = 10^3$, $C = 10^2$.

Meanwhile, by crossing simulation results obtained on tandem networks of FIFO, RND, LRU caches respectively in Figures 5.15(c), 5.17(b) and 5.19(b) we can see that LRU policy provides the best average hit probability at leaf node (i.e. at cache 5); while RND policy is more efficient at core node (e.g. cache 4). This simulation result was already suggested by the analysis of TTL-based caches where a deterministic TTL values were recommended to be used at a cache (such as leaf nodes of our experiments) if the inter-request times have a concave CDF; while TTL values with high coefficient of variation as possible are suitable for core nodes (i.e. when the CDF of inter-requests is no longer concave).

In all of the homogeneous k-ary tree networks we tested, our Hybrid approximation accurately predicts all performance metrics with excellent quality indicators i.e. the curve of the per-cache miss probability ratio close to one and the percentage of relative errors on the per-cache average hit probabilities less than 5%.

Next, we shall present preliminary results on heterogeneous tree networks.



(a) Cache 1, root node

(b) Average Hit Probability

(c) Miss Probability Ratio

(d) Percentage of relative errors

Figure 5.18: Comparison of approximations on ternary tree of depth five with 121 RND caches, $N = 10^3$, $C = 10^2$.

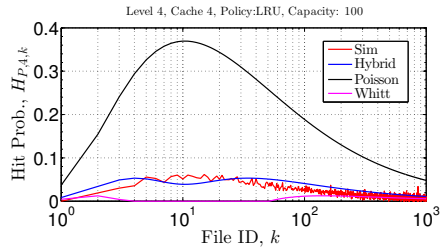
Heterogeneous tree network

Mix of FIFO, RND, and LRU caches. We first consider two ternary trees of depth five where the replacement policy at each cache is chosen uniformly at random among the FIFO, RND, and LRU policies. In the first case, the cache capacity is $C = 10^2$; while in the second case the latter parameter is set to $C = 10$.

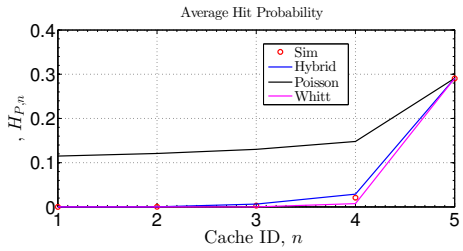
For both scenarios, we report the per-file hit probabilities at cache 9 located at level 3 of each tree. In the first case, this node is running the RND policy on cache size of 10^2; while the policy and the cache capacity are set to LRU and 10 in the second cache. Figures 5.22 and 5.22 show an accurate match between our Hybrid approximation and the simulation results.

Mix of replacement policies, variable cache capacity, and larger number of files. We consider a 4-ary tree of depth five having 341 caches. The cache capacities and the replacement policies are respectively chosen uniformly at random within the interval $[50; 150]$ and among the FIFO, RND, and LRU policies. The total number of files is set to $N = 10^4$.

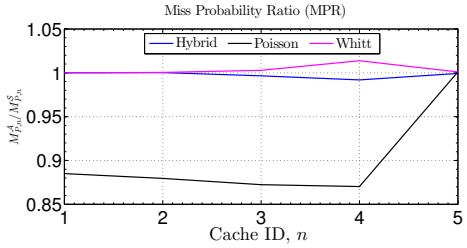
Figure 5.24(d) shows that the percentage of relative error on the average hit probability may reach 20–30% meaning that the values of the weighting function w we used should depend



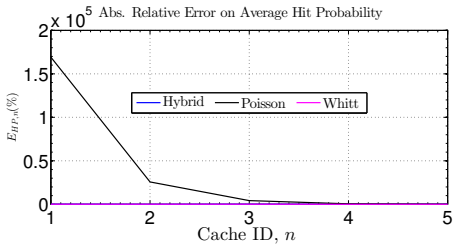
(a) Cache 4



(b) Average Hit Probability



(c) Miss Probability Ratio



(d) Percentage of relative errors

Figure 5.19: Comparison of all approximations on linear network with five LRU caches, N = 10^3, C = 10^2.

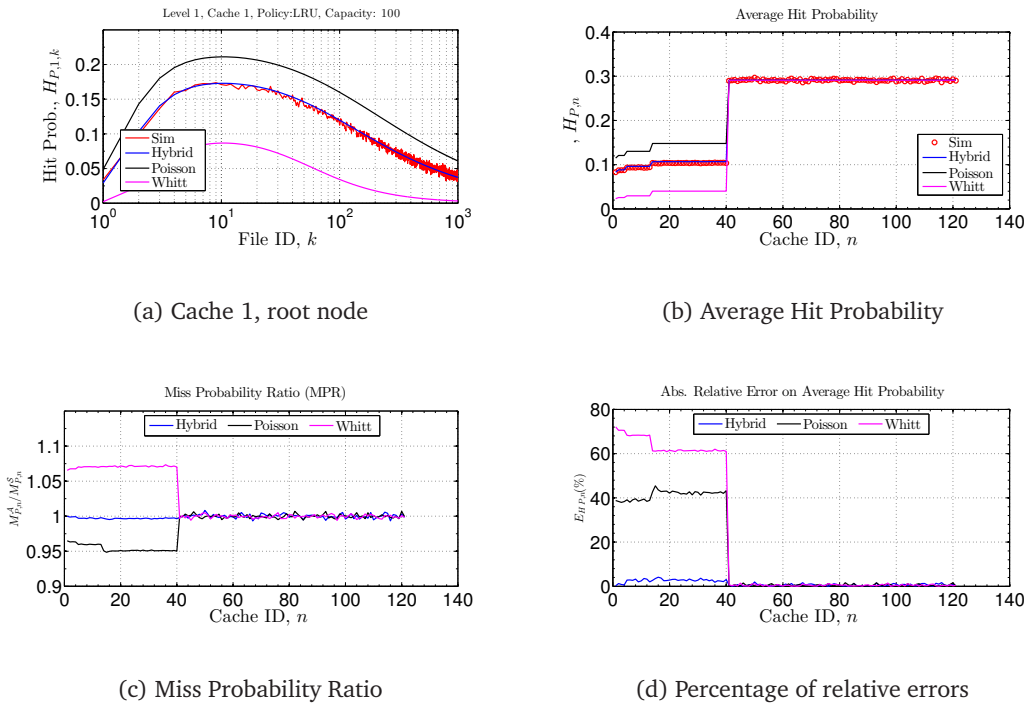
on the parameters of the system that might be the *in-degree of the cache, level of the cache in polytree routing topology, number of files, cache capacity*. This claim can be justified by the analysis carried out in [5] as shown in Section 5.7.4.

Five caches general network

In this section, we calculate the performance metrics of the five caches general network presented in Figure 5.4. The interest for this toy network is to investigate the accuracy of our model specially when requests are flowing in *opposite* directions in the network. In this case, our general Algorithm 9 is needed.

Two classes of files are requested and the total number of files is N = 10^3. Files labels from 1 to 500 (resp. 501 to 1000) belong to the first (resp. second) class of files which is accessed according to Poisson processes with total rate $\lambda = 1$. The access rates of files are modulated by a Zipf distribution of parameter α_1(resp. α_2) = 0.7.

Files of first class is routed as a polytree with maximum depth equal three; meanwhile those of second class are routed as tree. Hence, caches do not have the same level and degree with respect to each class of files and each routing topology. We recall that this information is needed



(a) Cache 1, root node

(b) Average Hit Probability

(c) Miss Probability Ratio

(d) Percentage of relative errors

Figure 5.20: Comparison of all approximations on ternary tree of depth five with 121 LRU caches, N $= 10^3$, C $= 10^2$.

to select the appropriate value of the weighting function. For instance, cache 3 has a degree of 2 and belongs to level 2 while cache 5 has a degree of 1 and belongs to level 2 of the polytree routing topology of files of class 1.

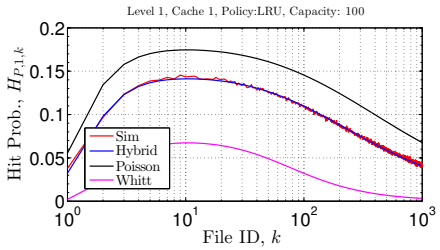
We implement our model using the weighting functions in (5.20) and (5.21) depending on the position of each cache in each routing topology (defined for each class of files).

LRU policy everywhere. First, we consider the case where caches are running the LRU policy. Performance metrics at caches 1 and 2 are similar to that of single cache in isolation; hence, we will report simulation results only at caches 3, 4, and 5 since their characteristic times are coupled due to requests flowing in opposite direction between caches 3 and 4.

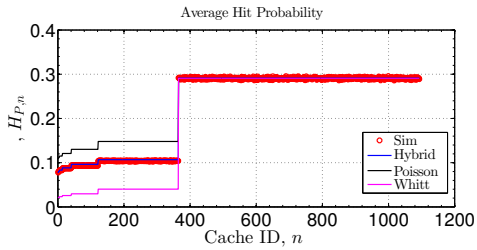
As shown in Figure 5.25, our hybrid approach is more accurate than others and predicts the per-cache average hit probabilities with a maximum relative error less than 8% in this test case.

5.7.4 Discussion on the weighting function w

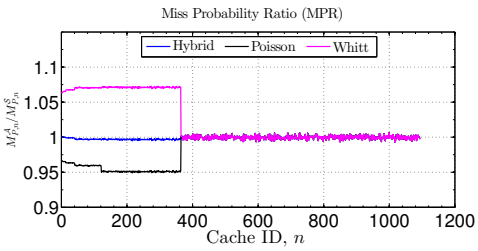
A suitable and appropriate weighting function w should be ideally network "*topologically free*". Instead of looking for expressions of w in function of the (in-)degree and the depth of



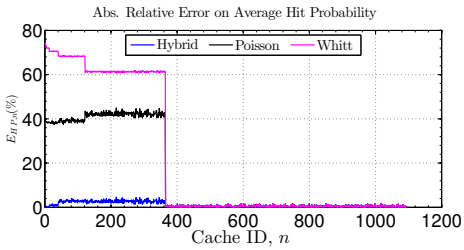
(a) Cache 1, root node



(b) Average Hit Probability



(c) Miss Probability Ratio



(d) Percentage of relative errors

Figure 5.21: Comparison of all approximations on a ternary tree of 1093 LRU caches, Depth 7, $N = 10^3$, $C = 10^2$.

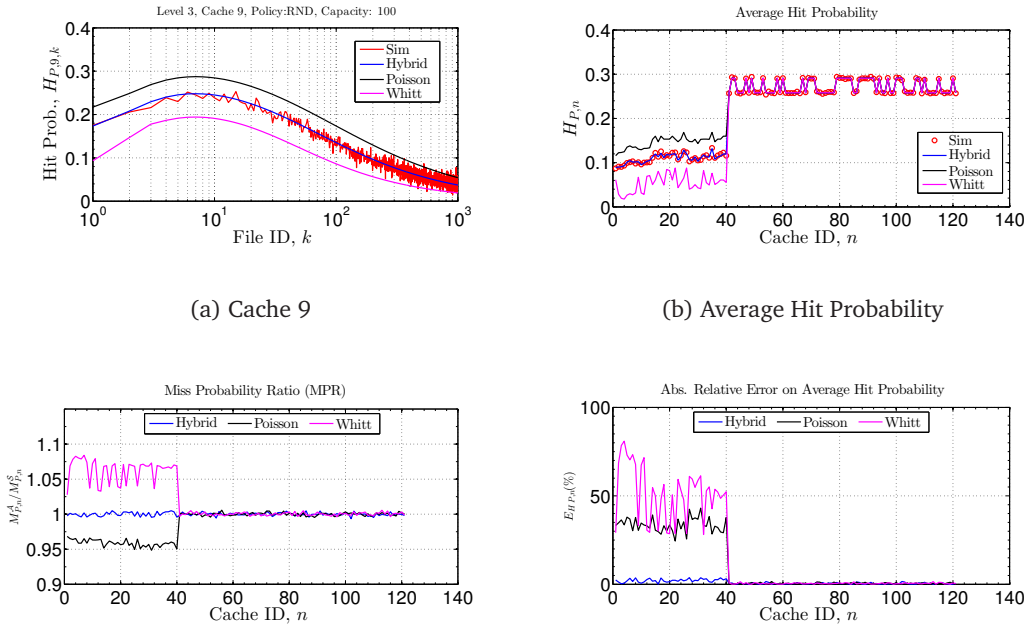
a cache as in (5.20) and (5.21) which were suggested by Table 5.2, we consider other cache parameters.

First, the **effective degree** d_{eff} defined in [5, Eq. (2)]. Consider a cache with degree d and a file i which is accessed on this cache through each of its k-th child with a rate $\lambda_{i,k}$. We denote by λ_i the total request rate of file i on our cache i.e. $\lambda_i = \sum_k \lambda_{i,k}$. The effective degree d_{eff} of our cache is given by

$$d_{eff} = \left[\sum_{k=1}^{d} \left(\frac{\lambda_{i,k}}{\lambda_i} \right)^2 \right]^{-1} \tag{5.22}$$

We can easily check that if $\lambda_{i,k} = \lambda_{i,k'}, \forall k, k'$ or $d = 1$, then $d_{eff} = d$. This was the case for most of the homogeneous tree networks of our training and test sets. Note also that if the cache has an exogenous request stream, the sum in (5.22) should start from $k = 0$ where $\lambda_{i,0}$ is the request rate of the exogenous request process.

Second, the **occupancy probability** $O_{P,i}$ of a file i in the cache. This parameter is the probability to find file i in the cache at a random instant. By analogy, $O_{P,i}$ is the cache equivalent of the intensity ρ of a queue which is also the stationary probability that there is a customer in the server at a random time instant. A nice and simple approximation of $O_{P,i}$ can be derived



(a) Cache 9

(b) Average Hit Probability

(c) Miss Probability Ratio

(d) Percentage of relative errors

Figure 5.22: Comparison of all approximations on a heterogeneous ternary tree of 121 caches, Depth 5, $N = 10^3$, $C = 10^2$.

from (5.4) and (5.5) which establish tight lower bounds of the occupancy probability $O_{P,i}$ and the characteristic time T as follows.

$$O_{P,i} \approx \frac{T}{E[X_i]} = \lambda_i T \approx \lambda_i T_{min} = \lambda_i \frac{C}{\Lambda} \tag{5.23}$$

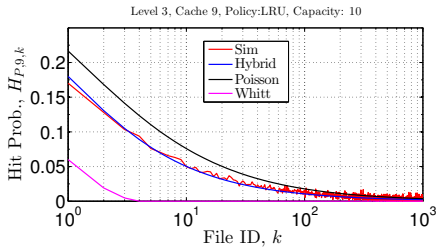
where C is the cache capacity and $\Lambda = \sum_i \lambda_i$ is the total request rate on the cache.

Using results and conclusions obtained from previous experiments on our training set of networks, we propose the following weighting function w similar to the one used in [5] to aggregate request processes of file i:

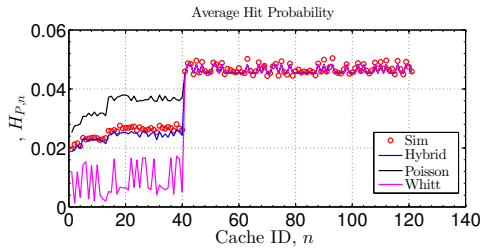
$$w_i = \frac{1}{1 + a\,(1 - O_{P,i})^b \times d_{eff}}, \quad O_{P,i} \approx \lambda_i \frac{C}{\Lambda}, \quad a = 1 \text{ and } b = 2, \tag{5.24}$$

We can easily check that if $O_{P,i} \to 1$ and d_{eff} fixed, then the weighting function $w_i \to 1$ and the Hybrid approximation behaves as the Whitt approximation. This case occurs mainly for popular files.

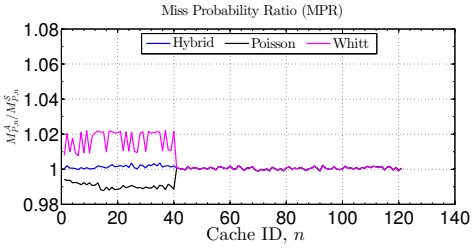
Meanwhile, as $d_{eff} \to \infty$ the weighting function $w_i \to 0$ and the Hybrid approximation converges to the Poisson approximation which was expected.



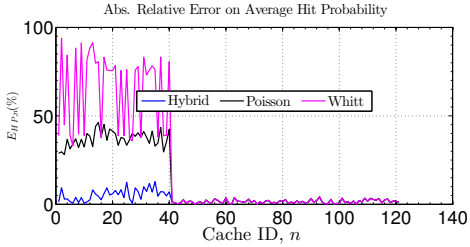
(a) Cache 9



(b) Average Hit Probability



(c) Miss Probability Ratio

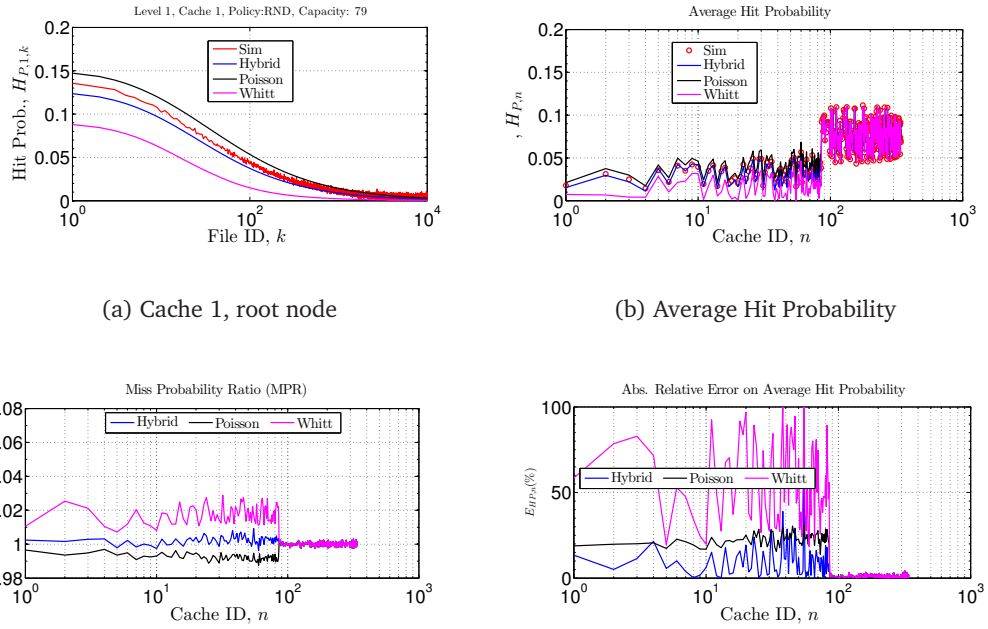


(d) Percentage of relative errors

Figure 5.23: Comparison of all approximations on a heterogeneous ternary tree of 121 caches, Depth 5, N $= 10^3$, C $= 10$.

We also added some corrections to accurately approximate the performance metrics of tandem networks in particular. Since we observed in Figures 5.15 and 5.17 that $w = 1$ is the appropriate value of the weighting function observed on linear networks of FIFO and RND caches, we decide to use this value for the latter tandem networks. However, we also noticed that $w = 1$ was a very loose value for tandem network of LRU caches (see Figure 5.19) in which case the values $w = 0.875$ and $w = 0.97$ of Table 5.2 should be used at the two successive nodes after the leaf; and $w = 1$ for the remaining nodes.

