THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Distributed Coded Caching with Application to Content Delivery in Wireless Networks

Jesper Pedersen

CHALMERS
UNIVERSITY OF TECHNOLOGY

Communication Systems Group
Department of Electrical Engineering
Chalmers University of Technology
Göteborg, Sweden, 2021

Distributed Coded Caching with Application to Content Delivery in Wireless Networks

Jesper Pedersen

ISBN 978-91-7905-475-5
Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 4942
ISSN 0346-718X

This thesis has been prepared using LaTeX and PGF/Tik*Z*.

Communication Systems Group
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Phone: +46 (0)31 772 1000
`www.chalmers.se`

Printed by Chalmers Reproservice
Göteborg, Sweden, April 2021

To Lisa, Charlie, and Frans.

Abstract

The amount of content downloaded to mobile devices, mainly driven by the demand for video content, threatens to completely congest wireless networks and the trend of ever increasing video traffic is expected to continue unabated for many years. A promising solution to this problem is to store popular content closer to end users, effectively trading expensive bandwidth resources for affordable memory, a technique known as *caching*.

In this thesis, we study the use of erasure correcting codes (ECCs) to increase the amount of data that can be downloaded directly from the caches when content is cached in a distributed fashion across several base stations (BSs) or mobile devices. When content is cached in mobile devices, users may download coded packets directly from caching devices using device-to-device communication and, if necessary, from the BS at a higher communication cost. Devices moving out of range or turning off will cause a loss of cached content. To restore the initial state of reliability in the network, data is transmitted to available mobile devices in a process known as content repair. We compare the amount of data transmitted in the network due to content download and content repair using various ECCs when content is repaired at periodic times. We analyze the performance when mobile devices enter the network with or without usable cached content and show that increasing the time between repairs, so called lazy repairs, can be beneficial. Furthermore, we analyze content caching in mobile devices using maximum distance separable codes for scenarios where the density of devices is high. We optimize the number of mobile devices to involve in the caching of content and demonstrate the significant gains that can be achieved in terms of data downloaded from caching devices.

We proceed to analyze how to optimally manage cached content over time when users request content according to a renewal process, i.e., a process with memory. Specifically, we consider the distributed coded caching of content at small BSs where coded packets may be evicted from the caches at periodic times. We prove that the problem of maximizing the amount of data that users can download from the caches is concave and that our problem formulation is a generalization of the previously studied cases where content is cached in a single cache and where content is not managed over time, so called static caching. We show that optimizing caching policies can offer considerable gains in the amount of data that can be downloaded from the caches, especially

when the request process is bursty. Conversely, we prove that static caching is optimal for request processes without memory. Finally, we suggest a multi-agent reinforcement learning approach to learn cache management policies for spatially non-uniform renewal request processes. Our algorithm obtains cache management policies, substantially increasing the amount of data that can be downloaded from the caches.

Keywords: Caching, content delivery networks, device-to-device communication, erasure correcting codes, machine learning, optimization, time-to-live.

List of Publications

This thesis is based on the following publications:

[A] **J. Pedersen**, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Distributed Storage in Mobile Wireless Networks with Device-to-Device Communication". *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4862–4878, Nov. 2016.

[B] **J. Pedersen**, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Optimizing MDS Coded Caching in Wireless Networks With Device-to-Device Communication". *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 286–295, Jan. 2019.

[C] **J. Pedersen**, A. Graell i Amat, J. Goseling, F. Brännström, I. Andriyanova, and E. Rosnes, "Dynamic Coded Caching in Wireless Networks". *IEEE Trans. Commun. (Early Access).*

[D] **J. Pedersen**, A. Graell i Amat, F. Brännström, and E. Rosnes, "Dynamic Coded Caching in Wireless Networks Using Multi-Agent Reinforcement Learning". To be submitted to *IEEE J. Sel. Areas Commun.*, available on *arXiv*.

Other publications by the author, not included in this thesis, are:

[E] **J. Pedersen**, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Repair Scheduling in Wireless Distributed Storage with D2D Communication". *Proc. IEEE Inf. Theory Workshop (ITW)*, Jeju, Korea, Oct. 2015, pp. 69–73.

Acknowledgments

First, I would like to express my gratitude to my supervisor Prof. Alexandre Graell i Amat for offering me the chance to pursue a PhD. Thank you for your exceptional commitment to the research we have done over the past years, the invaluable discussions and feedback, and the meticulous proof reading of my drafts. You have taught me a great deal about what it means to do research. I would also like to express my gratitude to my co-supervisor Prof. Fredrik Brännström. Thank you for your guidance and support, and for your scrupulous attention to detail. I would further like to thank my co-supervisors Prof. Iryna Andriyanova and Dr. Eirik Rosnes for their guidance during the earlier and later stages of my doctoral education. I am also grateful to my co-author Prof. Jasper Gosling for offering illuminating discussions on optimization.

A big thanks goes to my colleagues in the Communications Systems group. It has been a pleasure to work with you these last couple of years. To my office mates Chao and Chouaib—thank you for putting up with my contemporary jazz. To my wonderful wife Lisa—love of my life, mother of our kids. Where would I be without you? This has been one crazy ride and it just appears we made it through to the other side. Thank you for your love, your support, and our endless talks. Thank you to my family, to my parents—for listening and giving good advice, to my sisters—for being there to talk, and to Kerstin, the best mother-in-law one can ever hope for—thanks for all the support with the kids, the dinners, and the excellent laundry service. Last, but definitely not least, to Staffan, Mylen, Mats, Robert, and Emelie—I treasure our friendship, thanks for being there for me.

Jesper Pedersen
Varberg, April 2021

This work was funded by the Swedish Research Council under grant 2016-4253.