Implementing our model using the weighting function w in (5.24) and the previous corrections, we found that the Hybrid approximation predicts all performance metrics with the maximum relative errors on the per-cache average hit probability within 20–30%, the miss probability ratio close to one, and a good estimation of the per-file hit probabilities. We also observed that the expression in (5.24) is more robust and less sensible than those in (5.20) and (5.21), specially on heterogeneous tree networks (previously tested with values provided in Table 5.2) where caches may run different policies with different cache capacities.



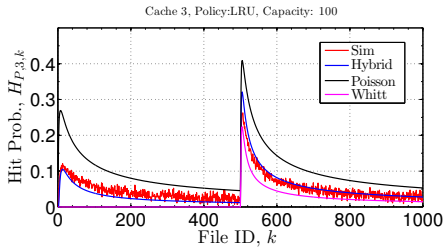
(a) Cache 1, root node

(b) Average Hit Probability

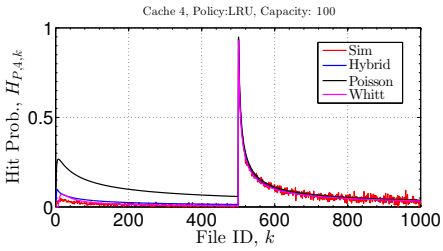
(c) Miss Probability Ratio

(d) Percentage of relative errors

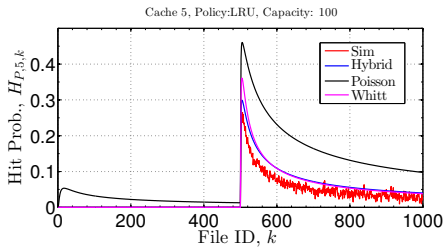
Figure 5.24: Comparison of all approximations on a heterogeneous ternary tree of 341 caches, Depth 5, $N = 10^4$, $C \in [50; 150]$.



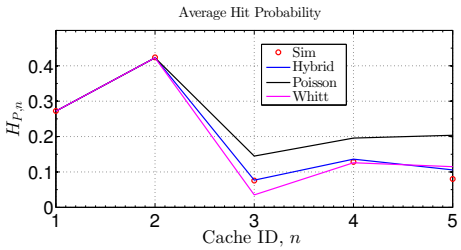
(a) Cache 3, core node



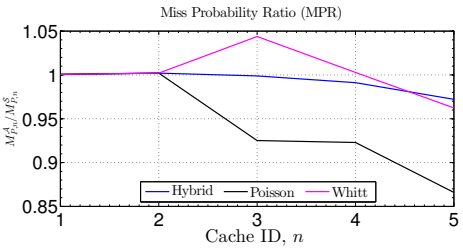
(b) Cache 4, root & edge node



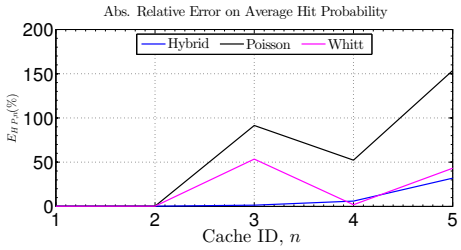
(c) Cache 5, root & core node



(d) Average Hit Probability



(e) Miss Probability Ratio



(f) Percentage of relative errors

Figure 5.25: Comparison of approximations on general network with five LRU caches, $N = 10^3$, $C = 10^2$.

5.8 Conclusion

In this chapter, we studied performance analysis of general and heterogeneous cache networks where caches are running LRU, FIFO and RND replacement policies under the assumption that request streams are described by hyper/shifted-exponential renewal processes. Our methodology is based on four independent and extensible building blocks that provide a step-by-step analysis of a cache within a network. These blocks translate the main network primitives into well-known operations on point processes such as superimposing, thinning, and filtering. Our approach is validated in the case of independent requests. Clearly the renewal assumption on all processes involved in the network is more accurate than Poisson assumption commonly used in the literature.

In all the tree network cases we tested, we observed that Poisson approximation is acceptable when the in-degree of a cache is large; meanwhile, Whitt approximation is more accurate for flat networks such as tandem of caches or planar graph cache network with nodes having small degree. Moreover, Whitt (resp. Poisson) approximation underestimates (resp. overestimates) our metrics of interest in general; while the Hybrid approximation we proposed can be very accurate has shown prediction error rates within 5–20% in our test set of cache networks.

However, this accuracy depends on a parameter called the weighting function which is needed to trade between Poisson and Whitt approximations. A complete characterization of the weighting function, a depth analysis of mesh (or non-tree) cache networks, and the implementation of our model for correlated requests are possible extensions of this work.

Finally, core networks can be easily designed using our models; meanwhile, users who take advantage of fast access and high speed provide by the backbone cache networks tend to stay almost always connected. This situation leads to an inefficient utilization of resources such as energy on the access networks. In the next chapter, we model the energy consumption in a wireless and cellular mobile networks.

6

Modeling Energy Consumption on Wireless Access Networks

6.1 Summary

In this chapter, we analyze the power save and its impact on web traffic performance when customers adopt the continuous connectivity paradigm. To this aim, we provide a model for packet transmission and cost. We model each mobile user's traffic with a realistic web traffic profile, and study the aggregate behavior of the users attached to a base station by means of a processor-shared queueing system. In particular, we evaluate user access delay, download time and expected economy of energy in the cell. Our study shows that dramatic energy save can be achieved by mobile devices and base stations, e.g., as much as 70%-90% of the energy cost in cells with realistic traffic load and the considered parameter settings.

Keyword 6.1 *Energy consumption models, power save mechanism, real web traffic, continuous connectivity, Queueing theory, sensitivity analysis, 3/4G access networks.*

6.2 Introduction

The total operating cost for a cellular network is of the order of tens of millions of dollars for a medium-small network with twenty thousand base stations [78]. A relevant portion of this cost is due to power consumption, which can be dramatically reduced by using efficient power save strategies. Power save can be achieved in cellular networks operating WiMAX, HSPA, or LTE protocols by optimizing the hardware, the coverage and the distribution of the signal, or

also by implementing energy-aware radio resource management mechanisms. In particular, we focus on power save in wireless transmissions, which would enable the deployment of compact (e.g., air conditioning free) and green (e.g., solar power operated) base stations, thus requiring less operational and management costs.

An interesting case study is offered by the behavioral analysis of users that remain online for long periods. These users request a continuous availability of a dedicated wideband data channel, in order to shorten the delay to access the network as soon as new packets have to be exchanged. This *continuous connectivity* requires frequent exchange of control packets, even when no data are awaiting for transmission. Therefore, in the case of continuous connectivity, a huge amount of energy might be spent just to control the high-speed connection, unless power save is enforced. However, since power save mode affects packet delay, some constraints have to be considered when turning to the power save operational mode.

Power save and sleep mode in cellular networks have been analytically and experimentally investigated in the literature, mainly from the user equipment (UE) viewpoint. E.g., power save in the UMTS UE has been evaluated in [97, 45] by means of a semi-Markov chain model. The authors of [87] propose an embedded Markov chain to model the system vacations in IEEE 802.16e, where the base station queue is seen as an M/GI/1/N system. The authors of [7] use an M/G/1 queue with repeated vacations to model an 802.16e-like sleep mode and to compute the service cost for a single user download. Using Laplace-Stieltjes Transform and Probability Generating Functions, [58] derives closed form expressions for the average power consumption (objective) and the average packet delay (constraint) for an UE. The authors of [58] also design a sleep mode mechanism based on traffic estimation and a solution of the optimization problem. Analytical models, supported by simulations, were proposed by Xiao for evaluating the performance of the UE in terms of energy consumption and access delay in both downlink and uplink [96]. Almhana *et al.* provide an adaptive algorithm that minimizes energy subject to QoS requirements for delay [6]. The works [11, 12] closely relate to our proposal and mainly focus on the analysis of the discontinuous reception mode in 3GPP LTE and IEEE 802.16m respectively. The authors consider both the uplink and downlink packets for the UE and show that uplink packets increase the power consumption and decrease the delay.

The existing work does not tackle the base station (or evolved node B, namely eNB) viewpoint nor analytically captures the relation between cell load and service rate statistics. Furthermore, for sake of tractability, many of those studies assume that packet arrivals follow a Poisson model. Instead, in real networks, the user traffic can be very bursty and follow long tail distributions [24]. In contrast, we use a G/G/1 queue with vacations to model the behavior of each UE, and we compose the behavior of multiple users into a single G/G/1 PS queue that models the eNB traffic. We analytically compute the cost reduction achievable thanks to power save mode operations, and show how to minimize the system cost under QoS constraints. In par-

ticular we refer to the mechanisms made available by 3GPP for *Continuous Packet Connectivity* (CPC), i.e., the discontinuous transmission (DTX) and discontinuous reception (DRX) [1].

The importance of DRX has been addressed in [98], where the authors model a procedure for adapting the DRX parameters based on the traffic demand, in LTE and UMTS, via a semi-Markov model for bursty packet data traffic. A description of DRX advantages in LTE from the user viewpoint is given in [16] by means of a simple cost model. In [63], the authors use heuristics and simulation to show the importance of DRX for the UE.

The contributions of our work are as follows: (i) we are the first to provide a complete model for the behavior of users (UEs) and base stations (eNBs) in continuous connectivity and with non-Poisson traffic (namely web traffic), (ii) we provide a cost model that incorporates the different causes of operational costs, (iii) we validate our model using packet-level simulations, (iv) we study the importance of a variety of model parameters by means of a *sensitivity analysis*, and (v) we show how to use the model to minimize operational costs under QoS constraints. Our results confirm that a tremendous cost reduction can be attained by correctly tuning the power save parameters. In particular, transmission costs can be lowered by more than 90% with realistic traffic loads.

The rest of the chapter is organized as follows: Section 6.3 presents power save operations in continuous connectivity mode. Section 6.4 describes a model for cellular users generating web traffic. Section 6.5 illustrates a model for downlink transmissions, and Section 6.6 describes how to evaluate flow performance and transmission costs. In Section 6.7 we validate the model through simulation. A sensitivity analysis is performed in Section 6.8, and a performance analysis and optimization is done in Section 6.9 showing the achievable power saving. Section 6.10 concludes the article.

6.3 Continuous connectivity

Cellular packet networks, in which the base station schedules the user activity, require the online UEs to check a control channel continuously, namely for T_{ln} seconds per system slot (i.e., per subframe T_{sub}). For instance, CPC has been defined by 3GPP for the next generation of high-speed mobile users, in which users register to the data packet service of their wireless operator and then remain online even when they do not transmit or receive any data for long periods [30]. A highly efficient power save mode operation is then strongly required, which would allow disabling both transmission and reception of frames during the idle periods. The UE, however, has to transmit and receive control frames at regular intervals, every few tens of milliseconds, so that synchronization with the base station and power control loop can be maintained. Therefore, idle periods are limited by the mandatory control activity that involves the UE. To save energy, when there is no traffic for the user, the UE can enter a power save

mode in which it checks and reports on the control channels according to a fixed pattern, i.e., only once every m time slots. Relevant energy economy can be achieved. However, the queued packets have to wait for the mth subframe before being served.

Discontinuous transmission. DTX has been first defined by 3GPP release 7. It is a UE operational mode for discontinuous uplink transmission over the Dedicated Physical Control Channel (DPCCH). With DTX, UEs transmit control information according to a cycle. There are actually two possible DTX cycles. The first cycle is short (a few subframes) and is used when some data activity is present in the uplink (normal operation). The second cycle is longer (up to tens of subframes) and is activated when an inactivity timer in the uplink data channel expires (power save mode operation). The threshold M for inactivity period is a power of 2 subframes (specified values are in $\{2^1, 2^2, \ldots, 2^9\}$).

Discontinuous reception. DRX is an operational mode defined by 3GPP release 6. It allows the UE to save energy while monitoring the control information transmitted by the eNB. DRX affects data delivery, since no data can be dependably received without an associated control frame. 3GPP specifications define a DRX cycle, that is the total number of subframes in a listening/sleeping window out of which only one subframe is used for control reception. Valid values for this cycle are 4 to 20 subframes (i.e., using a 2 ms subframe in HSPA yields cycles of 8 to 40 ms). DRX is activated only upon a timeout after the last downlink transmission, and like DTX, the timeout threshold M specified in the standard is a power of 2 subframes.

6.4 Power save model

We focus on the power consumption due to wireless activity on the air interfaces of mobile users (UEs) and base stations (eNBs). On one hand, we assume that uplink control transmission follows the DTX pattern. On the other hand, the UE has to decode the downlink control channel according to the DRX pattern, and receive packets accordingly [30]. Thus, uplink power save can be enabled by means of a long DTX cycle, with a timeout whose duration can be of the same order as the subframe size. Downlink power save is similarly enforced by setting the DRX cycle and timeout.

Thereby, power save issues in uplink and downlink can be modeled in a similar way, and there is little difference between the cost computation of a single UE and of a base station. Indeed, the overall cost at the eNB can be seen as the collection of costs over the control and data channels towards the various UEs, plus a fixed per-cell operational cost that the eNB has to pay to notify its presence and maintain the users synchronized. Therefore, here we focus on the downlink only, and begin our analysis with the behavior of a UE receiving a data stream.

Power save in downlink. As illustrated in Figure 6.1, downlink power savings can be obtained by alternating between two possible DRX cycles: after any downlink data activity there

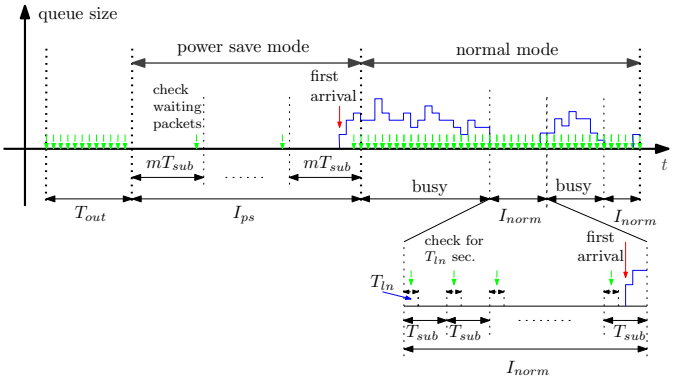


Figure 6.1: Downlink queue activity with power save and normal operation.

is a short cycle in which the UE checks the control channel at each subframe (normal operation mode); instead, upon the expiration of T_{out} (inactivity timer consisting of M subframes), there is a longer cycle in which the UE checks the control channel periodically, with period m subframes (power save mode).[1] In power save mode, the UE monitors the downlink control channel every m subframes, and returns to normal mode as soon as the channel sampling detects a control message indicating that the downlink queue is no longer empty. Note that UEs do not receive any service during: (i) I_{norm}, i.e., idle intervals in normal operation, (ii) timeout intervals, and (iii) I_{ps}, i.e., idle intervals in power save mode.

To quantify the power savings that can be achieved at the UE, in Section 6.5 we model the behavior of downlink transmissions with DRX operations enabled and users generating web traffic. Then, in Section 6.6 we discuss the tradeoff between per-packet performance and per-UE cost. Our model can be used for systems using slotted operations, and in particular LTE and HSPA [30]. The model can be applied to both uplink and downlink. However, for sake of clarity, we explicitly deal with the downlink case.

Achievable cost saving and performance metrics will be expressed as a function of the subframe length T_{sub} and the DRX parameters, namely the timeout duration, through the parameter M, and the DRX power save cycle duration, through the parameter m. We assume fixed-length packets, and the server capacity is exactly one packet per subframe. However, no packet is served when the UE is in power save mode, and the server capacity is shared, in each subframe, between the UEs operating in normal mode. Therefore, we model a system which behaves as a $G/G/1$ PS queue with repeated fixed-length vacations of mT_{sub} seconds.

Before proceeding with the model derivation, we introduce the traffic model adopted in this study.

[1] The actual system timeout is M-subframe long. However, since the UE checks for new traffic at the beginning of a subframe, the UE switches to power save mode if it does not receive any traffic alert at the beginning of the Mth idle subframe. Therefore, it is enough to have no arrivals for $M-1$ subframes and the UE will not receive any packet for M subframes.

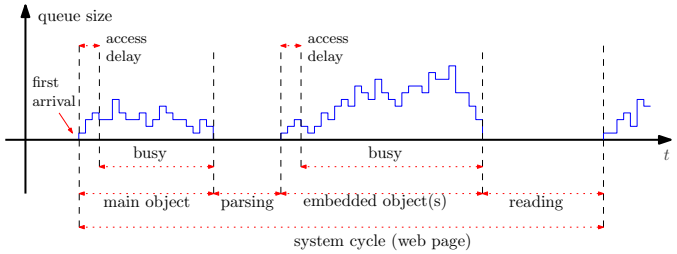


Figure 6.2: System cycle with web traffic as defined in [2].

Traffic model. We assume that downlink traffic is the composition of users' web browsing sessions. Traffic profiles are the same for all users and are as follows. The size of each web request is modeled as suggested by 3GPP2 in [2]: a web page consists of one main object, whose size is a random variable with truncated lognormal distribution, and zero or more embedded objects, each with random, truncated lognormal distributed size. The number of embedded objects is a random variable derived from a truncated Pareto distribution. Each web page request triggers downloads of the packets carrying the main object only. Then a *parsing time* is needed for the user application to parse the main object and request the embedded objects, if any. The parsing time distribution is exponential with rate λ_p. After having received the last packet of the last object, the customer *reads* the web page for an exponentially distributed *reading time*, whose rate is λ_r. If no object is embedded, the reading time includes the parsing time. Finally he/she requests another web page. Figure 6.2 represents the UE's downlink queue size at the eNB during a generic web page request and download. Table 6.1 summarizes the parameters used for the generation of web browsing sessions. Note that the probability ψ_0 to have no embedded objects in a web page can be computed through the distribution of the truncated Pareto random variable Y described in Table 6.1: $\psi_0 = P(y_{min} \leq Y < y_{min} + 1) = 1 - \left(\frac{y_{min}}{y_{min}+1}\right)^{\alpha}$. Note also that the downlink of the web page experiences a small access delay due to the completion of the current DRX cycle before the first packet of the new burst could be served.

In our model, we assume that the time to request a web object with a http GET command is negligible in comparison with the time needed to parse the main object, and therefore also in comparison with the time needed for a customer to read the web page. Hence we incorporate this request delay in the parsing time and in the reading time. In this way, we clearly focus our study on the sole impact of the wireless technology on the system performance and costs. Furthermore, packet arrivals are supposed to be bursty after each GET request, so that no power save mode can be triggered after an object download begins, i.e., all power save intervals are contained in either parsing or reading times. With these assumptions, we study the system performance through the analysis of a generic web page download and its fruition. More precisely, we study the system cycle defined as the time in between two consecutive web page requests.