Acronyms

5G	fifth generation
BB	branch-and-bound
BS	base station
CDF	cumulative distribution function
CDN	content distribution network
D2D	device-to-device
EB	exabyte
ECC	erasure correcting code
FIFO	first in first out
ILP	integer linear program
LFU	least frequently used
LP	linear program
LRU	least recently used
MARL	multi-agent reinforcement learning
MB	megabyte
MBR	minimum bandwidth regenerating
MBS	macro base station
MDP	Markov decision process
MDS	maximum distance separable
MILP	mixed-integer linear program
MIMO	multiple-input, multiple-output

ML machine learning

MSR minimum storage regenerating

NN neural network

PDF probability density function

PPP Poisson point process

RL reinforcement learning

RV random variable

RWP random waypoint

SBS small base station

TD temporal-difference

TTL time-to-live

Contents

Part I

Overview

CHAPTER 1

Introduction

Everywhere around us there are people using smartphones, perhaps watching the latest viral video on TikTok or Instagram, or perhaps listening to music or a podcast. Due to the proliferation of social media such as Facebook, YouTube, TikTok, and Instagram, streaming on-demand media providers like Netflix, Disney+, and HBO, and music/podcast platforms such as Spotify and Apple Music, the amount of content downloaded to smart devices in wireless networks is rapidly increasing [1]. We have grown accustomed to having access to these services wherever we are, whenever we want to, without significant latency or delays. It is also envisioned, with the rollout of the fifth generation (5G) cellular networks, that entirely new services, e.g., augmented or virtual reality, will be made available [2], which will increase the amount of downloaded data even more. As shown in Fig. 1.1, the amount of data downloaded to mobile devices globally is expected to reach 68 exabyte (EB) per month by the end of 2021 and to reach 226 EB per month by 2026, where the majority of the mobile data is video content [3].

In the old days, video was broadcasted to end-users, e.g., the live evening news transmission, where everyone would watch the transmission at the same time. Today, the demand for content is highly asynchronous, i.e., users wish

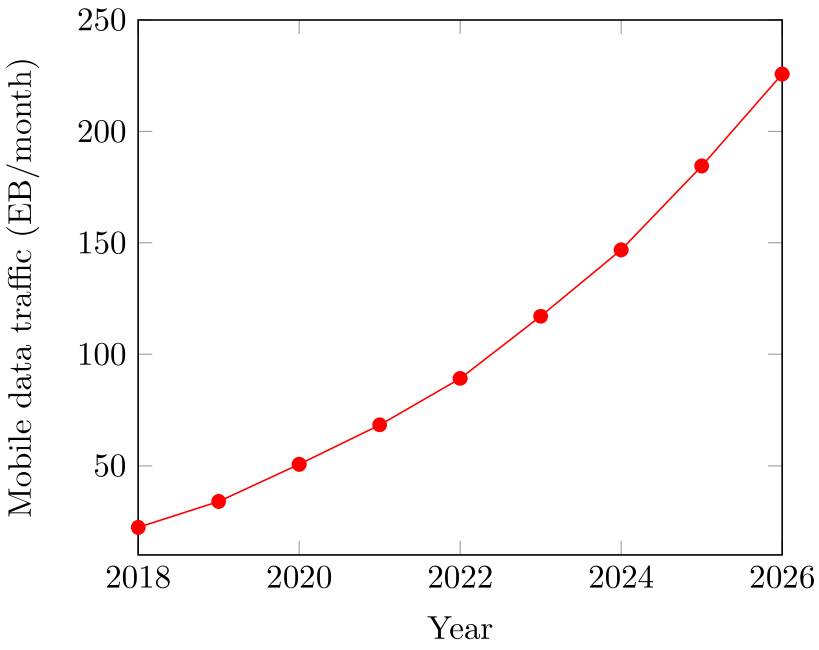


Figure 1.1: Past and expected global mobile data traffic per month.

to decide when they watch a particular video, not be restricted to watch at a particular time, which completely rules out broadcasting [4]. At the same time, requests for a few popular files are very frequent. Such file request characteristics put immense pressure on already burdened wireless links [5].

Several ideas to mitigate the problem of congested wireless links have been put forth, some of which are already included in the implementation of 5G cellular networks. For example, a densification of base stations (BSs) has been suggested with the deployment of so called small base stations (SBSs), where each SBS serves a smaller number of mobile users [6]. Furthermore, techniques to make new frequency spectrum available, so called millimeter wave frequencies, are being proposed [7], Finally, massive multiple-input, multiple-output (MIMO), i.e., drastically increasing the number of antennas, is envisioned to radically increase the capacity of wireless links [7]. It has, however, been contested whether these solutions alone will be enough to handle the deluge of mobile data traffic expected in the future [5].

There is another resource that remains largely untapped and that promises to alleviate the pressure on wireless links. The price of memory is, contrary to the trend of mobile data traffic, experiencing a decreasing trend, see Fig. 1.2

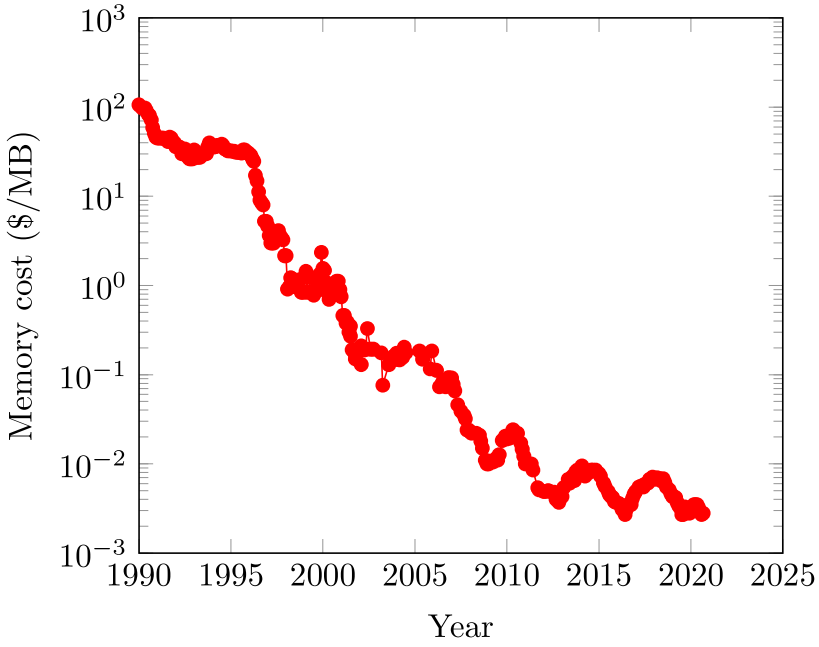


Figure 1.2: The price development of memory.

[8]. It has therefore been proposed to install hard drives at SBSs and utilize periods of low network traffic, e.g., during the night, to preallocate content that is expected to be frequently requested at the next period of high network load, e.g., the next day. Mobile devices can then download this content directly from the SBSs, thus reducing pressure on the backhaul. Alternatively, by the same argument of affordable and available memory resources, spare storage capacity on mobile devices can be utilized [7]. Content can then be downloaded from nearby caching devices using device-to-device (D2D) communication. Preallocating content closer to end users, effectively trading expensive bandwidth resources for inexpensive memory units, is known as *caching*.

The term cache was originally used to describe fast memory with small capacity in computer systems, where the memory would be used to provide swift access to frequently used data [9]. The idea was later extended to the internet where web pages would be replicated in a distributed fashion across servers (caches) to reduce network bandwidth usage, access time, as well as server congestion [9]. In the 1990s, internet traffic increased rapidly, which put more stringent requirements on the caches and led to the development of content distribution networks (CDNs) [9]. Recently, caching has also been

considered for improving content delivery in wireless networks [7], [9].

Irrespective of whether content is cached at SBSs or on mobile devices, it is expected that users will be in range of several SBSs, or encounter several caching devices over time, which means that content can potentially be downloaded from several sources [10]. However, it is unlikely that all caches will be accessible to the user requesting the content. Whenever a user can only download data from a subset of the caches, it is important to make sure that the requested content can be recovered from the downloaded data. If the content cannot be recovered, more data has to be downloaded over the backhaul. To this end, the use of erasure correcting codes (ECCs) has been shown to drastically reduce the amount of data that has to downloaded from the CDN over expensive backhaul [10], [11].

In this thesis, we investigate how much of the content requested by mobile users can be downloaded directly from caches located at SBSs or from caching mobile users, thereby alleviating traffic over backhaul links. We analyze what reductions in backhaul traffic can be achieved by using ECCs, where we optimize the code parameters to yield the maximum possible backhaul traffic reductions. When caching data on mobile devices and allowing users to download requested data from caching devices using D2D communication, the effects of device mobility have to be taken into account. In wireless networks exhibiting loss of data due to device mobility, the initial state of reliability has to be restored to continue to offer data offloading from servers, which is commonly referred to as data repair. We explore the tradeoff between data transmitted in wireless networks due to content download and content repair using various ECCs when repairs are carried out at periodic times, so called lazy repairs. We then proceed to analyze more realistic device mobility models, as well as user-centric communication protocols, and investigate what backhaul traffic reductions can be achieved for highly dense mobile device networks.

Due to how people interact socially, viral videos emerge and requests for this content arrive in bursts. When the request for a video increases the likelihood of another request in short succession, it is reasonable to cache more of the files that were recently requested. For such request processes, we explore ways to optimally manage cached content over time, when content is encoded and cached in a distributed fashion across several SBSs in the wireless network. We furthermore devise machine learning approaches to learn good

cache management policies without the need to know the file request statistics or the SBS distribution.

1.1 Thesis Organization

This thesis is organized as a *collection of papers*. The first part of the thesis is meant to introduce the models and tools used in the appended papers, which are found in the second part of the thesis.

The remainder of the introductory part of the thesis is organized as follows. In Chapter 2, we present the channel model that dictates how mobile devices and BSs can communicate in the wireless network and that is used throughout the thesis. Mobile device mobility models, relevant to Papers A and B, as well as distributions of devices and SBSs, mainly relevant to Papers C and D, are discussed. Several content request processes are presented. ECCs used for encoding cached content are introduced in Chapter 3, together with various cache replacement policies that dictate the management of cached content over time. Chapter 4 provides the reader with some background on optimization techniques that are used in Papers B and C. Finally, Chapter 5 provides an introduction to the branch of machine learning (ML) known as reinforcement learning (RL), the popular Q-learning algorithm, as well as policy gradient methods, relevant to Paper D.

1.2 Notation

The introductory part of the thesis uses the following notation conventions. A quantity proportional to another is denoted by $\propto$ and the symbol $\triangleq$ indicates a definition. We denote random variables (RVs) by capital letters, e.g., X and their realization by small letters, e.g., x. The symbol $\sim$ indicates the distribution of the random variable X, e.g., $X \sim \text{Exp}(\lambda)$ indicates that X is exponentially distributed with rate λ. The probability density function (PDF) and cumulative distribution function (CDF) of RV X is denoted by $f_X(x)$ and $F_X(x) \triangleq \Pr(X \leq x)$, respectively. The expected value of RV X is denoted by $\mathbb{E}[X]$. Furthermore, vectors are given in bold notation, e.g., $\boldsymbol{x}$, and sets are denoted by calligraphic letters, e.g., $\mathcal{B}$. Finally, $\mathbb{Z}$ denotes the set of integers.

CHAPTER 2

Wireless Network Models

A model is a mathematical construct to help analyze real-world phenomena. The model can be very complex to accurately capture situations in practice or less so to allow for a more tractable mathematical analysis. Usually, a trade-off has to be struck between complexity and accuracy. In this chapter, we describe the channel model used in the appended papers, dictating how the entities of the wireless network, i.e., the mobile devices, SBSs, and the macro base station (MBS), are allowed to connect and transmit data between them. We introduce the distributions of the locations of BSs and mobile devices, as well as several device mobility models. Finally, the user request processes used in the papers are explained.

2.1 Channel Model

In this thesis, we consider a path loss channel model, which describes the relationship between the transmitted and received signal power, P_{t} and P_{r}, respectively, and communication range r as follows [12, Ch. 2]

$$P_{\mathrm{r}} \propto P_{\mathrm{t}} r^{-2}. \tag{2.1}$$

We assume that communication can be carried out reliably, provided that the received signal power is high enough. For a fixed transmitted signal power, this corresponds to sufficiently small communication distances.

2.2 Base Station Distributions

When caching is performed at SBSs, we are concerned with how many SBSs, caching a particular piece of content, are within communication range of the user placing a request for that specific content. There are two popular models for describing the placement of SBSs from which the distribution of SBSs within communication range can be derived; uniformly random and grid-based placement.