Table 6.1: Parameters suggested by 3GPP2 for the evaluation of web traffic

Quantity	Probability distribution	Parameters
Main object size $S_{mo} = \lceil X \rceil$	$f_X(x) = \frac{\left(2\pi\sigma_X^2\right)^{-\frac{1}{2}} e^{-\frac{(\ln x - \mu_X)^2}{2\sigma_X^2}}}{\int_{x_{min}}^{x_{max}} \left(2\pi\sigma_X^2\right)^{-\frac{1}{2}} e^{-\frac{(\ln t - \mu_X)^2}{2\sigma_X^2}} dt}$, $x \in [x_{min}, x_{max}]$	$\mu_X = 8.35$ $\sigma_X = 1.37$ $x_{min} = 100$ bytes $x_{max} = 2 \cdot 10^6$ bytes
Number of embedded objects $N_{eo} = \lfloor Y \rfloor - y_{min}$	$f_Y(y) = \begin{cases} \alpha \frac{y_{min}^{\alpha}}{y^{\alpha+1}}, & y \in [y_{min}, y_{max}[\\ 1 - \left[\frac{y_{min}}{y_{max}}\right]^{\alpha}, & y = y_{max} \end{cases}$	$y_{min} = 2$ $y_{max} = 55$ $\alpha = 1.1$
Embedded object size $S_{eo} = \lceil Z \rceil$	$f_Z(z) = \frac{\left(2\pi\sigma_Z^2\right)^{-\frac{1}{2}} e^{-\frac{(\ln z - \mu_Z)^2}{2\sigma_Z^2}}}{\int_{z_{min}}^{z_{max}} \left(2\pi\sigma_Z^2\right)^{-\frac{1}{2}} e^{-\frac{(\ln t - \mu_Z)^2}{2\sigma_Z^2}} dt}$, $z \in [z_{min}, z_{max}]$	$\mu_Z = 6.17$ $\sigma_Z = 2.36$ $z_{min} = 50$ bytes $z_{max} = 2 \cdot 10^6$ bytes
Reading time Λ_r	$f_{\Lambda_r}(t) = \lambda_r e^{-\lambda_r t}$, $t \geq 0$	$\lambda_r = 0.0\overline{3}$ s
Parsing time Λ_p	$f_{\Lambda_p}(t) = \lambda_p e^{-\lambda_p t}$, $t \geq 0$	$\lambda_p = 7.69$ s

Therefore, the system cycle can be decomposed in four phases, as depicted in Figure 6.2: (i) download of the main object of the web page, (ii) parsing of the main object, (iii) download of embedded objects, and (iv) web page reading. The first three phases represent the web page download time, from the first packet arrival in the eNB queue to the last packet delivery to the UE. Access delay and download time characterize the service experienced by the customer.

6.5 Model derivation

Here we derive the time spent by the system in the various cycle phases. For ease of notation, we define $\beta_p = e^{-\lambda_p T_{sub}}$ and $\beta_r = e^{-\lambda_r T_{sub}}$ as the probabilities that, respectively, the exponentially distributed parsing time and reading time are longer than one subframe. Hence the timeout probability is β_r^{M-1} in reading time, and β_p^{M-1} in parsing time.

Timeouts in a cycle. Cycles always include one reading time, but the parsing time is present only if there are embedded objects (i.e., with probability $1 - \psi_0$). The average number of timeouts in a cycle is then:

$$E[N_{to}] = \beta_r^{M-1} + (1 - \psi_0)\, \beta_p^{M-1}. \tag{6.1}$$

Hence each cycle includes, on average, $E[N_{to}](M-1)T_{sub}$ seconds due to timeout occurrences.

Idle time in power save mode. The average time per cycle during which the system is in power save mode, denoted as I_0, is computed by summing up the time spent in power save mode (the intervals I_{ps} as in Figure 6.1) occurring in the reading time and in the parsing time,

if any is present in the cycle: $I_0 = I_{ps|reading} + I_{ps|parsing}$. Thanks to the memoryless property of exponential arrivals, the interval between the timeout expiration and the arrival of the next data packet is exponential too, and has the same exponential rate. In particular, the power save interval that begins in the reading time lasts a multiple number of checking intervals mT_{sub}, with the following distribution and average:

$$\begin{aligned}
&P\Big(I_0 = jmT_{sub}\Big|\text{ reading, timeout}\Big)\\
&\quad = P\Big(0 \text{ arrivals in } (j-1)mT_{sub}\Big)\Big[1 - P\Big(0 \text{ arrivals in } mT_{sub}\Big)\Big]\\
&\quad = (\beta_r^m)^{j-1}(1-\beta_r^m), \qquad j \geq 1;\\
&E[I_0|\text{reading}] = \beta_r^{M-1}\frac{mT_{sub}}{1-\beta_r^m};
\end{aligned}$$

where we also removed the conditioning on the timeout occurrence. Similarly, for the parsing time:

$$E[I_0|\text{parsing}] = \beta_p^{M-1}\frac{mT_{sub}}{1-\beta_p^m}.$$

Therefore, the expected time spent in power save mode in a system cycle is given by the following average:

$$E[I_0] = \beta_r^{M-1}\frac{mT_{sub}}{1-\beta_r^m} + (1-\psi_0)\,\beta_p^{M-1}\frac{mT_{sub}}{1-\beta_p^m}. \tag{6.2}$$

Note that $E[I_0]$ is a function of m and M, the web traffic parameters being fixed. It is easy to show that $\frac{\partial}{\partial m}E[I_0] > 0$, and $\frac{\partial}{\partial M}E[I_0] < 0$, hence the power save interval I_0 monotonically grows with the duration of the DRX cycle, and decreases with the duration of the timeout.

Idle time in normal mode. The amount of time spent in normal mode without serving any traffic is the sum of the normal mode idle intervals due to parsing and reading times. Since (6.1) accounts the time spent in timeouts, here we only count the intervals I_{norm}, whose sum over a system cycle is denoted by $I_1 = I_{norm|reading} + I_{norm|parsing}$. Considering that I_{norm} is always a multiple of T_{sub} but smaller than a timeout, and since the component of I_1 in reading time is $I_{norm|reading}$, the conditional distribution of I_1 in reading time and its expectation are as follows:

$$\begin{aligned}
P\big(I_1 = jT_{sub}\big|\text{reading}\big) &= P\Big(I_{norm} = jT_{sub}\Big|\text{ exp. arrivals with rate } \lambda_r\Big)\\
&= \begin{cases} \beta_r^{M-1}, & j = 0;\\ \beta_r^{j-1}(1-\beta_r), & 1 \leq j \leq M-1; \end{cases}\\
E[I_1|\text{reading}] &= T_{sub}\frac{1 - M\beta_r^{M-1} + (M-1)\beta_r^M}{1-\beta_r}.
\end{aligned} \tag{6.3}$$

Similarly, the expected value of the time spent in normal mode with no traffic to be served during parsing, not counting the timeout, is given by

$$E[I_1|\text{parsing}] = T_{sub}\frac{1 - M\beta_p^{M-1} + (M-1)\beta_p^M}{1-\beta_p}. \tag{6.4}$$

Therefore, the average duration of I_1, attained by using (6.3) and (6.4), is an increasing function of the timeout duration, as expressed by the following formula:

$$E[I_1] = T_{sub}\left[\frac{1-M\beta_r^{M-1} + (M-1)\beta_r^M}{1-\beta_r} + (1-\psi_0)\frac{1-M\beta_p^{M-1} + (M-1)\beta_p^M}{1-\beta_p}\right]. \tag{6.5}$$

Cumulative idle time. The cumulative amount of idle time I in a cycle is the sum of timeouts, I_0, and I_1. Its expected value is then as follows:

$$E[I] = \frac{\beta_r^{M-1} m T_{sub}}{1-\beta_r^m} + T_{sub}\frac{1-\beta_r^{M-1}}{1-\beta_r} + (1-\psi_0)\left[\frac{\beta_p^{M-1} m T_{sub}}{1-\beta_p^m} + T_{sub}\frac{1-\beta_p^{M-1}}{1-\beta_p}\right]. \tag{6.6}$$

$E[I]$ is a decreasing function of M, and increases with m. However, with our model assumptions, $E[I]$ is slightly larger than the sum of reading and parsing times. More precisely, its value is bounded as follows:

$$\frac{1}{\lambda_r} + \frac{1-\psi_0}{\lambda_p} < E[I] < \frac{1}{\lambda_r} + mT_{sub} + (1-\psi_0)\left(\frac{1}{\lambda_p} + mT_{sub}\right). \tag{6.7}$$

Given that m can be as large as several tens of milliseconds, and T_{sub} is only few milliseconds, the product mT_{sub} is negligible in comparison with the average parsing and reading times. Hence, for all realistic values of m, the per-cycle idle time can be considered constant and equal to its lower bound.

Busy time in a cycle. This is the time spent to serve the packets of a web page. Its expectation is the expected number of packets per web page, $E[N_p]$, times the expected packet service time $E[\sigma]$. The number of packets depends on the distribution of the web page objects. Assuming the 3GPP2 traffic model reported in Table 6.1 and 1500-byte long packets, we can compute:

$$E[N_p] = E\left[\left\lceil\frac{S_{mo}}{1500}\right\rceil\right] + E[N_{eo}]\cdot E\left[\left\lceil\frac{S_{eo}}{1500}\right\rceil\right] = 39.476.$$

The service time depends on the number of active UEs and on the server capacity, as we show later in this section.

System cycle duration. Putting together the results for the time spent in timeouts, idle intervals, and busy periods, the expected cycle duration is:

$$E[T_c] = E[I] + E[N_p]E[\sigma]. \tag{6.8}$$

The relation between $E[T_c]$ and $E[\sigma]$ is linear with a coefficient that is determined by the web page object distribution. Since $E[\sigma]$ too will be shown to grow with m and decrease with M (see next paragraph), the entire expected system cycle increases with m and decreases with M. Furthermore, as the expected service time increases with the number N_u of UEs attached to the eNB, the system cycle behaves likewise. However, both $E[I]$ and $E[\sigma]$ are barely affected by m and M, thereby $E[T_c]$ is mainly affected by N_u.

Service time. We assume that there are N_u homogeneous UEs in the cell. The activity factor of each UE is:

$$\rho = \frac{E[N_p]E[\sigma]}{E[T_c]} = \frac{E[N_p]E[\sigma]}{E[I] + E[N_p]E[\sigma]} < 1. \tag{6.9}$$

Equivalently, we can interpret ρ as the probability that a UE is under service. Note that $E[\sigma]$, $E[N_p]$, and $E[I]$ assume always positive values, and thus $E[T_c] > 0$ and $0 < \rho < 1$.

From the point of view of a generic queue, the service time in the lth subframe only depends on the number $N_a(l)$ of queues which transmit in that specific subframe. In fact, the downlink bandwidth is shared between all backlogged active queues, the total serving capacity being fixed to one packet per subframe. Thus, given that the ith queue has a packet under service in the lth system subframe, the service time for the ith queue is $T_{sub}N_a(l)$. Since we are interested in the service time for the ith queue, we condition the observation of the service time to the transmission of a packet queued in the ith queue. Hence, considering all queues as i.i.d., the number of active queues is a random variable $N_a = 1 + \nu$, with ν being a random variable exhibiting a binomial distribution between 0 and $N_u - 1$ with success probability ρ. Thereby, the average service time is:

$$E[\sigma] = T_{sub}E[1 + \nu] = T_{sub}[1 + (N_u - 1)\rho]. \tag{6.10}$$

Hence, considering the expression (6.9) of ρ as a function of $E[\sigma]$, we have a system of two equations in two variables, from which we can compute $E[\sigma]$.

Proposition 6.1 *The expected packet service time* $E[\sigma]$ *is the unique positive solution of the following quadratic equation:*

$$E[N_p]\,E^2[\sigma] + (E[I] - E[N_p]\,N_u T_{sub})\,E[\sigma] - E[I]T_{sub} = 0.$$

Proof *The equation is obtained by combining* (6.9) *and* (6.10). *Since* $E[N_p]$ *and* $E[I]$ *are positive numbers, the quadratic coefficient in the equation is always positive, whilst the constant term is negative: this is necessary and sufficient to have one positive solution and one negative solution. However, the negative solution has no physical meaning. Thus, the positive solution is the only acceptable solution candidate.* ∎

Corollary 6.1 *The expected packet service time is*

$$E[\sigma] = \frac{(E[N_p]N_u T_{sub} - E[I]) + \sqrt{(E[I] - E[N_p]\,N_u T_{sub})^2 + 4E[I]\,E[N_p]\,T_{sub}}}{2E[N_p]}.$$

As we stressed before, the term $E[I]$ increases with m and decreases with M, but its variations are quite limited. So, thanks to the Corollary, we conclude that $E[\sigma]$ behaves as $E[I]$, i.e., it is barely affected by m and M. Furthermore, $E[\sigma]$ grows with N_u, i.e., with the number of UEs in the cell. Notably, the impact of N_u on $E[\sigma]$ is amplified by a factor equal to the average page size $E[N_p]$.

Since a new web page is requested only after the reading time of the previous request, the number of customers has no theoretical upper bound. In fact, service time and system cycle just keep growing with the number of UEs, and the average cumulative traffic generated and served per subframe is $N_u \frac{E[N_p]}{E[N_p]\,E[\sigma]+E[I]} \leq \frac{1}{T_{sub}}$. Thus, as the system approaches saturation, $E[\sigma]$ tends to $N_u T_{sub}$, since in saturation the N_u users are always active and receive a fraction $1/N_u$ of the server capacity. The asymptotic distribution of the system cycle duration is constant and equal to $T_c^{up} = E[N_p]\,N_u T_{sub} + E[I]$, which scales linearly with the number of users and loosely depends on the power save parameters m and M. T_c^{up} is an upper bound on $E[T_c]$, and can be used to limit the maximum number of customers, thus guaranteeing a maximum web page processing time to any customer.

6.6 Performance and cost metrics

The impact of power save mode on web traffic can be evaluated in terms of access delay and page download time, assuming that all the traffic is served. Costs due to wireless transmission and reception of packets are to be traded off with such indicators. Therefore, we first derive an expression for performance metrics and show how to compute the fraction of time during which power save can be realistically obtained. Then we derive the parametric expressions for cost and power save at both UE and eNB.

6.6.1 Performance metrics and power save opportunities

Page download time. The time W needed to download a web page includes the time to download each and every page's packet, the time to parse the main object of the page, and the access delay. Hence, we can derive $E[W]$ as the difference between $E[T_c]$ and the expected reading time:

$$E[W] = E[T_c] - \frac{1}{\lambda_r}. \tag{6.11}$$

Considering (6.7) and (6.8), a tight lower bound on the expected page download time is $E[N_p]E[\sigma] + (1 - \psi_0)/\lambda_p$.

Access delay. The access delay is the delay experienced after any download request. In our model we consider only that part of the access delay which is due to the wireless access protocol. In particular, we have two epochs within each cycle at which a request can experience access delay: at the end of the reading time, corresponding to a new page request, and at the end of the parsing time, corresponding to the request for the embedded objects. Let D be the total access delay experienced within a web page download, accounting for the delay accumulated in both reading and parsing times. $E[D]$ can be easily computed by subtracting parsing, reading, and busy times from the expected system cycle duration (see Figure 6.2), i.e.:

$$E[D] = E[I] - \left(\frac{1}{\lambda_r} + \frac{1-\psi_0}{\lambda_p}\right). \tag{6.12}$$

The expected access delay is a function of the power save parameters used in the DRX configuration, plus the traffic profile parameters, through λ_r, λ_p, $E[N_p]$, and ψ_0. However, using the upper bound for $E[I]$, one can conclude that the access delay is upper bounded by $(2-\psi_0)mT_{sub}$.

Power save time ratio. Energy savings can be achieved by reducing the radio activity, including the possibly turning off the radio transceiver, according to the DTX/DRX pattern. Therefore, power save opportunities can be represented by the fraction of the cycle during which the transceiver can be deactivated. In practice, UE and eNB can save power during I_0, which is a multiple of mT_{sub} during which no transmissions occur. However, in the interval I_0, the UE has to periodically be active to listen to the control channel for exactly $T_{ln} \leq T_{sub}$ seconds out of m subframes. The power save time ratio is then defined as follows:

$$R \triangleq \left(1 - \frac{T_{ln}}{mT_{sub}}\right)\frac{E[I_0]}{E[T_c]}. \tag{6.13}$$

Recall that $E[T_c]$ is relatively insensitive to m and M, but increases with N_u, and that $E[I_0]$ increases with m and decreases with M. Therefore, R is an increasing function of m, and it decreases with M and N_u.

6.6.2 Cost analysis

Cost at the UE. The basic consumption rate of energy at the UE receiver is c_{on} if active and $c_{ps} < c_{on}$ otherwise. Receiving a packet *increases* the basic consumption rate by c_{rx}, while listening to the control channel *increases* it by c_{ln}. The average consumption rate is a combination of these four consumption rates. For sake of generality we assume that listening to the control channel can last different amount of time, depending on whether data are associated with the control message or not. For instance, in HSPA systems, the user can switch from the control to data channel after having decoded the initial part (one third) of the control frame indicating the arrival of a new data frame [30]. We denote by T_{ln} the listening time when no data are transmitted, and by T'_{ln} the listening time when data follow the control message.

Therefore, using definitions (6.9) and (6.13), and recalling that control channel listening is performed in each subframe in normal mode, but only in one out of m subframes in power save mode, we can compute the cost per UE by taking the average over a system cycle while keeping separated the listening occurrences with and without associated data transmissions. Namely,

$$C_{UE}(m, M, N_u) = (1 - R)c_{on} + Rc_{ps} + \rho c_{rx} + \left[\frac{\left[1 - \rho - \frac{m-1}{m}\frac{E[I_0]}{E[T_c]}\right] T_{ln}}{T_{sub}} + \frac{\rho T'_{ln}}{T_{sub}} \right] c_{ln}. \tag{6.14}$$

Considering a fixed web traffic profile, the cost is a function of the power save parameters m and M affecting R, ρ, $E[I_0]$, and $E[T_c]$, and of the number of users N_u which appears in $E[T_c]$ and hence in R. The cost with no power save mode is computed by plugging $E[I_0] = 0$, which is equivalent to setting $m = 1$ and $M \to \infty$, in (6.14):

$$C_{UE}(1, \infty, N_u) = c_{on} + \rho\, c_{rx} + \left[(1 - \rho)\frac{T_{ln}}{T_{sub}} + \rho\frac{T'_{ln}}{T_{sub}} \right] c_{ln}. \tag{6.15}$$

Finally, the relative power save gain that can be attained is:

$$G_{UE}(m, M, N_u) \triangleq \frac{C_{UE}(1, \infty, N_u) - C_{UE}(m, M, N_u)}{C_{UE}(1, \infty, N_u)} = \frac{\gamma(m) E[I_0]/E[T_c]}{C_{UE}(1, \infty, N_u)}, \tag{6.16}$$

where the quantity $\gamma(m)$ is a cost reduction factor which increases with the DRX power save cycle length m, namely:

$$\gamma(m) \triangleq \left(1 - \frac{T_{ln}}{mT_{sub}}\right)(c_{on} - c_{ps}) + \left(1 - \frac{1}{m}\right)\frac{T_{ln}}{T_{sub}} c_{ln}. \tag{6.17}$$

Note that T'_{ln} does not affect the cost reduction (numerator of (6.16)).

Summarizing, the relative gain is a function that increases with the duration of the DRX power save cycle (i.e., with m), and decreases with the timeout (i.e., with M) and with the number N_u of users in the cell.

Cost at the eNB. The power consumption rate at the eNB is the sum of a fixed component, c_f, that does not depend on the transceiver activity, and a variable component that depends on the activity of UEs in the cell. Namely, the power consumption rate at the eNB is

$$C_{BS}(m, M, N_u) = c_f + N_u C'_{UE}(m, M, N_u). \tag{6.18}$$

where $C'_{UE}(m, M, N_u)$ is the cost per time unit to transmit to a single UE. It can be written as follows:

$$C'_{UE}(m, M, N_u) = C'_{UE}(1, \infty, N_u) - \gamma'(m) \cdot \frac{E[I_0]}{E[T_c]}, \tag{6.19}$$

with

$$C'_{UE}(1, \infty, N_u) = c_{on} + \rho\, c_{tx} + \frac{T'_{ln}}{T_{sub}} c_{sg}; \tag{6.20}$$

$$\gamma'(m) = \left(1 - \frac{T'_{ln}}{mT_{sub}}\right)(c_{on} - c_{ps}) + \left(1 - \frac{1}{m}\right)\frac{T'_{ln}}{T_{sub}} c_{sg}. \tag{6.21}$$

Here, c_{tx} is a transmission cost rate and c_{sg} is a signaling cost. Last, the relative power save gain is:

$$G_{BS}(m, M, N_u) = \frac{\gamma'(m)}{C'_{UE}(1, \infty, N_u) + \frac{c_f}{N_u}} \cdot \frac{E[I_0]}{E[T_c]}. \tag{6.22}$$

Note that with few users the main eNB cost is represented by the fixed cost c_f. Hence, the gain increases with the number of users until the per-user cost becomes the predominant term in the denominator of (6.22).

6.7 Validation through simulations

In this section we evaluate the robustness of the model by comparing the analytical results to simulations. The main assumption used in the model states that queues related to different active UEs are i.i.d.; however, queues are correlated in practice as they share the same processor. This assumption is not met in the simulations.

We developed a C++ packet-level event-driven simulator that reproduces the behavior of a time slotted G/G/1 PS queue with N_u homogeneous classes. In the simulator, each class can be in two different operational modes, namely normal mode and power save mode. The shared processor resources are allocated equally to all classes in normal mode at the beginning of each time slot of duration T_{sub}. The traffic is homogeneously generated, in accordance to the 3GPP2 suggested web traffic model of Table 6.1. Furthermore, all simulated packets have the same size, i.e., 1500 bytes, and the processor capacity is 1500 bytes per slot. Hence, if only one class is under service, a packet is served completely in one slot. Otherwise, since the processor is shared, all classes in normal mode have a fraction of packet served in that slot. The fair per-class share is computed as one over the number of classes in normal mode. If a class has not enough backlog to use all its processor share, unused resources are redistributed among the remaining classes. Packet service is considered complete at the end of its last service slot.

Simulations are performed for different numbers of classes N_u, duration of the timeout M, and length of DRX power save cycle m. Hereafter, we will use $\lambda_r = 1/30$ s, $\lambda_p = 1/0.13$, $\psi_0 = 1 - (2/3)^{1.1}$, and $T_{sub} = 2$ ms. Each simulation consists of a warm-up period lasting 10,000 seconds (5,000,000 slots), followed by 100 runs, each lasting 10,000 seconds. Statistics are separately collected in each run. At the end of a simulation, all statistics are averaged over the 100 runs and 99% confidence intervals are computed for each average result.

We need to run simulations for such a long time to have statistics with relatively small confidence intervals. In fact, due to heavy tailed distributions involved in the generation of web traffic, the number of packets per cycle has a huge variance. Furthermore, simulations with a high number of users require very long CPU time (in our specific case, a single simulation point requires up to 12 hours of a 3 GHz Intel Core^{TM}2 Duo E6850 CPU), which makes it prohibitive

to explore in detail all possible values of the input parameters. As a reference, our model can be run with the Maple software in as few as 30 seconds on the same machine used for simulations.

The model, however, neglects the correlation between the activity of different users, e.g., in the computation of $E[\sigma]$. Nevertheless, the comparison between model and simulation shows that the model approximates the system performance with a good accuracy. Numerical results for $E[\sigma]$, obtained from both the model and the simulations, are reported in Figure 6.3. It is clear from the figure that the model slightly overestimates the service time for high values of

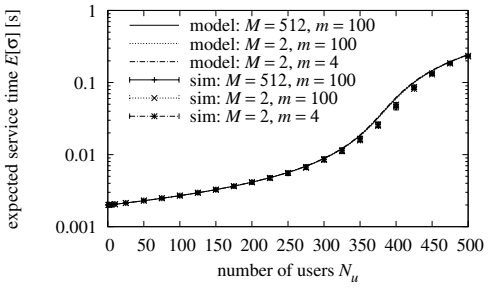
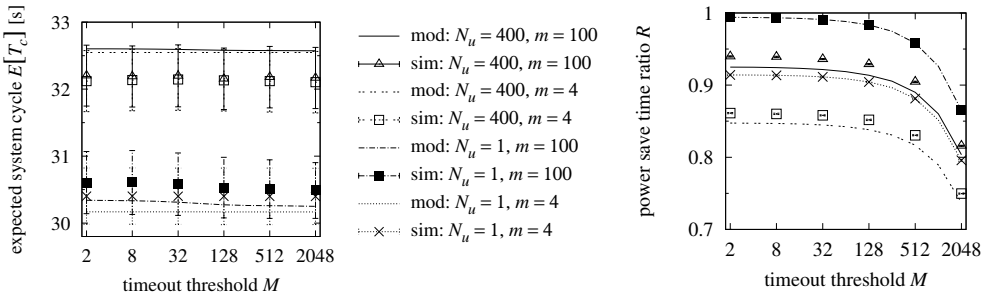


Figure 6.3: $E[\sigma]$ grows with N_u and is almost not affected by the timeout and the DRX power save cycle durations.

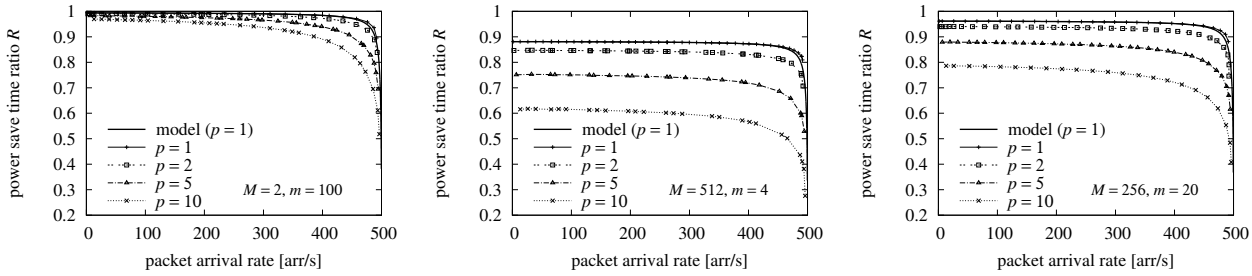
N_u, i.e., when the correlation between multiple users, in terms of probability to share the same transmission slot, becomes relevant. As predicted, m and M do not significantly affect $E[\sigma]$.

We now compare two performance metrics: the system cycle duration $E[T_c]$ and the power save time ratio R. $E[W]$ can be easily computed from $E[T_c]$; cf. (6.11). For clarity of presentation, we show only a subset of the results obtained. In particular we selected some extreme cases that well depict the variability of performance with m, M, and N_u.



(a) Expected system cycle duration $E[T_c]$ (b) Power save time ratio R

Figure 6.4: Comparison of analytic and simulation results: (a) $E[T_c]$, and (b) R.



(a) high save configuration (b) low save configuration (c) recommended setup

Figure 6.5: Impact of the number p of parallel user's browsing sessions on R, for $T_{ln} = \frac{T_{sub}}{3}$.

Figure 6.4.a compares the estimates of $E[T_c]$ obtained with the model (lines) and with the simulator (marked points) for two very different values of m (4, which is the minimum in the 3GPP recommendations, and 100). The lower part of the figure contains the results obtained with one user, and the upper part reports the results with $N_u = 400$ users. The results of the simulation are highly variable due to the heavy tailed distribution in web page size statistics, hence 99%-confidence intervals appear large over the zoomed y-scale used in the figure. Though the average values show some small difference, both simulations and model behave similarly. The maximum relative difference between model and simulation with one user is within 1%, and it is below 2% with $N_u = 400$. Noticeably, model estimates are within the 99%-confidence intervals of simulation estimates.

The main cause of the difference between the results of the model and the ones obtained via simulation is in the estimation of the service time, which linearly affects the cycle duration. Similar differences can be observed for the power save time ratio R with $N_u = 400$ in Figure 6.4.b. Analytic and simulation results remain however very close. The results are sensitive to m and N_u, while the effect of M is almost negligible for short timeouts.

In conclusion, simulations suggest that we can safely use the model to estimate the system performance and evaluate its potentialities for power save with good accuracy.

6.7.1 Impact of parallel user's browsing sessions

In real life, a user can activate more than one browsing window and switch from window to window while a page is being loaded. Thus, in practice it is not uncommon to have more than one browsing session active on the same device. Therefore, in that case, the arrival process at the user's download queue will result from the superposition of various per-browsing session arrival processes. Here we simulate the occurrence of multiple active http browsing sessions for each user, and we compare the performance with the case of single browsing session. Our

model does not capture the effect of parallel http sessions, hence the experiments proposed in this subsection are aimed at evaluating whether our study can be used to approximate the network behavior in more generic and realistic traffic scenarios. Specifically, we focus on one particular metric, namely the power save time ratio, since it is representative of the system's power save performance.

In Figure 6.5, we plot the power saving time ratio R for three scenarios: a configuration for the DRX parameters (M, m) yielding high power savings (Figure 6.5.a), a configuration yielding the minimum power saving for realistic values of (M, m) (Figure 6.5.b), and the configuration that we recommend in light of our optimization analysis reported in Section 6.9, i.e., $(M, m) = (256, 4)$ (Figure 6.5.c). The recommended configuration, yields a good tradeoff between power save and serving delay incurred by the packets due to DRX operations.

Note that, since a user generates a traffic volume which depends on the number p of parallel http browsing sessions, in Figure 6.5 we plot R as a function of the offered load, expressed in terms of packet arrivals per second. For each represented curve, we change the load by changing the number of users N_u, and report the corresponding arrival rate in the x-axis, and the power save time ratio R in the y-axis. As reference, we include in each figure the results obtained with the model by increasing N_u from 1 to 1000, then computing $E[\sigma]$ by solving the system consisting of Eqs. (6.9) and (6.10), or with the formula given in the Corollary in Section 6.5, for each given value of N_u, and eventually computing the load factor as $N_u E[N_p] T_{sub}/E[T_c]$. The latter formula represents the fraction of time spent in a cycle to serve the average aggregate volume of downlink packets $N_u E[N_p]$ generated in that cycle, when the volume of data corresponding to one packet is served in exactly one subframe T_{sub}.[2]

In the model, the load in packet arrivals per second is computed by scaling the load factor by $\frac{1}{T_{sub}}$, which is the maximum number of packets that can be served in a second, and is 500 in our case, corresponding to the capacity of a HSPA downlink with 2ms subframes. Clearly, a given arrival rate corresponds to a different number of users N_u when p changes, and the relation between the packet arrival rate, the number of browsing sessions p, and the number of users N_u cannot be predicted with our model. Therefore, for $p > 1$ we only show simulation results.

Observing Figure 6.5, one can notice that (i) the model accurately predicts the simulation for $p = 1$, and (ii) values of p as large as 10 can have a remarkable impact on the power save time ratio R. However the impact of p is important only for high loads and for large values of the DRX timeout M, causing up to a $\sim$ 25% drop in power save opportunities. However, for reasonable values of p, e.g., 2 to 5, R remains always very large and within a few percent

[2]Equivalently, the load factor can be computed as the sum of N_u activity factors expressed as in Eq.(6.9), multiplied by a coefficient $T_{sub}/E[\sigma]$ which represents the fraction of resources allotted to a user when sharing the processor with other users.

from the value achieved with $p = 1$. In light of this result, we argue that using our model can suitably approximate the computation of the power save opportunities of a system with users browsing a few (up to 5) web pages in parallel.

We will now perform a sensitivity analysis on our model to evaluate which parameters mostly affect the performance metrics.

6.8 Sensitivity analysis

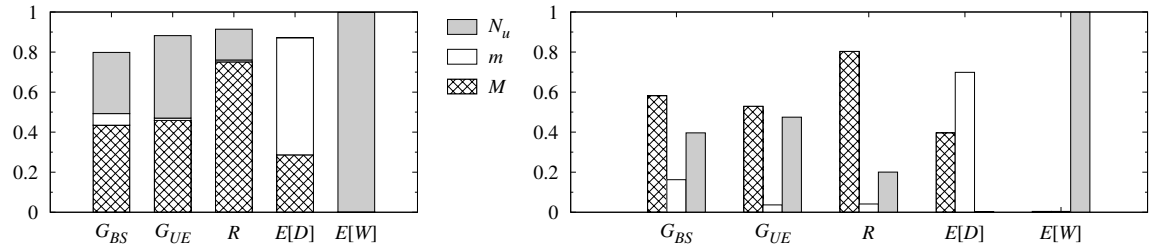
In the previous sections, we provided the expressions for the performance and cost metrics that enables us, by a partial derivation, to outline a preliminary behavior of our metrics according to the input parameters, namely M, m and N_u. Our objective now is to characterize qualitatively and quantitatively the impact of our input parameters on the variability of our metrics. We will further analyze the sensitivity of the metrics when the expected web page size $E[N_p]$, the expected reading rate λ_r, and the expected parsing rate λ_r are uncertain, in addition to the three input parameters.

Performing a *sensitivity analysis* allows us to understand how variability in the output of a model can be apportioned to different input parameters. Variance-based techniques define sensitivity indices (i) to measure the main effect of a given input on the output, (ii) to measure the relative importance of any combination of input in the output variability, and (iii) to measure the total effect of a given input on the output. More precisely, assuming the inputs to be random variables $X_1, \ldots, X_n$, and the model output to be a random variable $Y = f(X_1, \ldots, X_n)$, the first order and total sensitivity indices for random variable X_i are defined as follows:

$$S_i = \frac{\mathrm{Var}\Big(E[Y|X_i]\Big)}{\mathrm{Var}(Y)}, \qquad S_i^T = \frac{E\Big[\mathrm{Var}(Y|X_{-i})\Big]}{\mathrm{Var}(Y)}, \tag{6.23}$$

where X_{-i} denotes all input random variables except X_i. S_i is a quantitative measure of the main effect of X_i on output Y (through its variance) and S_i^T is a quantitative measure of the total effect of X_i, including the interactions with other input random variables. The difference $S_i^T - S_i$ measures the importance of interactions in the total effect of X_i. When there are no interactions between the input random variables, the sum $\sum_{i=1}^{n} S_i = 1$; otherwise, this sum is less than 1. If S_i^T is small, then this means that the value of X_i is not essential, and it can be considered as deterministic, taking any value within its range, without any significant impact on the model output. Note that an exhaustive sensitivity analysis requires the computation of $2^n - 1$ sensitivity indices, including those accounting for interactions between any combination of input random variables. The sum of these $2^n - 1$ indices amounts to 1.

One method for estimating S_i and S_i^T for non-correlated variables is the Extended Fourier Amplitude Sensitivity Test (EFAST), introduced by Saltelli et al. in 1999 [85]. The EFAST



(a) First order sensitivity indices S_i

(b) Total order sensitivity indices S_i^T

Figure 6.6: Sensitivity indices of M, m and N_u for the defined metrics.

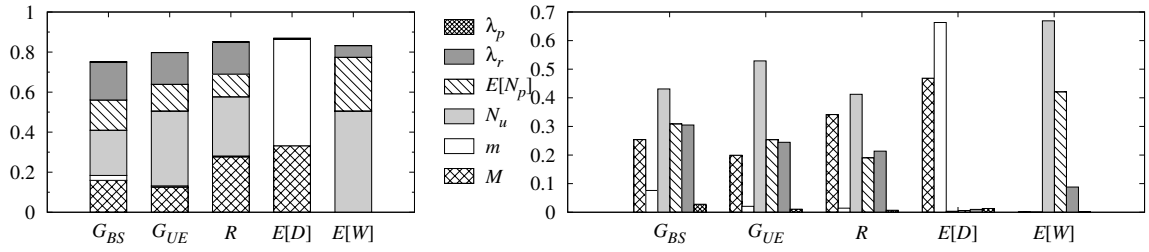
method does not require any knowledge on the function $f(.)$, which can be seen as a black box. The advantages of EFAST are its robustness, especially for small sample sizes, and its computational efficiency. EFAST expands the output of the model by using the Fourier Series, then assigns an integer frequency to each input parameter, to finally compute the variance of output as well as the contribution of each input to this variance. Using a brute-force approach, computing S_i and S_i^T requires the evaluation of multidimensional variance integral. The main advantage of EFAST is to reduce the computation of this complex integral to a monodimensional integral over a curve exploring the n-dimensional space. For a detailed description of the method we refer the reader to [85, 29].

We will now show the results of the sensitivity analysis (SA) for the five performance and cost metrics introduced in Sections 6.5 and 6.6. We have checked our results using two different software packages that implement SA.

6.8.1 SA results with three input parameters: M, m and N_u

We first apply the EFAST method to our model with the web traffic configuration specified by 3GPP2 (see Table 6.1). We consider the following ranges for the three input parameters: $M \in \{2, 2^2, \ldots, 2^{15}\}$; $m \in \{1, \ldots, 50\}$ and $N_u \in \{1, \ldots, 600\}$. All other parameters are constant in this analysis. We compute the first order (S_i) and the total (S_i^T) sensitivity indices of each of the parameters M, m and N_u for the five performance and cost metrics defined in Section 6.6. The results are displayed in Figure 6.6. It is clear that parameters with a small impact on some metric may well be essential for other metrics. We can make the following observations:

- The download time $E[W]$ is affected only by the number of cell users N_u; the DRX parameters M and m may take any value within their range without impacting $E[W]$.
- The cycle length m is essential for the access delay $E[D]$ and has a minor effect on the eNB's relative gain G_{BS}. Noticeably, about two thirds of the total effect of m on G_{BS}



(a) First order sensitivity indices S_i

(b) Total order sensitivity indices S_i^T

Figure 6.7: Sensitivity indices of $M, m, N_u, \lambda_r, \lambda_p$ and $E[N_p]$ for the defined metrics.

comes from interactions with other variables.

- The timeout threshold M is the most relevant parameter concerning the power save time ratio R and the gains G_{UE} and G_{BS}. The second most relevant input parameter affecting these metrics is N_u.
- Last, interactions between multiple variables are mostly relevant for the gain G_{BS}.

6.8.2 SA results with six input parameters: $M, m, N_u, \lambda_r, \lambda_p$ and $E[N_p]$

As the Internet (and so the web) is evolving very fast, it is easy to predict that the traffic parameters suggested by 3GPP2 (see Table 6.1) will have to be modified. Therefore, power save performance will change accordingly, and network optimization will require a different setup. In particular, the actual trend for mobile devices is to increase memory and data processing speed; meanwhile, web page sizes tend to increase because of the embedded objects, some of which are large images/videos or heavy scripts. Furthermore, some websites offer light versions of web pages specifically for mobile clients. To provide insights on the relevance of these changes with respect to our metrics of interest, we now present the results of our sensitivity analysis extended to the model parameters that characterize the user traffic behavior, namely the reading and parsing time, through λ_r and λ_p, and the web page average size $E[N_p]$.

In our sensitivity analysis, we consider the following ranges of variability for the three additional parameters: $E[N_p] \in \{20, \ldots, 100\}$, $\lambda_r \in [0.02, 0.1]$, and $\lambda_p \in [1, 50]$. The selected ranges include the original 3GPP2 parameters, and account for reasonable parameter modifications. For the resulting 6-parameter SA of our model, Figure 6.7 shows the first order and total sensitivity indices for cost and performance metrics. The following is observed.

- The parsing rate λ_p is definitely unessential (this is mainly due to the negligible value of the average parsing time compared to the other durations) and can be fixed to any value within its range.

- The earlier observation on $E[D]$ remains unchanged: it is only impacted by the DRX parameters M and m (including their interactions).
- $E[N_p]$ and λ_r are equally relevant as concerns G_{BS}, G_{UE} and R as they have almost the same total sensitivity index.
- The download time $E[W]$ is still mostly affected by N_u, but it is also impacted by the web page size $E[N_p]$ and to a lesser extent by the reading rate λ_r.

Our analysis reveals that λ_r and $E[N_p]$ are essential for our model. It is important to accurately estimate λ_r and $E[N_p]$ before using the model to optimize the power save configuration in the network.

6.9 Performance analysis and optimization

The section focuses on the analysis of the performance and on its optimization, using the model developed in Section 6.5 and validated in Section 6.7. Where not specified, we use the traffic parameters reported in Table 6.1.