Poisson Point Process

Consider an area A where mobile users can download cached content from SBSs within communication range r. The probability that an SBS is within communication range of some reference user is given by [13, Ch. 2]

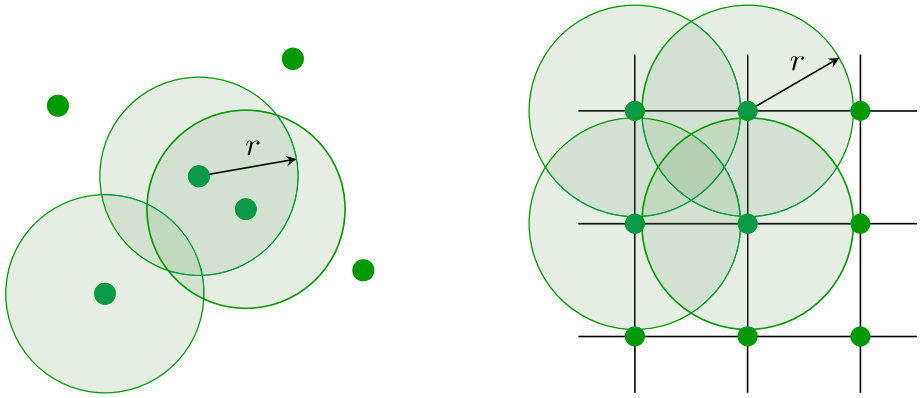
$$p \triangleq \frac{\pi r^2}{A}. \tag{2.2}$$

For n SBSs, independently and uniformly distributed in the area, the distribution of the number of SBSs within communication range, denoted by B, is given by the binomial distribution

$$f_B(b) = \binom{n}{b} p^b (1-p)^{n-b}, \tag{2.3}$$

with mean np [14, Ch. 2], and the SBS placement forms a binomial point process with *intensity* [13, Ch. 2]

$$\lambda = \frac{n}{A}. \tag{2.4}$$



(a) BSs distributed according to a PPP model.

(b) Grid-based placement of BSs.

Figure 2.1: An illustration of commonly used BS (dots) distribution models with communication range highlighted by opaque circles for some BSs.

When n tends to infinity and p tends to zero such that $\lambda\pi r^2$ is kept fixed, by the Poisson limit theorem [14, Ch. 2], (2.3) tends to the Poisson distribution

$$\gamma_b \triangleq f_B(b) = \frac{(\lambda\pi r^2)^b}{b!}\mathrm{e}^{-\lambda\pi r^2}, \tag{2.5}$$

with mean $\lambda\pi r^2$, which describes a Poisson point process (PPP) in the plane [13, Ch. 2]. BSs distributed according to a PPP model is shown in Fig. 2.1a. See, e.g., [15], [16] for applications of the PPP model to the study of wireless caching networks.

Grid-Based Model

An alternative to a random placement of SBSs is a deterministic placement, e.g., SBSs placed on a grid (see Fig. 2.1b) [10], [11]. Analyzing an isolated cell in this grid, the probability of the number of SBSs within communication range of a mobile user placing a request for a particular file can be derived. Finding the probability for a uniformly placed user is equivalent to finding the fraction of the area covered by a number of SBSs for a 1×1 square. For such a square area and $r = 1/\sqrt{2}$, the probabilities are readily obtained by

elementary geometry as follows,

$$f_B(0) = f_B(3) = f_B(4) = 0, \tag{2.6}$$
$$f_B(2) = 8(\pi r^2/8 - 1/8) = \pi/2 - 1, \tag{2.7}$$
$$f_B(1) = 1 - f_B(2) = 2 - \pi/2. \tag{2.8}$$

Hence, the expected number of SBSs within range of a uniformly distributed user is

$$\mathbb{E}[B] = \pi/2 \approx 1.57. \tag{2.9}$$

For the case $r = 1$, the probabilities can be straightforwardly obtained by integration,

$$f_B(0) = f_B(1) = 0, \tag{2.10}$$
$$f_B(2) = 8 \int_0^{1/2} 1 - \sqrt{1 - x^2}\, \mathrm{d}x = 4 - \frac{2\pi}{3} - \sqrt{3}, \tag{2.11}$$
$$f_B(4) = 4 \int_{1/2}^{\sqrt{3}/2} \sqrt{1 - x^2} - \frac{1}{2}\, \mathrm{d}x = 1 + \frac{\pi}{3} - \sqrt{3}, \tag{2.12}$$
$$f_B(3) = 1 - f_B(2) - f_B(4) = 2\sqrt{3} + \frac{\pi}{3} - 4, \tag{2.13}$$

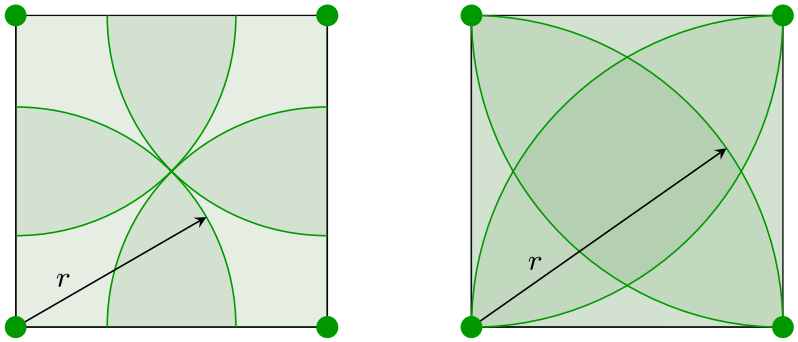
with the expected number

$$\mathbb{E}[B] = \pi \approx 3.14 \tag{2.14}$$

of SBSs within range. Obtaining the probability of the number of SBSs within communication range for other values of r may be handled analogously, or via Monte Carlo simulation. The coverage regions for communication range $r = 1/\sqrt{2}$ and $r = 1$ are shown in Fig. 2.2.

2.3 Device Mobility and Distributions

Mobile devices present in the wireless network are commonly assumed to be spread randomly in space, especially in conjunction with randomly placed SBSs as explained in Section 2.2 [10], [11]. There are, however, various models used to capture the mobility of devices. The application of such models is particularly interesting when spare memory capacity on the devices are used



(a) $r = 1/\sqrt{2}$ **(b)** $r = 1$

Figure 2.2: Coverage regions for the scenario where SBSs (dots) with communication range r are placed deterministically on a 1×1 grid.

to cache frequently requested content.