Access delay. The access delay is the performance metric most impacted by the tunable parameters M (timeout threshold) and m (DRX cycle length), as confirmed by the sensitivity analysis. The access delay experienced in the network is reported in Figure 6.8 for the parameter set given in Table 6.1. $E[D]$ is sensitive to m, especially with low timeout values. However, reasonable values of m, e.g., below 20, yield access delay times not higher than 40 ms. As for the timeout threshold, an interesting value is $M = 256$ (see shape of $E[D]$ around $M = 256$ in Figure 6.8).

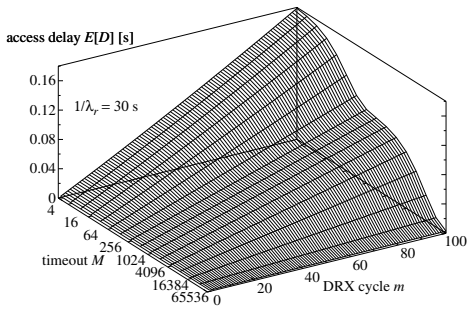
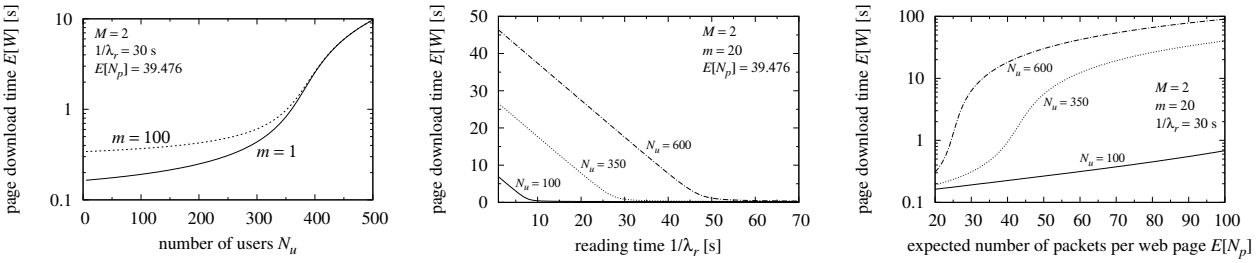


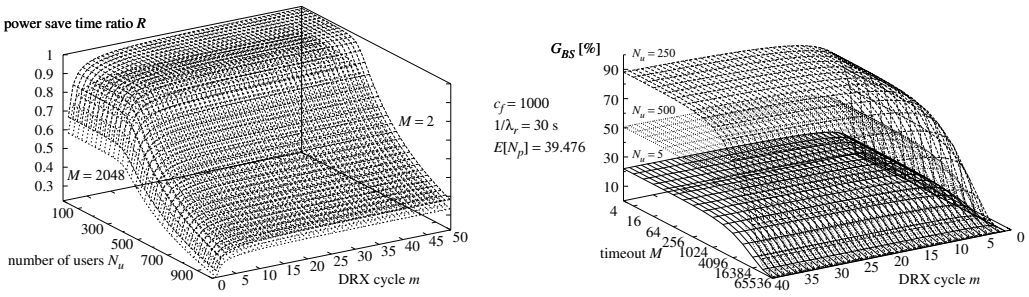
Figure 6.8: Access delay (independent of N_u).

Page download time. Figure 6.9 depicts the behavior of the expected page download time when one of the following terms is varied: (a) the number of users in the cell N_u; (b) the



(a) $E[W]$ grows with N_u (b) $E[W]$ decreases with reading time (c) $E[W]$ increases with $E[N_p]$

Figure 6.9: The expected page download time is insensible to DRX cycle length m and timeout M; it is roughly $E[\sigma]E[N_p]$ (tight lower bound).



(a) Great save opportunities with $m \geq 4$ (b) Relevant gain with $M \leq 1024$ and $m \geq 4$

Figure 6.10: Power save time ratio vs. m and N_u (a) and eNB gain vs. m and M (b).

expected reading time $1/\lambda_r$ (user's behavior); and (c) the expected web page size in packets $E[N_p]$. The following is observed. The expected page download time is small as long as $N_u < 350$. For a larger number of users, $E[W]$ increases abruptly (and linearly) with N_u. The value of m has only a negligible impact on the page download time: the latter is sensibly the same whatever the value of m. The same is observed concerning the parameter M (not reported here). Also, $E[W]$ increases with the reading rate λ_r as can be inferred from Figure 6.9.b. Indeed, larger reading times lower the load on the shared processor, thereby decreasing the expected service time and consequently the page download time. Last, longer web pages (this is a trend currently observed due mainly to large embedded objects and heavy scripts) yield larger download times.

Power save time ratio. We now consider the power save time ratio R. The most important parameters are m and N_u. We report the analytical results in Figure 6.10.a. The power save

time ratio R remains constant for a large range of number of users values but decreases as soon as $N_u > 350$.

Relative power save gain at eNB. Reasonably, the cost of transmitting a data packet is larger than the cost of transmitting a control packet, which usually takes less bandwidth. Both transmitting and signaling costs are much higher than the cost to stay on, which, in turn, is at least one order of magnitude greater than the cost to stay in power save mode. As an example, we use the following values: $c_{tx} = 100$, $c_{sg} = 50$, $c_{on} = 10$, and $c_{ps} = 1$. Additionally, as suggested by experimental measurements [33], we consider a base station cost one order of magnitude larger than the transmission cost: $c_f = 1000$. In the following $T_{ln} = T_{sub}/3$ and $T'_{ln} = T_{sub}$.

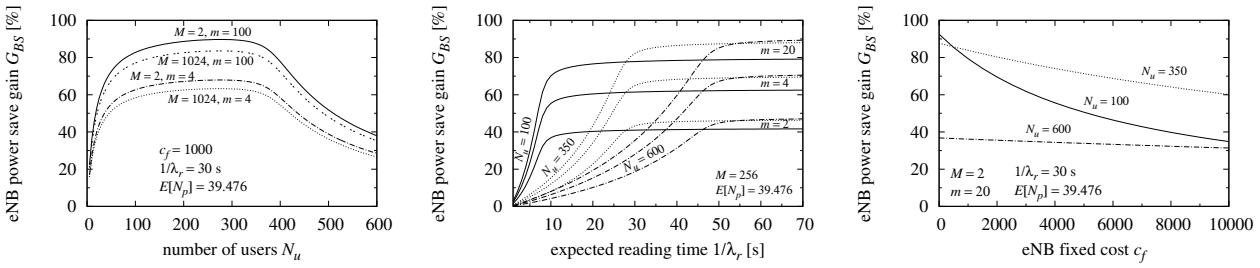
With the chosen cost parameters, the function $\gamma'(m)$ grows very fast for small m, but it quickly saturates. In practice, values of m larger than 20 do not give substantial gain advantages with respect to $m = 20$, that is the maximum value suggested by 3GPP for CPC. The relative gain at the eNB is reported in Figure 6.10.b for a few values of N_u. One can notice that low to medium values of the timeout, jointly with moderately high values of m, allow to obtain most of the potential gain for the current value of N_u. Observe that when few users are attached to the eNB, the main cost figure is c_f, which is fixed. However, as shown in Figure 6.11.a, if the number of users grows beyond 350, the gain recedes. In fact, with too many users, the system saturates and power saving opportunities diminish (cf. Figure 6.10.a).

We have investigated the effect of the expected reading time $1/\lambda_r$ on the eNB gain. The results are depicted in Figure 6.11.b for various values of N_u and m, the timeout M being fixed to 256. We observe that the relative gain at the eNB saturates as soon as the reading time reaches a threshold, which depends on the number of users. The saturation level of G_{BS} depends on the cycle length m. It is clear that small variations around the current recommended simulation value, that is equal to 30 seconds, will not affect the gain in cells with a moderate number of users (say $N_u \leq 300$). Decreasing the expected reading time in very large cells will yield less gain.

We now vary c_f. It is expected to obtain smaller relative power save gain should the fixed cost at the eNB be larger, and vice-versa (larger gain if smaller fixed cost). This is observed in Figure 6.11.c.

Relative power save gain at UE. The last metric we analyzed is the user's relative power save gain. We have varied successively the number of users N_u, the reading time $1/\lambda_r$, and the web page size (in packets) $E[N_p]$. Results are reported graphically in Figure 6.12 for timeout threshold $M = 2$. Substantial user power savings are possible if $m \geq 20$. Having timeout thresholds larger than 2 subframes slightly decreases the user gain (about 2% loss in G_{UE} if $M = 256, 1/\lambda_r = 30$ s, and $E[N_p] = 39.476$).

DRX protocol parameter optimization. We want to compute now the optimal values of M

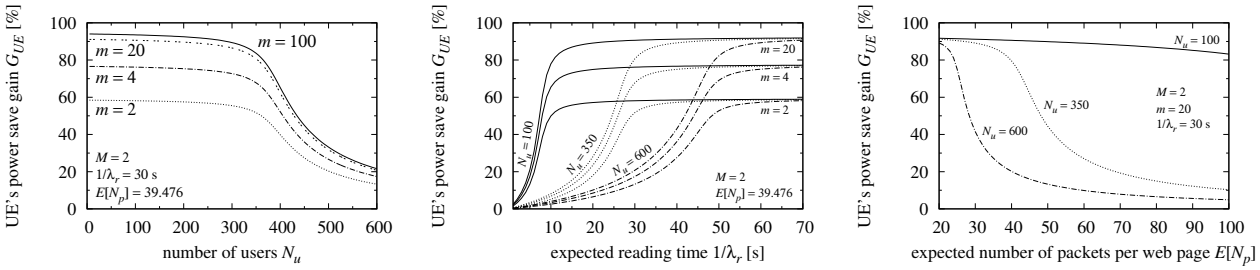


(a) Large gain for many N_u's if $m \geq 4$

(b) Gain saturates as λ_r decreases

(c) Smaller c_f yields a larger gain

Figure 6.11: G_{BS} vs. the number of cell users N_u, the reading time $\frac{1}{\lambda_r}$ and eNB's fixed cost.



(a) Lesser gain with larger capacity cell

(b) More gain if load decreases

(c) Larger web pages impair user gain

Figure 6.12: Analytical evaluation of the relative power save gain at UE.

and m that yield the highest gain while keeping low the access delay and the download time. We consider the eNB cost only, but the results can be easily extended to the UE.

Figure 6.13 shows some specific cases of system optimization. In the figure, W_x and D_x denote the maximum allowable download time and access delay, respectively. Each optimization is performed over M and m, given a fixed number of users N_u. Each optimized value of the gain is labeled with the pair (M, m) corresponding to the optimum. The figure shows that the gain exceeds 70% in cells with at least 50 users, while keeping the total web page download time bounded to less than 0.3 s, and the access delay below 0.05 s. However, with 300 users, the minimum download time grows above 0.3 s and the system cannot be optimized unless W_x was raised to at least 0.44 s. Note also that the optimization with very small values of the access delay (e.g., $D_x = 0.05$ s) can only be obtained by setting relatively long timeout and short DRX cycle values (e.g., $M = 64$ and $m = 9$). With higher access delay bounds (e.g., 0.1–0.3 s), and in cells with at most 100 users, the optimal timeout is the smallest possible, i.e., $M = 2$. In all cases reported in Figure 6.13, the optimization suggests to use very large values for m (larger

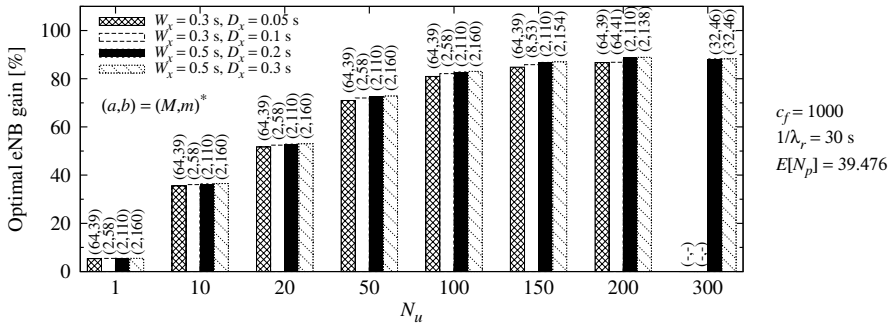


Figure 6.13: Relative gain for different number of users, optimized over bounded download time and access delay.

than 39). However, from Figure 6.10.b, it is clear that near-optimal gain can be obtained with values of m as low as 20.

Lessons learned. Our cost and sensitivity analysis shows that significant power savings are possible while users are guaranteed to experience high performance. In particular, we have unveiled that the threshold timeout does not need to be excessively short in order to enable a remarkable power save, e.g., using $M = 256$ turns into reasonable access delay (tens of milliseconds). We also observed that using $m = 20$ is a very good tradeoff between power save and access delay. In order to limit the download time, it is crucial to limit the number of active users in the cell (to less than 350 users, which is reasonable for 3GPP LTE, 802.16 and HSPA networks). What is also needed is to limit the web page size. In conclusion, we suggest that enforcing a *green attitude* for web designers, in terms of reducing the web page size and the number of embedded objects, would enable the cellular operator to use reasonable power save parameters (e.g., $m = 20$, $M = 256$) and so achieve a dramatic cost economy at both base station and mobile user sides, without any quality degradation.

6.10 Conclusions

In this chapter, we have shown how to model the activity of cellular users adopting the continuous connectivity model under realistic traffic conditions. To this aim, we used a G/G/1 PS queueing model, which has been validated through simulation. We first modeled the per-user activity and evaluated the service share that the base station processor can grant to each user. Thus, we have derived close-form expressions for busy and idle periods for each mobile user's connection. Second, we provided performance metrics and a cost model enlightening the impact of traffic and power save parameters on quality and cost of transmission. Third, we provided a sensitivity analysis to figure out the impact of each model parameter on performance and power consumption. Forth, we showed how to optimize the power save parameters to min-

imize the transmission cost under bounded access delay and page download time. Remarkably, we showed that with the considered parameter settings up to 90% or more of the transmission cost can be saved while preserving the quality of packet flows.

7

OPEN PROBLEMS

7.1 Summary

In this chapter, we present a set of questions that we believe could be interesting research opportunities. Some of these problems are derived from our studies and then they are partially addressed; meanwhile others remain totally open.

Keyword 7.1 *Optimization, control, quality of service, geographical locality, temporal locality, delayed cache networks,* q*-LRU, Least Frequently Used (LFU), sensitivity analysis.*

7.2 Optimization and control of cache networks

This last decade, optimization problems on cache networks have received considerable attention and researchers focused on finding the optimal *cache/content placement* or *location* within a network. However less has been done for the control of self-organizing caches or the management of quality of service specially when caching is performed on-demand or on-path.

One of the claims of this thesis is that *TTL-based caches are more flexible and fully configurable than popular LRU, FIFO or RND caches*. While the latter caches run a specific and rigid algorithm that either always keeps the least recently used content, orders files according to their access times, or manages files without distinction, TTL-based caches provide different parameters, namely the *TTLs*, to decide which contents should be kept or discarded from the memory. Indeed TTLs are *control parameters* which can be used for network optimization (e.g. by monitoring the load on end-servers and adjusting their values), for service differentiation (e.g. by allowing

certain type of contents to be stored or discarded based on the applications), and even for quality of service (e.g. by providing a minimum caching performance to users or End-to-content paths).

In the remainder of this section, we formulate a general optimization problem on TTL-based cache networks since TTL-based policies are more general than other classical replacement policies. Then, we carefully identify the information that can be shared between customers and cache network providers. And finally we address the cache selection problem that guarantee a minimum quality of service on End-to-content paths.

7.2.1 General optimization problem on TTL-based cache networks

Here, we formulate a general optimization problem where clients and cache network providers (network operators, content providers, or content distribution networks) sign a contract in order to satisfy the total of utility of users. Later in Section 7.2.2 we will identified the key information that could be in the terms of such a agreement.

Scenario considered. We consider an information-centric architecture represented by a bipartite graph $\mathcal{G}(U, V, E)$ where U is the set of users, V is the set of cache providers and $E = \{(u, v) : u \in U, v \in V\}$ is the set of all possible agreements between users and cache providers. We denote by $\mathcal{P}(u)$ the set of cache providers who have signed a contract with user u and $\mathcal{C}(v)$ the set of clients who registered to cache provider v.

We assume that each user u always requests the same file f_u from a catalog $\mathcal{F}$ and that each cache provider v manages a single cache of capacity C_v measured in number of files or chunks it may store. Note this latter assumption is not so restrictive since the case of a user requesting several files or a cache provider owning several caches can be easily accommodated by adding a *virtual user* per file requested or a *virtual cache provider* per cache facility. We further assume that each user has a utility function $\mathcal{U}_u(H_{P,u,v})$ which is a function of its hit probability; this utility can be the delay, energy consumption, hit rate, bandwidth, etc... Moreover, requests generated for each file (or equivalently by a user) may be described/approximated by renewal processes whose inter-request times have an exponential distribution (i.e. Poisson processes), 2-stage hyper-exponential distribution (i.e. Interrupted Poisson processes) or shifted-exponential distribution [95]. Since miss streams of TTL-based caches are also renewal processes under the latter assumption, the formulation of this problem remains valid for caches within a network.

Log-concave utility function. We showed in Chapter 4 that the hit probability $H_{P,u,v}$ is an increasing function of the TTL value $T_{u,v}$ that file f_u receives from the cache provider C_v which is maximized when $T_{u,v}$ is constant (resp. hyper-exponentially distributed) when the CDF $F_u(t)$ is concave (resp. not concave) under our renewal traffic assumption. Therefore, we will express

the utility function $\mathcal{U}_u(H_{P,u,v})$ directly in function of the control parameter $T_{u,v}$ i.e. $\mathcal{U}_u(T_{u,v})$. Our objective is to find the optimal configuration $\{T^*_{u,v} = E[T_{u,v}], u \in \mathcal{C}(v)\}$ that maximizes the total utility of users.

Without loss of generality, we focus on a single cache provider. We omit the subscript v that refers to the cache label in the variables and functions previously introduced. Hence the user utility function $\mathcal{U}_u(.)$ depends on the expected TTL T_u and assuming $|\mathcal{U}| = |\mathcal{F}| = N$, we formulate our optimization problem as follows.

7.1 Definition (Optimization problem)

$$\max_{T_1,\dots,T_N} \sum_{i=1}^{N} \mathcal{U}_i(T_i) \tag{7.1}$$

subject to

$$\sum_{i=1}^{N} \lambda_i (1 - H_{P,i}(T_i))\, Q_i = C, \tag{7.2}$$

$$\sum_{i=1}^{N} \lambda_i (1 - H_{P,i}(T_i))\, T_i \leq C, \tag{7.3}$$

$$\sum_{i=1}^{N} \lambda_i T_i \geq C \tag{7.4}$$

where λ_i *is the request rate of user* i, Q_i *is the expected sojourn time of file* i *in the cache, and* $H_{P,i}(T_i)$ *is a known function of* T_i.

For TTL-non-renewing caches we have $Q_i = T_i$, while for TTL-renewing caches $Q_i = E[T_i\, \mathbf{1}(X_i > T_i)] + E[(X_i + Q_i)\, 1(X_i < T_i)]$.

Our objective in (7.1) is to maximizes the total utility by finding the optimal vector $(T_1, \dots, T_N)$. The equality constraint in (7.2) is related to the total occupancy of files which should be equal to the cache size in average : $\sum_{i=1}^{N} O_{P,i} = C$, where $O_{P,i} = \frac{Q_i}{E[Y_i]}$ and $E[Y_i]^{-1} = \lambda_i(1 - H_{P,i})$ is the average inter-miss times by renewal theory.

The inequalities in (7.3) and (7.4) say that $\{T_i\}_i$ cannot take any positive values and they are derived as follows. By definition $T_i \leq Q_i$ and also $\lambda_i(1-H_{P,i})Q_i = O_{P,i}$, we get $\lambda_i(1-H_{P,i})T_i \leq O_{P,i} \leq \lambda_i T_i$. The last inequality follows from the fact that the average number of file i in the cache cannot exceed to the total number of request of file i during T_i. (7.3) and (7.4) are derived by summing over all files the inequalities on the occupancy probabilities.

Note that if one assume $T_i = T, \forall i$ and denote by $\Lambda = \sum_i \lambda_i$ (resp. $M_R = \sum_i \lambda_i(1 - H_{P,i})$) the total request (resp. miss) rate, (7.3) and (7.4) give the following bounds for values of T

$$\frac{C}{\Lambda} \leq T \leq \frac{C}{M_R}$$

The latter inequalities explain that T cannot be zero and an increase of its value increase its upper bound until the total miss rate M_R becomes zero. In this case, the occupancy $O_{P,i}$ and the hit probability $H_{P,i}$ are both equal to one. This implies the cache size should be $C \geq N$ than the number of files. To avoid this trivial case, we further assume $N > C$.

7.2.2 Cache selection problem: case of minimum quality of service guaranteed

In this section, we discuss the terms of an agreement when a client (user or cache) decides for data plan with one or several cache provider(s). First we identify the information that can be shared among the two parties; and we propose a dynamic cache selection algorithm that guarantees a minimum QoS to the client.

Which information can be shared?

In our scenario, clients and cache providers sign a contract; however, they do not necessary want to disclose information about their uses or infrastructure that they may consider as private or confidential. Our agreement is described as follows.

1. A user who subscribes for a cache plan may not want to give information about its daily activity (because he does not know *a priori*). This information can be translated in terms of its request rate. However, the request rate is important for the cache provider who may want to grab several users to maximize its profit with a small expenditure on cache facilities. Hence, a cache provider and a client may then agree on the **maximal request rate** λ_u allowed.

2. A cache provider usually considers that any information about its infrastructure (e.g. the maximal bandwidth B_{max} on their network, cache size C) cannot be disclosed for security reasons and thus not available to clients whose main concerns are about the quality of the service they will experience. Meanwhile, they have to provide some *useful* information that helps clients to decide and go for a cache plan with them. Cache provider may advertise the type (i.e. TTL-renewing or TTL-non-renewing) and the **minimum expected TTL value** T_{min} of the cache in terms of the contract.

3. Finally, the cache service can be priced based on the **occupancy** $O_{P,u}$ of file f_u in the cache which is a non-trivial function of the maximal client request rate λ_u.

Note that the value T_{min} can be computed for any cache replacement policy as the ratio of the cache capacity C and the maximum bandwidth available B_{max} on the link:

$$T_{min} = \frac{C}{B_{max}}.$$

On contrary to the cache capacity and the maximum available bandwidth, the information T_{min} is less sensitive, it can be estimated on fly and thus it can be disclosed. At the same time, a cache provider v will be able to predict the number of users they might still grab without damaging the user experience or the current quality of service that its users receive. Given the maximal client request rate λ_u, an estimate of the current minimum expected TTL value T_v can be found with the following equation:

$$T_{v,min} = \frac{C_v}{\sum_{u \in \mathcal{C}(v)} \lambda_u}.$$

Moreover, a client can easily estimate–without being aware of the traffic from other clients–the value of its utility function by computing the minimum hit ratio $H_{P,u,v}(T_{v,min})$ guaranteed by the cache provider. For end-users whose inter-request times have a concave CDF $F(t)$, if the cache provider adopts deterministic TTLs (which is the optimal caching strategy in this case) then the minimum hit ratio are given by

$$H^{\texttt{LRU}}_{P,min} = F(T_{min}) \quad \text{For TTL-renewing caches} \tag{7.5}$$

$$H^{\texttt{FIFO}}_{P,min} = \frac{M(T_{min})}{1 + M(T_{min})} \quad \text{For TTL-non-renewing caches} \tag{7.6}$$

where $F(t)$ is the CDF of inter-request times and $M(t)$ is the renewal function associated to $F(t)$.

M-QCSA: "Minimun QoS-based" Cache Selection Algorithm The user utility function is defined by the minimum QoS guaranteed i.e. the user selects the cache provider who ensures the best minimum level of hit probability. In fact, a client simply selects the cache that provides the best minimum expected TTL value T_{min} and he can be sure to get the best minimum quality of service. We can easily think about a simple scenario where cache servers initially send T_{min} to their clients and later update this value with the current minimum characteristic time they have estimated based on their current load.

Algorithm 10: *M-QCSA*

input : `MinExpectedTTL` $\{T_{v,min}, v \in \mathcal{P}(u)\}$

output: `BestProvider` v^*

1 **while** `Request on client` u **do**

2 `Check current min. expected TTL` $\{T_{v,min}, v \in \mathcal{P}(u)\}$;

3 $v^* \xleftarrow{\texttt{Select server}} \arg\max_{v \in \mathcal{P}(u)} \{T_{v,min}\}$;

4 **end**

It might happen that several clients have selected the same cache, this is actually an *advantageous* situation for the cache provider (minimal number of resources for huge revenue

in terms of number of clients). Clearly clients would get a minimal performance while other caches could provide better, but we recall that our aim is to guarantee a minimum QoS formulated in the agreement.

7.2.3 Perspectives on optimization and control

The optimization problem (7.1–7.4) is defined for each cache; hence, a distributed algorithm can be used to solve this problem at each cache of the network. If the *M-QCSA* is a simple algorithm that helps the client to select a cache based on the minimum QoS guaranteed, this algorithm can be implement at nodes of a network to build minimum-guaranteed QoS *End-to-content* paths. It remains to prove that optimizing utility locally at each cache leads to global optimum or sub-optimum at network level.

7.3 Modeling the geographical locality in cache networks

Cache network optimization is of a particular interest when accounting for the geographical popularity of contents and the physical deployment of caches (e.g. at base stations of mobile networks). The main goal is to minimize the average miss probability on cache deployed in a geographical area. We refer to this objective as the *geographical locality problem*.

7.3.1 Geographical locality problem statement

We consider an area with a set of users U requesting files from a catalog $\mathcal{F}$ which can be accommodated in a set of TTL-based caches V deployed according to some general independent stationary point processes $\{\mathcal{R}_v, v \in V\}$ with finite intensities $\{\lambda_v, v \in V\}$ respectively. Without loss of generality, user u always requests the same file f_u which is divided into m_u chunks of equal size.

We assume that chunks of all files are requested according to popularity distribution $\{q_{u,k}, u \in U, 1 \leq k \leq m_u\}$. Moreover, a cache v of size C_v which is located within a vicinity of radius r_u from user u can stored at most $n_{u,v}$ chunks of f_u such that $n_{u,v} \leq m_u$ and $C = \sum_{v \in V} C_v < M = \sum_{u \in U} m_u$. We denote by $\mathcal{N}(u)$ the subset of caches in the neighborhood of user u which contains any cache v located at a distance $d(u,v) \leq r_u$. Once added into cache v, the k-th chunk of file f_u receives a TTL $T_{v,u,k}$ which is a random variable having a CDF $T_{v,u,k}(t)$. We recall that the expected TTL value $E[T_{v,u,k}] \approx \tau_v$ at cache v can be obtained by applying either the characteristic time approximation (of LRU, RND or FIFO caches for example) or just assigned by the cache provider such that the capacity constraint is satisfied.

Our objective is to calculate the miss probability $Q_u(r_u)$ of user u on file f_u within an area of radius r_u; and latter, we also aim at finding the optimal number of caches x_u and also the

expected TTL values $T_{v,u,k}$ that minimize our metric of interest.

7.3.2 Geographical locality model

First we note that user u may not be able to download its file f_u

(c_1) if the number of caches $N_u = |\mathcal{N}(u)|$ is such that

$$\sum_{v \in \mathcal{N}(u)} n_{u,v} < m_u;$$

(c_2) otherwise a chunk of file u is no more available in the cache (i.e. its TTL expired).

In the following, we will only consider that if (c_1) is not satisfied, the file can be downloaded with probability one. In fact, the analysis can be easily extended to account the expiration condition (c_2) just by calculation the conditional miss probability.

Let us define the indicator function $\Delta_v(r_u) = \mathbf{1}(d(u,v) < r_u)$ for $(u,v) \in U \times V$. In other words, $\Delta_v(r_u) = 1$ if cache v covers an area where user u is located and $\Delta_v(r_u) = 0$ otherwise. Define $Z_V(r_u)$ as the number of different caches whose coverage area overlap such that user u is located in their intersected area. We have:

$$Z_V(r_u) = \sum_{v \in V} \Delta_v(r_u) \tag{7.7}$$

By definition $N_u = Z_V(r_u)$; therefore the miss probability $Q_u(r_u)$ of user u follows directly from Condition (c_1) and (7.7)

$$Q_u(r_u) = P\left(\sum_{v \in V} \Delta_v(r_u) n_{u,v} < m_u\right) \tag{7.8}$$

The average miss probability Q of the cache network is given by

$$Q(r) = \sum_{u \in U} q_u Q_u(r), \quad \text{where, we define} \quad q_u = \sum_{k=1}^{m_u} q_{u,k} \tag{7.9}$$

Since cache v is deployed independently of others according to the process $\mathcal{R}_v$, the variable $\Delta_v(r_u)$ is a Bernoulli random variable with parameter $p_v(r_u) = \lambda_v^{-1} \int_{r=0}^{r_u} P(d(v_0,v) > r)dr$ where $d(v_0,v)$ is the distance between any two points of the stationary process $\mathcal{R}_v$. The distance $d(u,v)$ can be seen as the forward recurrence point of $\mathcal{R}_v$.

It follows from (7.7) that $Z_V(r_u)$ is the sum of independent Bernoulli random variables $\Delta_v(r_u)$; therefore, $Z_V(r_u)$ **is a Poisson-Binomial random variable with parameters** $\{p_v(r_u), v \in V\}$. We established in Proposition 2.2 a necessary and sufficient condition $\mu_V(r_u) = E[Z_V(r_u)] <$

∞ under which the random variable $Z_V(r_u)$ converges almost surely. Then Propositions 2.3, 2.4 and 2.5 provide several approximations of the CDF of $Z_V(r_u)$ under specific conditions.

Note in the current formulation of the geographical locality problem, there is no restriction on the number of chunks m_u of file f_u and the number of chunks $n_{u,v}$ that can be stored in cache v. These quantities might be also random variables as well (e.g. m_u and $n_{u,v}$ could be exponentially distributed).

However, in the remainder we assume that m_u and $n_{u,v}$ are constant; moreover, $n_{u,v} = n_u, \forall v \in V$. In this case, (7.8) reduces to

$$Q_u(r_u) = P\left(Z_V(r_u) < x_u\right), \quad x_u = \left\lceil \frac{m_u}{n_u} \right\rceil \tag{7.10}$$

In the next two sections, we will find the probability $Q_u(r_u)$ when the caches are deployed according to (i.e. $\{\mathcal{R}_v, v \in V\}$ are) Poisson processes. The results generalize easily to hyper-exponential (resp. hypo-exponential or shifted-exponential) renewal processes described in Section 2.7.3 of Chapter 2.

7.3.3 Caches are deployed with same rate: single cache provider

In this case, we assume the intensity of $\mathcal{R}_v$ is Poisson point process with rate $\lambda_v = \lambda, \forall v \in V$. One can show that the series $\mu_V(r_u)$ diverges as $|V|$ goes to infinity, and thanks to Proposition 2.5 $Z_V(r_u)$ may be approximate by a deterministic function $\mu_V(r_u)$ such that it exists a minimal radius $r_u^{(0)}$ which satisfies

$$\mu_V(r_u^{(0)}) = x_u \quad \text{and} \quad Q_u(r_u) = 1, \forall r_u < r_u^{(0)}$$

However, when $|V|$ is finite the random variable $Z_V(r_u)$ has a Binomial distribution with parameters $\{|V|, p = 1 - e^{\lambda r_u}\}$. Bounds of the miss probability $Q_u(r_u)$ are then obtained from Hoeffding's inequality, Chernoff's inequality or classical results by Arratia and Gordon [9], and Feller [36].

7.3.4 Caches are deployed with modulated rates: multiple cache providers

We are interest in the situation where caches are deployed by a single provider with rates modulated by the number of caches already added or each provider puts one cache with a rate that depends on the number of providers already installed.

In this case, the intensity of $\mathcal{R}_v$ is $\lambda_v = v^{-\alpha}, \alpha > 0$ (Zipf distribution), $\lambda_v = \rho^v, \rho > 0$ (Geometric distribution), $\lambda_v = ce^{-\delta v^\beta}, c, \delta, \beta > 0$ (Light-tail distribution), or $\lambda_v = \phi(v)$ where $\phi(.)$ is a positive and decreasing function $\forall v \in V$.

We established in Sections 2.7.1 and 2.7.2 of Chapter 2 main results on the convergence and the approximation of the probability $Q_u(r_u) = P\left(Z_V(r_u) < x_u\right)$

7.3.5 Perspectives on geographical locality aware caches

The remainder on this work may consist in the derivation of approximate expressions of the miss probability $Q_u(r_u)$ that would ease the approximation of the average miss probability Q defined in (7.9), and also facilitate question of finding the number x_u which minimizes our metrics of interest.

7.4 Accounting for temporal locality in cache networks

In this section, we address the problem of performance evaluation of cache networks under non-stationary request traffic. We focus on cache running the Least Recently Used policy. Little work [20, 4, 94] has been done in this subject although it is commonly agreed that users behaviors/request are not stationary over the time (day/night, week day/weekend activities periods).

Following our effort in Chapter 2, we provide conditions of validity of the characteristic time approximation of LRU caches when requests are described by Cox processes or doubly stochastic Poisson processes. This results provides a theoretical foundation of experiments conducted in [4] on LRU caches where requests are described by special cox processes. Note that Cox processes is a class of non-stationary processes which generalizes the Markov-modulated request rate model of Carofiglio et *al.* [20] and the Shot Noise Model (SNM, i.e requests are described by non-homogeneous Poisson processes with time-varying rate) of Traverso et *al.* [94].

7.4.1 Non-stationary workload and cache models

We consider a catalog of N different files labeled $n \in 1, \dots, N$. Successive requests for file n follow **a non-stationary Cox point process** $\mathcal{R}_n := \{t_i^n\}_{i\geq 1}$ with stochastic rate process $\{\lambda_n(t), t \geq 0\}$ such that the average value of the intensity function exists and is equal to some constant, call it

$$\lambda_n^{avg} = \lim_{t\to\infty} \int_0^t \frac{\lambda_n(s)ds}{t}, \quad \text{with probability 1.}$$

We denote by t_i^n ($i \geq 1$) is the occurrence time of the i-th request of file n after origin time $o = 0$ (the analysis derived later easily generalizes when the origin is chosen $o > 0$). Throughout we will assume that the expected time $E[t_1^n]$ is finite for each file n, so that the CDF of the first request to file n after time $t = 0$ is given $p_n(t) = P(t_1^n < t)$ by

$$p_n(t) = 1 - E\left[e^{-\int_0^t \lambda_n(u)du}\right].$$

7.4.2 Limit behavior of LRU caches under Cox request processes

Define the indicator function $X_n(t) = 1$ if the file n is requested within $[0, t)$ and $X_n(t) = 0$ otherwise; and the counting process $M_N(t) = \sum_{n=1}^{N} X_n(t)$. One can check that Propositions 2.2, 2.3, 2.4 and 2.5 of Chapter 2 still hold.

Since Proposition 2.2 establishes the deterministic limit of $M_N(t)$ and thus that of $T_{B,N}$ when the expected value $\mu_N(t)$ of $M_N(t)$ diverges, we investigate here the remaining case i.e. when $\mu_N(t)$ converges and we establish sufficient conditions for such deterministic behavior holds. The process $\mathcal{R}_n$ is still a Cox process with a non-negative intensity rate function $\lambda_n(t)$. We denote by $\Lambda_n(t) = \int_0^t \lambda_n(u)du$ and we assume $\Lambda_n(t) < \infty$ for $t \in [0, \infty)$ (i.e. no explosions). One can show the following result.

Proposition 7.1 (Convergence and Approximation for Cox processes) *If the rate of the aggregated process $\mathcal{R}$ defined as $r_N(t) = \sum_{n=1}^{N} \lambda_n(t)$ converges to $r(t)$, then the mean $\mu_N(t)$ converges and it is bounded by $E[\int_0^t r(u)du]$ as N goes to infinity and for any finite $t > 0$.*
Moreover, if $\lambda_n(t) = \lambda(t)\phi(n)$ where $\phi(x)$ is a positive and decreasing function on $[0, +\infty)$, the rate $r(t)$ and the mean $s_N(t)$ can be approximated by $\hat{r}(t)$ and $\mu(t)$ respectively with an error bound $0 \leq \mu_\infty(t) - \mu(t) \leq \epsilon_1(t)$ given as following

$$\hat{r}(t) = \lambda(t) \times \int_1^\infty \phi(x)dx \tag{7.11}$$

$$\mu(t) = \int_1^\infty (1 - \mathcal{W}_t(-\phi(x)))\, dx\ ,\ \epsilon_1(t) = \int_0^1 (1 - \mathcal{W}_t(-\phi(x)))\, dx \tag{7.12}$$

where $\mathcal{W}_t(x) = E\left[e^{x\Lambda(t)}\right]$ is the moment generating function of $\Lambda(t) = \int_0^t \lambda(u)du$.

Proof Given that $\{\lambda_n(t)\}_n$ are non-negative random variables at time t, the intensity rate $r_N(t) = \sum_{n=1}^{N} \lambda_n(t)$ of the aggregated process $\mathcal{R}$ is an increasing non-negative sequence. By the Kolmogorov's $0 - 1$ law, $r_N(t)$ converges almost surely. Assuming that $r_N(t)$ converges to $r(t)$, it follows from the Monotone Convergence Theorem that $E[r_N(t)]$ converges to $E[r(t)]$. Having this in hand, we can easily show that the first moment of $M_N(t)$ converges. Since $1 - e^{-\Lambda_n(t)} \leq \Lambda_n(t)$, it follows that $p_n(t) \leq E[\Lambda_n(t)]$. Hence,

$$\mu_N(t) \leq \sum_{n=1}^{N} E[\Lambda_n(t)] = E\left[\sum_{n=1}^{N} \int_0^t \lambda_n(u)du\right] = E\left[\int_0^t r_N(u)du\right] \longrightarrow_{N\uparrow\infty} E\left[\int_0^t r(u)du\right].$$

If we assume also that $\lambda_n(t) = \lambda(t)\phi(n)$ where $\phi(.)$ is a positive and decreasing function on $[0, +\infty)$, we can show that

$$r(t) = \lambda(t) \sum_{n\geq 1} \phi(n) \approx \hat{r}(t) = \lambda(t) \int_1^\infty \phi(x)dx$$

and

$$p_n(t) = 1 - E\left[e^{-\phi(n)\int_0^t \lambda(u)du}\right] = 1 - E\left[e^{-\phi(n)\Lambda(t)}\right] = 1 - \mathcal{W}_t(-\phi(n)), \quad (7.13)$$

$$\mu_N(t) = \sum_{n=1}^{N} 1 - \mathcal{W}_t(-\phi(n)) \approx \mu(t) = \int_1^{\infty} (1 - \mathcal{W}_t(-\phi(x)))\, dx \quad (7.14)$$

$$0 \leq \mu_\infty(t) - \mu(t) \leq \epsilon_1(t) = \int_0^{1} (1 - \mathcal{W}_t(-\phi(x)))\, dx \quad (7.15)$$

The series-integrals approximation in (7.14)) holds since $1 - \mathcal{W}_t(-\phi(x))$ is decreasing. ∎

The function $\phi(x)$ defined in Proposition 2.14 and 7.1 has the following properties.