Birth-Death Process

The times between events are often modeled as exponentially distributed, e.g., packet arrivals to a network router, lightning strikes, or the birth of a giraffe in a park, and the counting of the number of occurrences is a Poisson counting process [14, Ch. 8]. When we are also concerned with events reducing the count in the system, e.g., customers leaving a store, packets forwarded from the network router, or the giraff in the park sadly dying, this is commonly referred to as a birth-death process [14, Ch. 9]. As the times between events are assumed to be exponentially distributed, this process is Markovian, i.e., the future is independent of the past [14, Ch. 9.4]. In the Markov nomenclature, the count in the system is referred to as the state. For arrival and departure rates denoted by λ_i and μ_i for state $i = 0, 1, 2, \ldots$, respectively, the steady-state distribution, i.e., the probability to be in a particular state m as time passes to infinity for the Markov process is [14, Th. 9.4]

$$\pi_m = \frac{\prod_{i=1}^{m} \frac{\lambda_{i-1}}{\mu_i}}{1 + \sum_{j=1}^{\infty} \prod_{i=1}^{j} \frac{\lambda_{i-1}}{\mu_i}}. \tag{2.15}$$

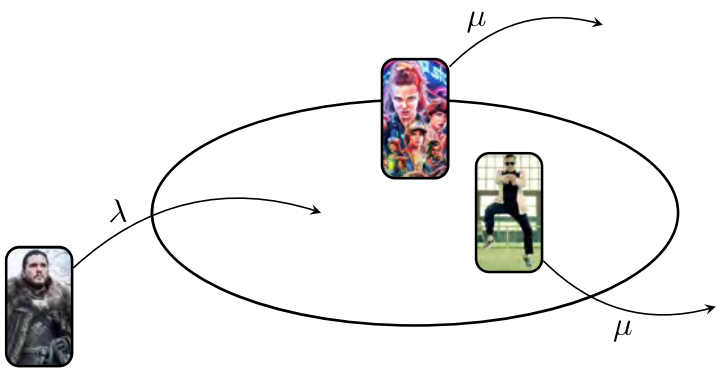


Figure 2.3: Mobile devices arriving to and departing from a cell at exponentially distributed random times with rates λ and μ, respectively.

Recently, the Markov birth-death process was used to study the device mobility in wireless networks [17].

Consider a cell in a wireless network where all devices in the cell can communicate using D2D communication. Assume that mobile devices arrive to the cell at exponentially distributed inter-arrival times with constant rate $\lambda_i = \lambda$, i.e., the PDF of the inter-arrivel times T is

$$f_T(t) = \lambda \mathrm{e}^{-\lambda t}. \tag{2.16}$$

Furthermore, assume that the devices stay in the cell for an exponentially distributed time with rate μ. The scenario with mobile devices arriving to and departing from the cell is depicted in Fig. 2.3. If, at any given time, there are i mobile devices in the cell, the aggregate departure rate from the cell is $\mu_i = i\mu$, by the additive property of the Poisson process [18, Section 2.3]. This describes an M/M/∞ queue, i.e., a system with Markovian arrivals and departures, as well as infinitely many servers [14, Ch. 9.4]. Since

$$1 + \sum_{j=1}^{\infty} \prod_{i=1}^{j} \frac{\lambda_{i-1}}{\mu_i} = 1 + \sum_{j=1}^{\infty} \prod_{i=1}^{j} \frac{\lambda}{i\mu} = \sum_{j=0}^{\infty} \frac{(\lambda/\mu)^j}{j!} = \mathrm{e}^{\lambda/\mu}, \tag{2.17}$$

and using (2.15), the steady-state distribution of the number of devices in the cell is

$$\pi_m = \frac{(\lambda/\mu)^m}{m!} \mathrm{e}^{-\lambda/\mu}, \tag{2.18}$$

which is recognized as the Poisson distribution (2.5).

Random Waypoint Model

Modeling device arrivals and departures to a cell in a wireless network as a birth-death process does not offer any granularity in the position of mobile devices within the cell. Furthermore, the model is not user-centric, i.e., dictated by a channel model where devices can communicate with other mobile devices within a specific communication range using D2D communication, as explained in Section 2.1. Therefore, an alternative model capturing device mobility is the random waypoint (RWP) model [19]. In this model, mobile devices

1. are placed randomly in an area,
2. pause for a fixed time,
3. pick a uniformly distributed target at random, and
4. go in a straight line towards the target at a speed chosen uniformly at random between a minimum and a maximum speed.

Once a device reaches its target, the process repeats from step 2 [19].

It has been shown that the spatial distribution of devices under the RWP model in its steady state is not uniform, due to the boundaries of the area [20]. However, in order to simplify the analysis, various techniques can be used to regain the uniform spatial distribution, e.g., consider a wrap-around, where devices departing the area reemerge on the other side, continuing their path, or wrapping the area around a sphere [21]. Mobile devices within communication range can download cached data using D2D communication. We refer to the time when two devices are in range as the *contact time*. We assume that the expected contact time is μ^{-1} but that the distribution is unknown. However, the *inter-contact time* between two devices, i.e., the time when the two devices are out of communication range of each other, has been shown to be exponentially distributed with rate λ [22]. See Fig. 2.4 for an illustration of the device mobility, the communication range, as well as the contact and inter-contact time. Provided that the area A is much larger than the circular area imposed by the communication range r, i.e., $A \gg \pi r^2$, and small differences in minimum and maximum device transverse speeds, the *inter-arrival time*,