Remark 7.1 (The rate modulating function $\phi(n)$) *If the function $\phi(x)$ is a positive and monotonically decreases in $[0, +\infty)$, then the aggregated rate $\sum_n \lambda_n$ or $\sum_n \lambda_n(t)$ converges; therefore, results in Propositions 2.14 and 7.1 hold respectively. Moreover,* (7.12)*,* (7.13) *and* (7.15) *also hold if $\Lambda_n(t) = \int_0^t \lambda_n(u)du = \Lambda(t) \times \phi(n)$.*

Remark 7.2 (Non vanishing rate) *If the rate $\lambda_n = \phi(n)$ does not vanish at ∞ or the stochastic rate $\lambda_n(t)$ (resp. $\Lambda_n(t) = \int_0^t \lambda_n(u)du$) is a sample of a random variable $\lambda(t)$ (resp. $\Lambda(t)$), the term $p_n(t) = 1 - e^{-\phi(n)t}$ in the former case or the term $p_n(t) = 1 - E\left[e^{-\Lambda_n(t)}\right]$ in the latter case does not converge to zero. Hence $\mu_N(t)$ diverges and Proposition 2.5 hold.*

LRU caches under Shot Noise Model (SMN)

Another interesting case where the characteristic time approximation was shown to be accurate is the analysis of LRU caches under a generalized SNM assumption [4]. The authors considered that the request process $\mathcal{R}_n$ is a non-homogeneous Poisson process with a time-varying rate function $\lambda_n(t) = V_n \times \lambda(t - d_n)$ where V_n (the volume of requests related to file n) is a sample of a random variable V having a moment generating function $\mathcal{V}(x) = E\left[e^{xV}\right]$ and d_n (the instant at which the file n is introduced in the catalog) is a uniform variable within $[0, t)$ (with a CDF $D(t) = P(d_n < t)$) for all $1 \leq n \leq N$. The aggregated request process $\mathcal{R}$ they obtained is special Cox process with intensity rate $r_N(t) = \sum_{n=1}^{N} V_n \times \lambda(t - d_n)$. We have

$$p_n(t) = P(X_n(t) = 1) = P(X_n(t) = 1, d_n < t) = \int_{x=0}^{t} \left\{1 - \mathcal{V}\left(-\int_{u=x}^{t} \lambda(u - x)\, du\right)\right\} dD(x).$$

So $p_n(t)$ does not depends on n and it is equals to a value $p(t)$. We can also write $p_n(t)(1 - p_n(t)) = p(t)(1 - p(t))$ with $0 < p(t)(1 - p(t)) < p(t) < 1$. It follows that the mean $\mu_N(t)$ and variance $\sigma_N^2(t)$ diverge as $N \uparrow \infty$ since their general terms do not converge to zero. Therefore Proposition 2.5 applies and the approximation of $T_{B,N}$ the characteristic time by a constant $t_{B,N}$ is very accurate since $c_N^2(t)$ and $\gamma_N(t)$ vanish by the inequalities (2.8) and (2.9). This provides the theoretic foundation of the characteristic time approximation performed in [4].

Markov-Modulated Rate Process with Zipf popularity

In [20], the authors assume that there are N classes of contents and each class contains K identical content items. The request process $\mathcal{R}_n$ on the class n is a Poisson process with rate $\lambda_n = \lambda c n^{-\alpha}$ where $\alpha > 1$, $c > 0$ and λ is the rate of the aggregated request process $\mathcal{R}$. The content item k of the class n to be requested is chosen uniformly at random (i.e. the corresponding request process $\mathcal{R}_{n,k}$ is a Poisson process with rate $\lambda_{n,k} = \lambda_n/K$). Each content item k is divided into a geometric number C of chunks with mean κ. A request on a content item k corresponds to the request of its first chunk, and the chunk request rate is a constant ρ_k equals to the inverse of the virtual round trip time to get (i.e. the time it takes to download) a chunk of the content item k. The rate ρ_k is identical for any content item of the same class, hence we shall substitute the subscript k by the index of the corresponding class n.

The request process $\mathcal{R}_n$ on the class n is a Markov-Modulated Rate Process (MMRP) [20] where the inter-request time τ_n of contents in the class n is exponentially distributed with rate λ_n. During the time τ_n, we can distinguished two periods: (1) the ON-period having a length equals to $C\rho_n^{-1}$ where C is the geometric number of chunks of the requested content and (2) the OFF-period which corresponds to the time since the last chunk of the previous content was requested to the instant of request of the first chunk of the next content. Note that, both the ON-period and the OFF-period are exponentially distributed with the rates $\alpha_n = \rho_n\kappa^{-1}$ and β_n respectively such that

$$\alpha_n^{-1} + \beta_n^{-1} = \lambda_n^{-1} \tag{7.16}$$

They assumed a large cache i.e. $B = \delta N$, $0 < \delta < 1$. Given their settings, it is easy to prove the accuracy of their result [20, Proposition 5.1] which states that the miss probability of a content item k in class n is $e^{-\frac{\lambda}{K}ck^{-\alpha}t_{B,N}}$ where $t_{B,N}$ is a constant (actually an approximation of cache characteristic time). This result was first established by [57] with different arguments. In fact, we note that the Riemann series $\sum_n \lambda_n$ converge since $\alpha > 1$. By proposition 2.14, $\mu_N(t)$ converges and we can approximated $M_N(t)$ by its mean $\mu_N(t)$. The cache characteristic time $T_{B,N}$ may be approximated by the root of the following $\mu_\infty(t_{B,N}) = B$ or approximated by the closed-form expression $t_{B,N} = N^\alpha \Psi^{-1}(\delta) + o(N^\alpha)$ as in Section 2.7.2.

Generalized Shot Noise Model (GSNM): 2-state Cox request processes

On contrary to [20] where it was assumed that the request process $\mathcal{R}_n$ on the class n is a Poisson process, we consider that $\mathcal{R}_n$ is the superposition of K request processes $\{\mathcal{R}_{n,k}\}_{k=1}^K$. The request process $\mathcal{R}_{n,k}$ of the k-th content item in the class n is a special Cox process with a stochastic request rate $\lambda_{n,k}(t) = \rho_{n,k} \times \chi_n(t)$ where $\rho_{n,k}$ is a mark that captures the randomness

of the virtual round trip time [20, Observation 4.2] and $\{\chi_n(t)\}_t$ is a 2-state continuous time Markov chain having a state space $\{0 = \text{OFF}, 1 = \text{ON}\}$ with transition rates β_n from state 0 to state 1 and α_n from state 1 to state 0 respectively. The processes $\mathcal{R}$ and $\mathcal{R}_n$ are Cox processes.

Note that if the Markov chain $\{\chi_n(t)\}_t$ has only one state, say state 1, and $\chi_n(t)$ is a deterministic function of time, we retrieve the SNM of [4]. If in addition, we set $\rho_{n,k} = 1$ the process $\mathcal{R}_{n,k}$ reduces to the non-homogeneous Poisson process with time varying rate of [94].

Let us introduce the following Bernoulli event $\chi_n(t)$ associated to the state of the Markov chain and we denote by $q_n(t) = P(\chi_n(t) = 1)$. We have

$$q_n(t) = E[\chi_n(t)] = \frac{\beta_n}{\beta_n + \alpha_n} + \frac{\alpha_n}{\beta_n + \alpha_n} e^{-(\alpha_n + \beta_n)t}.$$

We assume that the K content items in each class n have identical round trip times i.e. $\{\rho_{n,k}\}_{n,k}$ only depends on the network queuing delay and thus they are samples of a random variable ρ having a moment generating function $\mathcal{V}(x) = E[e^{x\rho}]$.
The total request rate $r_N(t)$ of the aggregated process $\mathcal{R}$ is given by

$$r_N(t) = \sum_{n=1}^{N} \sum_{k=1}^{K} \lambda_{n,k}(t) = K\rho \sum_{n=1}^{N} \chi_n(t) \tag{7.17}$$

This rate $r_N(t)$ converges almost surely if and only if the series $\sum_n q_n(t)$ converge. In fact, $r_N(t)$ can be written as the sum of Bernoulli events $\{\chi_n(t)\}_n$ as in Eq(7.17). By Proposition 2.2, $\sum_n \chi_n(t)$ converges almost surely if and only if the series $\sum_n q_n(t)$ converge. In this latter case, $q_n(t) \to_{n\uparrow\infty} 0$ i.e. it exists a class with a rank n_0 such that for any class n having a rank $n \geq n_0$ the Markov chain $\{\chi_n(t)\}_t$ has an absorbing state at $\chi_n(t) = 0$ $(= \text{OFF})$. The existence of this absorbing state means that items of classes $\{n \geq n_0\}$ are never requested. A sufficient condition on rates α_n and β_n such that the series $\sum_n q_n(t)$ converge is

$$\alpha_n^{-1} + \beta_n^{-1} = \omega(1/n^{-\alpha}). \tag{7.18}$$

Note that the latter condition (7.18) is similar to (7.16) needed in [20, Proposition 5.1]. Since the convergence of $r_N(t)$ implies that of $\mu_N(t) = E[M_N(t)]$ by Proposition 7.1, the convergence of $M_N(t)$ follows from Proposition 2.2.
We denote by $Q(t)$ the sum of the series $\sum_n q_n(t)$ when (7.18) holds. Using the equality

$$\Lambda_{n,k}(t) = \int_0^t \lambda_{n,k}(u)du = \rho \times \int_0^t \chi_n(u)du,$$

one can show that

$$p_{n,k}(t) = q_n(t)\,(1 - \mathcal{V}(-t)) \tag{7.19}$$

$$\mu_N(t) = \sum_{n=1}^{N} \sum_{k=1}^{K} p_{n,k}(t) = K\,(1 - \mathcal{V}(-t)) \sum_{n=1}^{N} q_n(t) \longrightarrow_{N\uparrow\infty} \mu(t) = KQ(t)\,(1 - \mathcal{V}(-t)) \tag{7.20}$$

since

$$1 - p_{n,k}(t) = E\left\{E\left[e^{-\int_0^t \lambda_{n,k}(u)du}|\rho\right]\right\} = E\left[1 - q_n(t) + q_n(t)e^{-\rho t}\right] = 1 - q_n(t)E\left[1 - e^{-\rho t}\right].$$

Finally, the cache characteristic time $T_{B,N}$ may be approximated by a constant $t_{B,N}$ solution of the following fixed point equation

$$\mu(t_{B,N}) = B \quad \text{i.e.} \quad K\left(1 - \mathcal{V}(-t_{B,N})\right) = B/Q(t_{B,N}).$$

7.4.3 Perspectives on temporal locality aware caches

If performance metrics of caches under doubly stochastic Poisson processes can be addressed using results [4] or following the Ross's analysis of non-stationary queues (see Ross's conjecture [84]); the question that remains open is the characterization of the miss stream of a cache under our non-stationary generalized shot noise model in order to provide new insights on cache networks.

7.5 Delayed cache networks

Delays are usually studied as objective (i.e. minimize a delay function) or constraints of optimization problems (see Section 7.2.1). However it is not well understood how they modified the request processes within network. Most of cache network models [23, 82, 41, 42] including our TTL-based model consider that processing times and delays are negligible; hence they assume instantaneous transmission. If this assumption eases the calculation, it is not well explained how existing models could be extended to account delay in cache networks.

In this section we show how our TTL-based model can easily be extended for more realistic scenarios accounting for non-zero delays and we describe a delayed TTL-based model that could be use to analyze delayed cache networks.

7.5.1 Delay as a Bernoulli random variable

A slight modification of our zero-delay TTL-based model to account for transmission delay can be described as follows. When a cache miss occurs on file n, the file is successfully retrieved (i.e. not lost) and instantaneously downloaded into the cache with a probability d_n as shown in Figure 7.1. Assuming renewal request processes, once can easily show that miss streams are also renewal processes where inter-miss times are of two types.

7.5.2 Delay as a continuous random variable

We consider that upon a cache miss, it takes a random duration D_n to retrieve file n from the server. Requests that occur during D_n are not forwarded and not replied until file n becomes

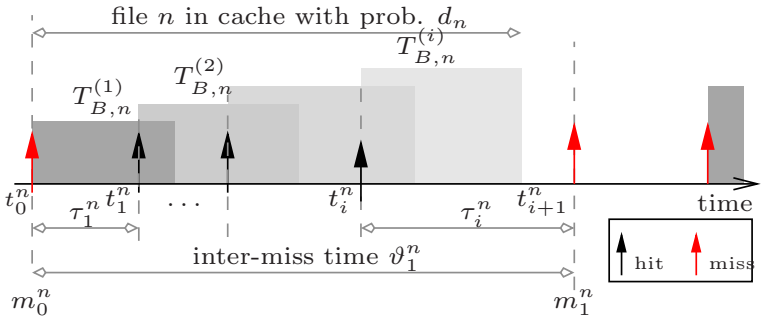


Figure 7.1: A delayed TTL-based cache, delay on file n as Bernoulli rv of parameter d_n.

available in the cache as illustrated in Figure 7.2. Assuming that requests are described by a

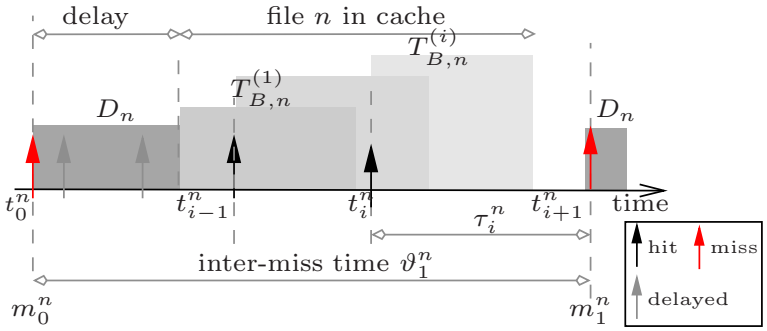


Figure 7.2: A delayed TTL-based cache, delay D_n on file n as continuous random variable.

renewal process, the miss process is also a renewal process. Moreover the metrics of interest, the distribution of the generic inter-miss time ϑ_1^n, and the extension to cache network are obtained from straightforward calculation.

7.5.3 Perspectives on delayed cache networks

It can be reasonable to consider a per-hop delay i.e. define a delay between a cache and each of its neighbors. However the appropriate distribution of the delay is still unknown. We believe performance of delayed cache networks can be significant contribution with respect to the zero-delay models studied in this dissertation and existing work.

7.6 On design of cache networks

7.6.1 q-LRU, LFU and mixed TTL-based policies

q-LRU and LFU policies The q-LRU policy behaves as LRU policy but it adds a file into the cache upon a cache miss with a probability q. Caches running the q-LRU replacement policy can be study using the TTL-based model described in Figure 7.1. The performance metrics and the characterization of the miss stream of a caches running the Least Frequently Used (LFU) can be deduced from that of q-LRU caches by letting $q \to 0$. This results was proven by Martina et *al.* [72]; therefore we can derive the analysis of general and heterogeneous cache networks where caches may run LRU, q-LRU, FIFO, RND, and LFU replacement policies.

Mixed TTL-based policies A slight extension of TTL-based caches is obtained by combining the behavior of TTL-based renewing caches and that of TTL-based non-renewing caches as follows. Upon a cache miss, the TTL is always initialized and when a cache hit occurs the TTL is reset with a probability r. If the request arrival process is a renewal process, the analysis of this modified TTL-based cache can be easily derived by considering a TTL renewing caches fed by requests described by the thinned-renewal request process [52] resulting from the Bernoulli thinning with parameter r of the original request process.

7.6.2 Cooperative caching strategies

Several meta-algorithms for hierarchical Web caches has been proposed in [67]; these algorithms may be extended to general and heterogeneous cache networks since requests are routed as polytrees. Their analysis becomes more easier as shown by Martina et *al.* [72] for tree-based network topology where requests flow in same direction i.e. from leaves to the root. The main hint is to translate a network caching strategy on simple caches into a simple caching strategy (where contents are stored everywhere) on probabilistic TTL-based caches. One can also study general and heterogeneous cache networks with heterogeneous caching strategies i.e. where several cooperative caching strategies are deployed at different levels of the networks (e.g. at core/edge or leaf/intermediate/root caches).

7.6.3 Sensitivity analysis of performance metrics

The impact of parameters (e.g. the topology, number of caches, cache sizes, cache replacement policies, etc.) on performance metrics of cache networks are not well understood and not yet deeply addressed in the literature. Sensitivity analysis such as the one performed in Section 6.8 of Chapter 6 could provide clear insights for the design of cache networks.

7.7 Conclusion

In this chapter we clearly formulate two problems we believe can lead to significant contributions in the networking community but also that could be of a particular interest for industrial. The optimization and control of cache networks where caching is done on-demand and on-path appears simple thanks to our TTL-based caches. Also fine grain performance evaluation of geographical and temporal aware cache networks can be easily held within our TTL-based framework. The direct consequence of relying on our programmable TTL-based caches is that all results apply to popular replacement policies such as LRU, q-LRU, RND, FIFO, and LFU. However, there are other concerns which are not presented in this chapter. For instance, (i) the temporal locality problem due to non-stationary traffic patterns, (ii) the non-zero delay problem akin to congested links, and (iii) sensitivity analysis of performance metrics in cache networks are issues yet to be addressed.

8

CONCLUSIONS

In this dissertation, we presented new models for performance evaluation of general cache networks and Power save analysis in wireless access networks. These models provide solutions and tools to adapt users needs/behaviors, to better design core and access networks, to optimize future technologies, to control quality of service, to improve quality of experience, and to predict states in content-oriented and wireless networks.

Our initial concern was about moving contents close to mobile or fixed users by the mean of on-demand caching in the networks. First, we generalized the well-known concept of expiration-based policy and we defined general Time-To-Live (TTL)-based caches. We showed that caches running popular replacement policies such as LRU, RND and FIFO asymptotically behave as TTL-based caches when they are fed by general stationary request processes. This leads us to see LRU, RND, FIFO and other variants as existing implementations of TTL-based caches; moreover, we propose Pra-TTL caches as a general and feasible practical deployment of TTL-based caches. Second we provide a unifying framework for the exact performance evaluation of general and heterogeneous TTL-based cache networks where requests are correlated and described by Markov-renewal processes. Our framework is built on top of the *Theory of Geiger Counters* and *Markov-renewal theory* in one hand, and also on the translation of network primitives (resp. multiple arrivals and multiple destinations) into well-known operations on processes (resp. superimposing and dependent-thinning of request processes) in the other hand. We studied three valid application cases for our theoretic findings. In the first case study, TTL-based caches are proposed to be deployed as cache replacement policy on routers of Content-Centric Networks. In the second case, TTL-based caches are used to model the behavior of DNS caches that override the timer marked on resource records by authoritative servers. Finally, we derive TTL-based model of general networks made of LRU, FIFO and RND caches.

In all these applications, we assume that requests were described by renewal processes and we found that our framework accurately predicts all metrics of interest with general relative errors of order of 5–20% in all the scenarios we tested.

The other issue we addressed in this thesis was about the energy balance in next generation wireless access network. We focused the situation where users are continuously connected to base stations (or eNBs) and access contents according to general stationary request processes. Our energy cost and sensitivity analysis shows that significant power save can be achieved while users are guaranteed to experience high performance. More precisely, we found an optimal configuration of parameters of DRX power saving protocols that leads to a very good tradeoff between power save and access delay. Our studies also provide some recommendations for both network operators and content providers: the number of active users in the cell should be less than 350 users at least for 3/4G mobile networks such as 3GPP LTE, 802.16 and HSPA; and web/applications designers should adopt a *green attitude* which consists of reducing the web page size and the number of embedded objects. Remarkably, we showed that with the considered parameter settings one can achieve a dramatic cost economy i.e. up to 90% or more of the transmission cost at both base station and mobile user sides while preserving the QoS.