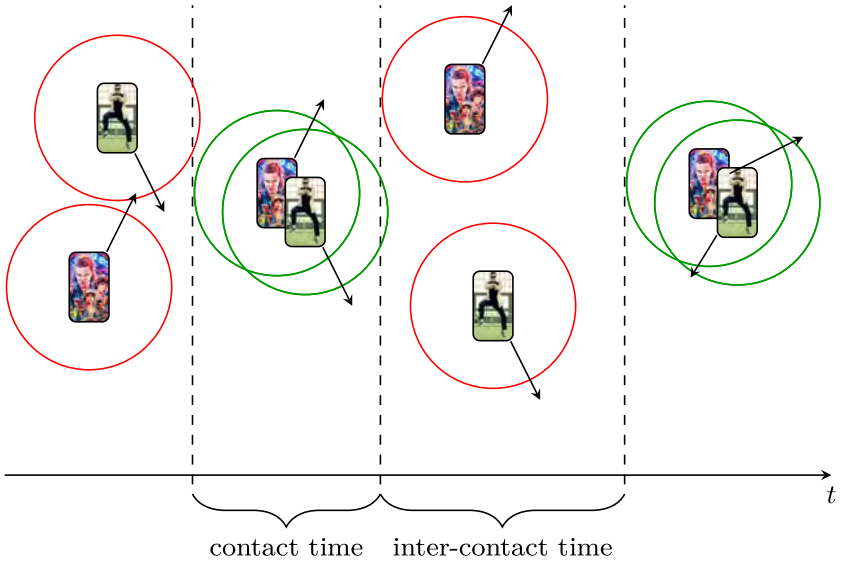


Figure 2.4: An illustration of the device mobility, the communication range, as well as the inter-contact time for two devices under the RWP model.

defined as the sum of the contact and inter-contact times, is approximately exponentially distributed with rate λ [23]. Hence, the number of mobile devices within communication range in steady state describes an M/G/∞ queue (where G marks a generalized departure or contact time). Under the assumptions of uniformly distributed devices in the area, the number of mobile devices within range is characterized by the Poisson distribution (2.18) with arrival and departure rates λ and μ, respectively [24].

2.4 Content Request Processes

Zipf's law is an empirical law that postulates that frequency and rank have an inverse relationship. Although the law was initially used to describe the frequency of words in relation to their rank in written text [25], the phenomenon is also expressed elsewhere, e.g., the frequency of mathematical expressions [26] and notes in music [27]. The law is also a widely accepted model to capture the popularity of web content [28], e.g., video files on YouTube [29]. Zipf's law states that, for a library of N files, the probability that file i is

requested is

$$p_i \triangleq \frac{1/i^\alpha}{\sum_{j=1}^{N} 1/j^\alpha}, \; i = 1, 2, \ldots, N, \; \alpha \geq 0 \tag{2.19}$$

where α is the skew parameter of the distribution [28]. Note that, for $\alpha = 0$, the Zipf distribution tends to the uniform distribution [14, Section 3.4].

Poisson Request Process

For exponentially distributed inter-request times, the number of requests in a certain time period is a Poisson process, as previously described in Section 2.3. Poisson processes have the appealing property that, if several processes are merged, they form another Poisson process, where the rate is the sum of the rates of the constituent processes [18, Section 2.3]. For example, assuming that the number of requests for file i is described by a Poisson process with rate

$$\omega_i \triangleq \omega p_i, \tag{2.20}$$

where ω is the aggregate request rate, merging the requests over all files yields another Poisson process with rate

$$\sum_{i=1}^{N} \omega_i = \omega \sum_{i=1}^{N} p_i = \omega. \tag{2.21}$$

To see this, let $X \sim \text{Poisson}(\omega_1)$, $Y \sim \text{Poisson}(\omega_2)$, $\omega = \omega_1 + \omega_2$, and $Z = X + Y$. Then

$$f_Z(z) = \sum_{x=-\infty}^{\infty} f_X(x) f_Y(z - x) \tag{2.22}$$

$$= \sum_{x=0}^{z} \frac{\omega_1^x}{x!} \mathrm{e}^{-\omega_1} \frac{\omega_2^{z-x}}{(z-x)!} \mathrm{e}^{-\omega_2} \tag{2.23}$$

$$= \frac{\mathrm{e}^{-\omega}}{z!} \sum_{x=0}^{z} \binom{z}{x} \omega_1^x \omega_2^{z-x} \tag{2.24}$$

$$= \frac{\omega^z}{z!} \mathrm{e}^{-\omega}, \tag{2.25}$$

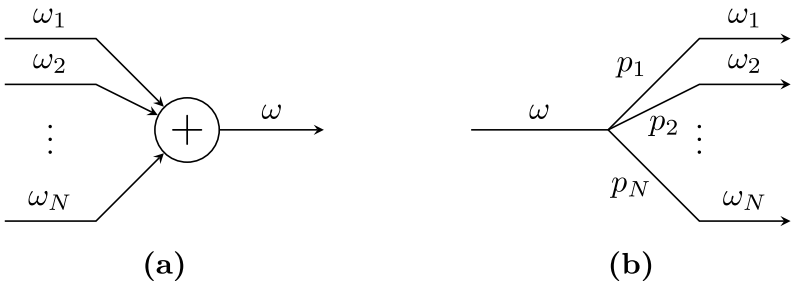


Figure 2.5: The merging (a) and splitting (b) of Poisson processes yields new Poisson processes.

where we have used the Binomial theorem [14, Section 2.7] in the last equality. Hence, $Z \sim \text{Poisson}(\omega)$ as claimed. Similarly, assume that files are requested according to a Poisson process with aggregate rate ω and a particular request is for file i with probability p_i, then file i is requested according to a Poisson process with rate ω_i [18, Section 2.3]. Hence, the interpretation of p_i as the popularity of file i, translates proportionally to the request rate for the same file. The merging and splitting of Poisson processes is illustrated in Fig. 2.5.

A positive random variable (RV) T is memoryless if [18, Section 2.2.1]

$$\Pr(T > t + x) = \Pr(T > t) \Pr(T > x), \text{ for all } x, t \geq 0. \tag{2.26}$$

If T is exponentially distributed with rate ω, this holds, as

$$\Pr(T > t + x) = 1 - \int_0^{t+x} \omega \mathrm{e}^{-\omega\tau} \, \mathrm{d}\tau = \mathrm{e}^{-\omega(t+x)} \tag{2.27}$$

$$= \mathrm{e}^{-\omega t}\mathrm{e}^{-\omega x} = \Pr(T > t) \Pr(T > x), \tag{2.28}$$

using (2.16). Conversely, an RV is memoryless only if it is exponential [18, Section 2.2.1]. Using the fact that T is exponential, we have

$$\Pr(T > t + x \mid T > x) = \Pr(T > t), \tag{2.29}$$

with the interpretation that, if T is the time until the next request for a file and there is no request for a time x, the distribution of the remaining time is the same as the original distribution, i.e., the process has no *memory* of what has occurred previously [18, Section 2.2.1].

Renewal Request Process

Renewal processes in general have memory and are used extensively to model situations where the event rate is changing over time, e.g., the failure rate of equipment should increase with its age or the mortality rate is higher among infants and elders. Due to how we spread information about new content, e.g., a video clip going viral, a renewal process is a reasonable choice for modeling requests for content [30]. For a renewal request process with inter-request times T, the *hazard function*

$$h(t) \triangleq \frac{f_T(t)}{\Pr(T > t)} \tag{2.30}$$

describes the probability to observe a request a time t after the previous request [31, Section 2.5], and a decreasing hazard function models a bursty request process. Weibull distributed inter-request times have been demonstrated to accurately capture the requests for educational video content [30]. The PDF of a Weibull-distributed inter-request time T is

$$f_T(t) = \frac{a}{b}\left(\frac{t}{b}\right)^{a-1} \mathrm{e}^{-(t/b)^a}, \tag{2.31}$$

where a and b are the shape and scale parameters of the distribution, respectively. Using (2.30), the hazard function for this distribution is

$$h(t) = \frac{a}{b}\left(\frac{t}{b}\right)^{a-1}. \tag{2.32}$$

We can see that, for $a < 1$, the hazard function is decreasing, i.e., this describes a bursty request process. Furthermore, for $a = 1$, T is exponentially distributed with rate $1/b$ and the hazard function is constant. Hence, the Poisson request process is a special case of this particular renewal process.

CHAPTER 3

Caching for Content Delivery

The idea behind caching is to estimate what content will be requested some time in the future and focus memory resources toward caching this content to minimize the amount of data that has to be fetched over congested backhaul links. The caches can for example be replenished over night, when network traffic is low, and users can download requested content directly from the caches during the day, when traffic is high. This requires accurate estimates of the file popularity profile as well as the request statistics (see Section 2.4). Alternatively, data may be cached in a more dynamic manner where a request for a file triggers it to be cached for a specific time, or in place of other content that is evicted from the cache. Irrespective of what method is used, the goal is that when users place a request for a file, the file is found and downloaded from the caches, known as a cache hit. If the requested content is not found in the caches, referred to as a cache miss, the data has to be fetched over the congested backhaul. Furthermore, the file popularity profile may be varying with time, i.e., the rate at which users request particular files may be time-dependent, which can be modeled as a renewal process (see Section 2.4). When faced with such scenarios, managing cached content over time can significantly reduce the amount of data that has to be fetched over the backhaul [32].

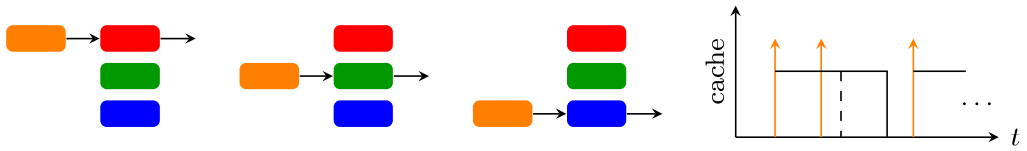
In wireless networks, users requesting files may be within communication range of several caching SBSs (see Section 2.2), or come in contact with a number of caching mobile devices (see Section 2.3). In this case, we can consider caching content across SBSs or mobile devices. However, it is likely that users can only download data from a subset of the caches, in which case it is important to ascertain that the requested content can be recovered from the downloaded data. If the requested content cannot be recovered, additional data has to be downloaded over the backhaul. To this end, the use of ECCs have been shown to drastically reduce the amount of data that has to be fetched over the congested backhaul, both when content is cached in a distributed fashion across several SBSs [10], [11], [33], or across a number of mobile devices [4], [17], [34], [35]. In the following, we describe a range of caching policies, dictating how to manage cached content over time, as well as the various ECCs considered in the appended papers.

3.1 Cache Replacement Policies

Popular choices for dynamic cache management include various types of cache replacement policies, e.g., least recently used (LRU), least frequently used (LFU), and first in first out (FIFO). Due to the difficulty in analyzing these caching policies [36], the more analytically tractable time-to-live (TTL) cache eviction policy has been suggested. It has been shown that the TTL policy has similar performance to LRU, LFU, and FIFO [37]–[40]. In this section, we give a brief description of how these policies operate.

Least Recently Used

The LRU cache replacement policy caches content together with a time stamp of when the content was cached. All requested content is cached until the cache is full. Furthermore, a request for cached content resets the time stamp. If there is a request for uncached content and the cache is full, the oldest cached content, i.e., the least recently requested content, is evicted from the cache, the recently requested content is cached, and the time stamp is reset. The operation of the LRU policy is illustrated in Fig. 3.1a.



(a) LRU **(b)** LFU **(c)** FIFO **(d)** TTL

Figure 3.1: The operation of the various cache replacement policies as requested orange content is being cached. In (a)–(c), the red content is the least recently requested, the green content is superseded by orange in number of requests, and the blue content is cached first. In (d), a request for orange content (marked with arrows) triggers it to be cached for a predetermined time.

Least Frequently Used

For the LFU policy, a counter of the number of requests for a particular content is maintained in the cache. Each request for a particular content increases its counter value. Similar to LRU, all requested content is initially cached, until the cache is full. If the number of requests for the recently requested content surpasses the cached content with the smallest counter value, i.e., the least frequently used content, this content is replaced. Fig. 3.1b shows how the LFU policy works.

First In First Out

The FIFO cache replacement policy simply tracks the order in which content was cached. If there is a request for uncached content, this content replaces the content first stored in the cache. We refer to Fig. 3.1c for an illustration of the FIFO policy operation.