Finally, thanks to our TTL-based caches we formulated few problems yet to be addressed and that we believe could be significant contributions in the networking research community. These problems are described more realistic conditions such geographical and temporal localities of request processes, delays due to congestion, control and optimization in cache networks. We also believe that a sensitivity analysis of cache network performance may help to better understand how our metrics of interest are impacted by input parameters. Such analysis could help to find the optimal network topology and cache settings with respect to traffic patterns. Definitely TTL-based cache networks are far to be fully understood although a huge effort have been done here to evaluate their performance; however they remains an exploratory area of new scientific contributions.

BIBLIOGRAPHY

[1] 3GPP TS 25.214. Physical layer procedures (FDD), release 8, v8.9.0, March 2010. 175

[2] 3GPP2 C.R1002-B v1.0. CDMA2000 evaluation methodology - Revision B, Dec. 2009. xvii, 178

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, July 2012. 75, 128

[4] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, and S. Niccolini. Analyzing the performance of lru caches under non-stationary traffic patterns. *CoRR*, abs/1301.4909, Jan. 2013. 7, 16, 18, 209, 211, 213, 214

[5] S. Albin. Approximating a point process by a renewal process, II: Superposition arrival processes to queues. *Operations Research*, 32(5), Sept. 1984. 11, 133, 149, 158, 164, 166, 167

[6] J. Almhana, Z. Liu, C. Li, and R. McGorman. Traffic estimation and power saving mechanism optimization of IEEE 802.16e networks. In *Proc. of IEEE ICC 2008*, pages 322–326, Beijing, China, May 2008. 9, 174

[7] S. Alouf, E. Altman, and A. Azad. M/G/1 queue with repeated inhomogeneous vacations applied to IEEE 802.16e power saving. In *Proc. of ACM SIGMETRICS 2008*, volume 36 of *Performance Evaluation Review*, pages 451–452, Annapolis, Maryland, USA, June 2008. 9, 174

[8] S. Alouf, V. Mancuso, and N. Choungmo Fofack. Analysis of power saving and its impact on web traffic in cellular networks with continuous connectivity. *Pervasive and Mobile Computing*, 8(5):646–661, 2012. 13

[9] R. Arratia and L. Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of Mathematical Biology*, 51(1):125–131, 1989. 208

[10] F. Baccelli and P. Brémaud. *Elements of Queueing Theory. Palm Martingale calculus and Stochastic recurrences*. Applications of Mathematics, Stochastic Modelling and Applied Probability, Springer, Berlin, Germany, second edition, 2003. 11, 19, 20, 30, 31, 32, 34, 57, 67, 78, 120, 152

[11] S. Baek and B. D. Choi. Analysis of discontinuous reception (DRX) with both downlink and uplink packet arrivals in 3GPP LTE. In *Proc. of ACM QTNA 2011*, pages 8–16, Seoul, Korea, August 2011. 9, 174

[12] S. Baek and B. D. Choi. Performance analysis of sleep mode operation in IEEE 802.16m with both uplink and downlink packet arrivals. In *Proc. of IEEE CAMAD 2011*, pages 112–116, Kyoto, Japan, June 2011. 9, 174

[13] R. J. Bayardo, R. Agrawal, D. Gruhl, and A. Somani. Youserv: a web-hosting and content sharing tool for the masses. In *Proc. ACM WWW'02*, pages 345–354, New York, USA, 2002. 101, 105

[14] G. Bianchi, N. Blefari-Melazzi, A. Caponi, and A. Detti. A general, tractable and accurate model for a cascade of caches. *CoRR*, abs/1309.0718, 2013. 128

[15] J. R. Bitner. Heuristics that monotonically organize data structures. *SIAM J. Computing*, 8:82–110, 1979. 5, 16

[16] C. Bontu and E. Illidge. DRX mechanism for power saving in LTE. *IEEE Communications Magazine*, 47(6):48–55, June 2009. 175

[17] P. Burville and J. Kingman. On a model for storage and search. *Journal of Applied Probability*, 10:697–701, 1973. 5, 6, 16, 98

[18] J. M. Cabe. On serial files with relocable records. *Operations Research*, 13:609–618, 1965. 5, 16

[19] T. Callahan, M. Allman, and M. Rabinovich. On modern DNS behavior and properties. *ACM SIGCOMM Computer Communication Review*, 43(3):7–15, July 2013. 29, 101, 102, 105, 113

[20] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino. Modeling data transfer in content-centric networking. In *Proc. ITC 23rd*, San Francisco, CA, USA, Sept. 2011. 50, 128, 209, 212, 213

[21] G. Casale, E. Zhang, and E. Smirni. KPC-Toolbox: Simple yet effective trace fitting using markovian arrival processes. *5th Intl. Conf. on the Quantitative Evaluation of SysTems (QEST'08)*, 2008. 120

[22] E. Çinlar. Introduction to stochastic processes. *Prentice Hall*, 1975. 11, 15, 29, 53, 72, 73

[23] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: modeling, design and experimental results. *IEEE J. on Selected Areas in Communications*, 20(7):1305–1314, 2002. 7, 8, 9, 12, 15, 16, 17, 18, 21, 23, 24, 35, 41, 50, 66, 72, 127, 128, 129, 130, 131, 134, 214

[24] H. Choi and J. Limb. A behavioral model of Web traffic. In *Proc. of ICNP 1999*, pages 327–334, Washington, DC, USA, Oct. 1999. 174

[25] N. Choungmo Fofack, P. Nain, G. Neglia, and D. Towsley. Analysis of ttl-based cache networks. In *Proc. ACM ValueTools'12*, Cargèse, Corsica, France, Oct. 2012. 13, 16, 127, 128, 130, 135

[26] N. Choungmo Fofack, P. Nain, G. Neglia, and D. Towsley. Analysis of ttl-based cache networks. Technical Report RR-7883, Inria, Sophia Antipolis, France, July 2012. 77, 82, 135

[27] E. Coffman Jr. and P. Jelenkovic. Performance of the move-to-front algorithm with markov-modulated request sequences. *Operations Research Letters*, 25:109–118, 1999. 6, 16

[28] D. R. Cox. *Théorie du Renouvellement*. Monographies DUNOD., Paris, 1966. 11, 105, 132, 152

[29] R. Cukier, J. Schaibly, and K. Shuler. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. III. Analysis of the approximations. *Journal of Chemical Physics*, 63(3):1140–1149, 1975. 191

[30] E. Dahlman, S. Parkvall, J. Skold, and P. Beming. *3G Evolution: HSPA and LTE for Mobile Broadband*. Academic Press, Oxford, UK, Second edition, 2008. 175, 176, 177, 184

[31] A. Dan and D. Towsley. An approximate analysis of the lru and fifo buffer replacement schemes. In *Proc. ACM SIGMETRICS'90*, pages 143–152, Boulder, CO, USA, May 1990. 6, 7, 8, 16, 113, 128, 129, 130, 131

[32] R. Durrett. *Probability: Theory and Examples*. Cambridge University Press, fourth edition, 2010. 22

[33] F. Corrêa Alegria and F.A. Martins Travassos. Implementation details of an automatic monitoring system used on a Vodafone radiocommunication base station. *IAENG Engineering Letters*, 16(4):529–536, Nov. 2008. 195

[34] R. Fagin and T. G. Price. Efficient calculation of expected miss ratios in the independent reference model. *SIAM J. Comput.*, 7(3):288–297. 7

[35] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. In *Proc. IEEE INFOCOM'97*, Kobe, Japan, Apr. 1997. 67, 70, 74

[36] W. Feller. Generalization of a probability limit theorem of cramer. *Transactions of the American Mathematical Society*, 54(3):pp. 361–372, 1943. 208

[37] J. A. Fill. Limits and rate of convergence for the distribution of search cost under the move-to-front rule. *Theoretical Computer Science*, 176:185–206, 1996. 5, 7

[38] J. A. Fill and L. Holst. On the distribution of search cost for the move-to-front rule. *Random Structures Algorithms*, 8(3):179–186, 1996. 16, 17, 50, 70, 74, 113, 128

[39] W. Fischer and K. Meier-Hellstern. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 18:149–171, Jan. 1991. 44, 91

[40] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39:207–229, 1992. 5, 6, 16

[41] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for LRU cache performance. *CoRR*, abs/1202.3974, 2012. 8, 16, 17, 18, 21, 23, 24, 26, 32, 35, 38, 39, 41, 42, 44, 131, 134, 136, 214

[42] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy. Performance of the random replacement policy for networks of caches. In *Proc. ACM SIGMETRICS/PERFORMANCE'12*, pages 395–396, London, England, UK, June 2012. 8, 9, 16, 24, 50, 113, 128, 214

[43] E. Gelenbe. A unified approach to the evaluation of a class of replacement algorithms. *IEEE Trans. Computers*, 22(6), 1973. 7, 16, 143

[44] J. Gray and F. Putzolu. The 5-minute rule for trading memory for disc accesses and the 5-byte rule for trading memory for CPU time. Technical Report TR 86.1, PN 87615, TandemComputers, Sophia Antipolis, France, May 1985. 16

[45] K. Han and S. Choi. Performance analysis of sleep mode operation in IEEE 802.16e mobile broadband wireless access systems. In *Proc. of IEEE VTC 2006-Spring*, volume 3, pages 1141–1145, Melbourne, Australia, May 2006. 9, 174

[46] Q. He and H. Zhang. On matrix exponential distributions. *Adv. in Applied Probability*, 39:271–292, 2007. 67, 82

[47] W. J. Hendricks. The stationary distribution of an interesting markov chain. *Journal of Applied Probability*, 9:231–233, 1972. 5, 16

[48] Y. Hong. On computing the distribution function for the poisson binomial distribution. *Journal Computational Statistics & Data Analysis*, 59(0):41–51, Oct. 2012. 21

[49] G. Horváth. Matching marginal moments and lag autocorrelations with MAPs. In *Proc. ACM ValueTools'13*, Torino, Italy, Dec. 2013. 127, 149

[50] Y. T. Hou, J. Pan, B. Li, and S. Panwar. On expiration-based hierarchical caching systems. *IEEE J. on Selected Areas in Communications*, 22(1), Jan. 2004. 50, 102

[51] Y. T. Hou, J. Pan, K. Sohraby, and S. X. Shen. Coping miss synchronization in hierarchical caching systems with nonlinear ttl functions. *IEEE Communications Society*, 2004. 101

[52] V. Isham. Dependent thinning of point processes. *Journal of Applied Probability*, 17(4):987–995, Dec. 1980. 57, 74, 127, 134, 144, 145, 146, 147, 216

[53] V. Jacobson, D. Smetters, J. Thorntorn, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *Proc. ACM CoNEXT'09*, Rome, Italy, Dec. 2009. 50, 66, 75, 128

[54] P. Jelenkovic. Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. *The Annals of Probability*, 9(2):430–464, 1999. 5, 6, 16, 18, 41, 44

[55] P. Jelenkovic and A. Radovanovic. Least-recently used caching with dependent requests. *Theoretical Computer Science*, 326:293–327, 2004. 6, 7, 16

[56] P. Jelenkovic, A. Radovanović, and M. Squillante. Critical sizing of lru caches with dependent requests. *Journal of Applied Probability*, 43(4):1013–1027, Dec. 2006. 7, 16

[57] P. R. Jelenkovic and X. Kang. Characterizing the miss sequence of the lru cache. *SIGMETRICS Perform. Eval.*, 36(4):119–121, Aug. 2008. 16, 17, 18, 50, 128, 129, 212

[58] S. Jin, X. Chen, D. Qiao, and S. Choi. Adaptive sleep mode management in IEEE 802.16m wireless metropolitan area networks. *Computer Networks*, 55(16):3774–3783, November 2011. 9, 174

[59] J. Jung, A. W. Berger, and H. Balakrishnan. Modeling TTL-based internet caches. In *Proc. IEEE INFOCOM'03*, San Francisco, CA, USA, Mar. 2003. 11, 16, 29, 50, 68, 70, 74, 102, 105, 110, 112, 114

[60] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. In *Proc. ACM SIGCOMM Workshop on Internet Measurement (IMW '01)*, New York, NY, USA, Nov. 2001. 66, 102

[61] S. Karlin. Total positivity. 1, 1968. 67, 82

[62] W. F. King. Analysis of demand paging algorithms. *Information Processing*, 71:485–490, 1972. 5, 6, 16, 98

[63] T. Kolding, J. Wigard, and L. Dalsgaard. Balancing power saving and single user experience with discontinuous reception in LTE. In *Proc. of IEEE ISWCS 2008*, pages 713–717, Reykjavik, Iceland, 2008. 175

[64] V. S. Korolyuk. Superposition of Markov renewal processes. *Cybernetics and Systems Analysis*, 17(4):556–560, 1981. 57, 58, 59, 61

[65] V. G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall/CRC texts in statistical science series, Taylor and Francis Group, Boca Raton, FL, USA, second edition, Dec. 2009. 11, 152

[66] N. Laoutaris. A closed-form method for LRU replacement under generalized power-law demand. *CoRR*, abs/0705.1970, May 2007. 17, 18, 21, 35, 134

[67] N. Laoutaris, S. Syntila, and I. Stavrakakis. Meta algorithms for hierarchical web caches. In *Proc. 23rd IEEE Intl. Performance Computing and Communications Conference IPCCC'04*, Phoenix, Arizona, USA, Apr. 2004. 68, 128, 131, 216

[68] I. Lassoued, A. Krifa, C. Barakat, and K. Avrachenkov. Network-wide monitoring through self-configuring adapative system. In *Proc. IEEE INFOCOM'11*, Shanghai, China, Apr. 2011. 88

[69] A. T. Lawrence. Dependency of intervals between events in superposition processes. *Journal of the Royal Statistical Society, Series B (Methodological)*, 35(2):306–315, 1973. 8, 67, 78, 86, 115, 120, 130, 152

[70] V. Mancuso and S. Alouf. Power save analysis of cellular networks with continuous connectivity. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–9, 2011. 13

[71] O. Manor. Bernoulli thinning of a Markov renewal process. *Applied Stochastic Models and Data Analysis*, 14(3):229–240, Jan. 1998. 57, 58, 59, 61, 144

[72] V. Martina, M. Garetto, and E. Leonardi. A unified approach to the performance analysis of caching systems. *CoRR*, abs/1307.6702, Sept. 2013. 7, 9, 12, 16, 17, 18, 24, 26, 35, 38, 44, 45, 50, 72, 73, 99, 110, 127, 128, 129, 130, 131, 134, 141, 143, 156, 216

[73] N. Choungmo Fofack and Sara Alouf. Modeling modern DNS caches. In *Proc. ACM ValueTools'13*, Torino, Italy, Dec. 2013. 13, 127, 128, 130, 135

[74] N. Choungmo Fofack and Sara Alouf. Non-renewal TTL-based cache replacement policy and applications: Case of modern DNS hierarchy. Technical Report RR-8414, Inria, Sophia Antipolis, France, Dec. 2013. 101, 135

[75] A. N. Nakonechnyi. A refinement of the Rényi Theorem. *Cybernetics and Systems Analysis*, 29(6), 1993. 25, 26, 27

[76] K. Neammanee. A refinement of Normal approximation to Poisson Binomial. *International Journal of Mathematics and Mathematical Sciences*, 5:717–728, 2005. 23

[77] B. L. Nelson and I. Gerhardt. On the capturing dependence in point processes: Matching moments and other techniques. Working Paper, Jan. 2010. 67, 70, 133

[78] Nujira Ltd. State of the art RF power technology for defense systems. white paper, Feb. 2009. `http://www.nujira.com/_uploads/whitepapers/State_of_the_Art_RF_Power_Technology_for_Defence_Systems_EU.pdf`. 173

[79] P. Olivier and A. Simonian. Performance of a cache with random replacement and zipf document popularity. In *Proc. ACM ValueTools'13*, Torino, Italy, Dec. 2013. 8, 24

[80] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. On the responsiveness of dns-based network control. In *Proc. IMC'04*, Taormina, Sicily, Italy, Oct. 2004. 101, 102, 105

[81] A. D. Polyanin and A. V. Manzhirov. *Handbook of Integral Equations*. CRC Press., Boca Raton, first edition, 1998. 117

[82] E. Rosensweig, J. Kurose, and D. Towsley. Approximate models for general cache networks. In *Proc. IEEE INFOCOM'10*, San Diego, CA, USA, Mar. 2010. 8, 50, 66, 113, 127, 128, 129, 130, 131, 146, 156, 214

[83] E. J. Rosensweig, D. S. Menasche, and J. Kurose. On the steady-state of cache networks. In *Proc. IEEE INFOCOM'13*, Torino, Italy, Apr. 2013. 50, 60

[84] S. M. Ross. Average delay in queues with non-stationary poisson arrivals. *Journal of Applied Probability*, 15(3):602–609, 1978. 214

[85] A. Saltelli, S. Tarantola, and K. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, Feb. 1999. 190, 191

[86] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and M. Levy. An analysis of internet content delivery systems. *SIGOPS Operating System Review*, 36:315–327, 2002. 50, 66

[87] J. Seo, S. Lee, N. Park, H. Lee, and C. Cho. Performance analysis of sleep mode operation in IEEE 802.16e. In *Proc. of IEEE VTC 2004-Fall*, volume 2, pages 1169–1173, Los Angeles, CA, USA, Sept. 2004. 9, 174

[88] I. G. Shevtsova. An improvement of the convergence rate estimates in the lyapunov theorem. *Doklady Mathematics*, 82(3):862–864, 2010. 22

[89] A. D. Solov'ev. Asymptotic behavior of the arrival time of a rare event in a regenerative process. *Izv. Akad. Nauk SSSR, Tekhn. Kibern*, (6):79–89, 1971. 25

[90] X. Tang, J. Xu, and W. Lee. Analysis of TTL-based consistency in unstructured peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(12), Dec. 2008. 50

[91] K. Teerapabolarn. A non-uniform bound on poisson approximation for sums of bernoulli random variables with small mean. *Thai Journal of Mathematics*, 4(1):179–196, 2006. 22, 23

[92] C. Y. Teresa Lam and J. P. Lehoczky. Superposition of renewal processes. Technical Report TR-89-12, University of Michigan, Ann Arbor, MI 48109-2117, USA, oct 1990. 152

[93] M. Tortorella. Numerical solutions of renewal-type integral equations. *INFORMS Journal on Computing*, 17:73–96, 2005. 116

[94] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini. Temporal locality in today's content caching: Why it matters and how to model it. *ACM Computer Communication Review*, 43(5), Oct. 2013. 18, 209, 213

[95] W. Whitt. Approximating a point process by a renewal process, I: Two basic methods. *Operations Research*, 30(1), 1982. 11, 70, 105, 111, 112, 127, 132, 133, 145, 152, 155, 158, 202

[96] Y. Xiao. Performance analysis of an energy saving mechanism in the IEEE 802.16e wireless MAN. In *Proc. of IEEE CCNC 2006*, volume 1, pages 406–410, Las Vegas, Nevada, USA, Jan. 2006. 9, 174

[97] S. Yang and Y. Lin. Modeling UMTS discontinuous reception mechanism. *IEEE Transactions on Wireless Communications*, 4(1):312–319, Jan. 2005. 9, 174

[98] L. Zhou, H. Xu, H. Tian, Y. Gao, L. Du, and L. Chen. Performance analysis of power saving mechanism with adjustable DRX cycles in 3GPP LTE. In *IEEE VTC 2008-Fall*, Calgary, Alberta, Canada, Sept. 2008. 175