Time-To-Live

The TTL cache eviction policy works in a significantly different manner than the LRU, LFU, and FIFO policies in that it is not a cache replacement that precipitates content eviction. The system designer decides on a time during which a particular content is to be cached a priori. A request for content triggers it to be cached and the timer is reset. If there is another request for the content before the expiration of the timer, the timer is simply reset. If the

timer expires before there is another request, the content is evicted from the cache (see Fig. 3.1d for an illustration). Whereas the LRU, LFU, and FIFO policies adhere to a strict cache size constraint, a long-term average cache size constraint is met for the TTL policy, which is the main reason for its analytical tractability. Although this could be seen as limiting the applicability of TTL policies, it has been demonstrated that, for a reasonably large file library, the instantaneous amount of used cache space does not deviate significantly from the average [40].

3.2 Erasure Correcting Codes

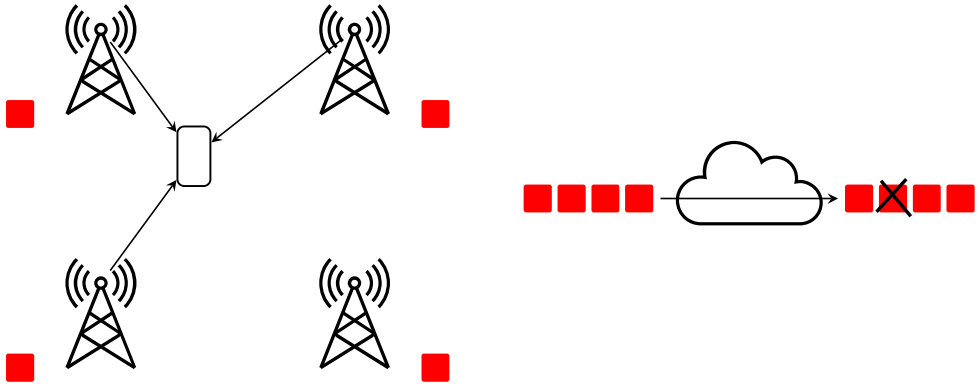
The use of ECCs, or simply codes, have been shown to significantly increase the amount of data that can be downloaded from the caches [10], [11]. In fact, the distributed coded caching scenario, shown in Fig. 3.2a, bears striking similarities with sending an encoded message over an unreliable (erasure) channel, where some packets are not received, illustrated in Fig. 3.2b. Furthermore, if cached data is lost due to, e.g., cache failure or caching devices departing from the network or running out of battery, the initial state of reliability in the caching network has to be restored, which is known as *repair*. The amount of data transmitted in a caching network due to repairs is referred to as the *repair bandwidth* and the number of caches involved in the repair is known as the *repair locality*. Repairs can be carried out either immediately or with a delay, known as lazy repairs, which can sometimes reduce the repair bandwidth [41].

When encoding data of size s, e.g., a frequently requested video file, using a code, it is first partitioned into k packets (or symbols) of size s/k, where k is referred to as the *code dimension*. The k symbols are then encoded into n symbols, also of size s/k, where n is the *code length*.[1] We specify a code using the dimension and length, i.e., we refer to it as an (n, k) code. The *rate* R of the code is the ratio between the code dimension and length, i.e.,

$$R \triangleq \frac{k}{n} \leq 1. \tag{3.1}$$

An important parameter for a code is the *Hamming distance*, or simply dis-

[1] We use the same letter n to describe both the code length and the number of SBSs since it is frequently the case that one coded symbol is stored per SBS.



(a) Distributed coded caching where *one* SBS is out-of-range of the user requesting the file.

(b) Transmission where *one* packet is erased by the channel.

Figure 3.2: A comparison between distributed coded caching and transmitting encoded data over an erasure channel. Both cases results in the loss of one coded packet.

tance. For codewords $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{C}$, where $\mathcal{C}$ is the set of all codewords, the distance, denoted by $d(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, is defined as the number of positions that $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ differ. Another metric, directly determining the erasure correcting capability of the code, is the *minimum distance* [42, Sec. 3.1.3]

$$d_{\min} = \min_{\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{C}: \boldsymbol{x} \neq \tilde{\boldsymbol{x}}} d(\boldsymbol{x}, \tilde{\boldsymbol{x}}). \tag{3.2}$$

Since any two codewords differ in at least $d_{\min}$ positions, the code can correct up to

$$d_{\min} - 1 \tag{3.3}$$

erasures. For a linear code, the Singleton bound states that [43]

$$d_{\min} \leq n - k + 1. \tag{3.4}$$

Maximum Distance Separable Codes

A maximum distance separable (MDS) code is optimal in the sense that it attains the Singleton bound (3.4). Hence, using (3.3), the code can recover

from any

$$d_{\min} - 1 = n - k \tag{3.5}$$

erasures and any k symbols suffice to recover the original data. Therefore, the total amount of data retrieved to recover the file is $k \cdot s/k = s$, the size of the file. For MDS codes, when considering repairing a cached coded symbol, e.g., when a caching device departs from the network, this is unfortunately also the amount of data that has to be recovered and re-encoded to restore the initial state of reliability in the network, i.e., the repair bandwidth is s. Furthermore, the decoding complexity of MDS codes, e.g., Reed-Solomon codes [44], increases drastically with the code length n [45]. A trivial example of an MDS code is the $(n, 1)$ repetition code.

Regenerating Codes

Addressing the poor performance of MDS codes in terms of repair bandwidth, regenerating codes were proposed in [46]. These codes, based on network coding principles, achieve the best trade-off between repair bandwidth and amount of data stored in each cache, shown in Fig. 3.3. Codes achieving the minimum occupied cache space, in fact the same cache memory occupancy as an MDS code, are called minimum storage regenerating (MSR) codes and codes attaining the minimum repair bandwidth are referred to as minimum bandwidth regenerating (MBR) codes [46]. Both MSR and MBR codes are able to reduce the repair bandwidth by increasing the repair locality which, in turn, makes them vulnerable to multiple cache failures or caching device departures [47]. Furthermore, a drawback of using MBR codes, although they are very efficient in terms of minimizing the repair bandwidth, is that they increase the amount of data that has to be downloaded to recover a requested file.

Locally Repairable Codes

Finally, a limitation of both MDS codes and regenerating codes is the large repair locality, e.g., using MDS codes, k symbols have to be retrieved to perform repair. To address this issue, locally repairable codes divide a codeword into *repair groups*, where the repair locality within each group is small [48]. Furthermore, the repair bandwidth is reduced as compared to that of MDS codes. However, if there is more than a single erasure within a repair group,

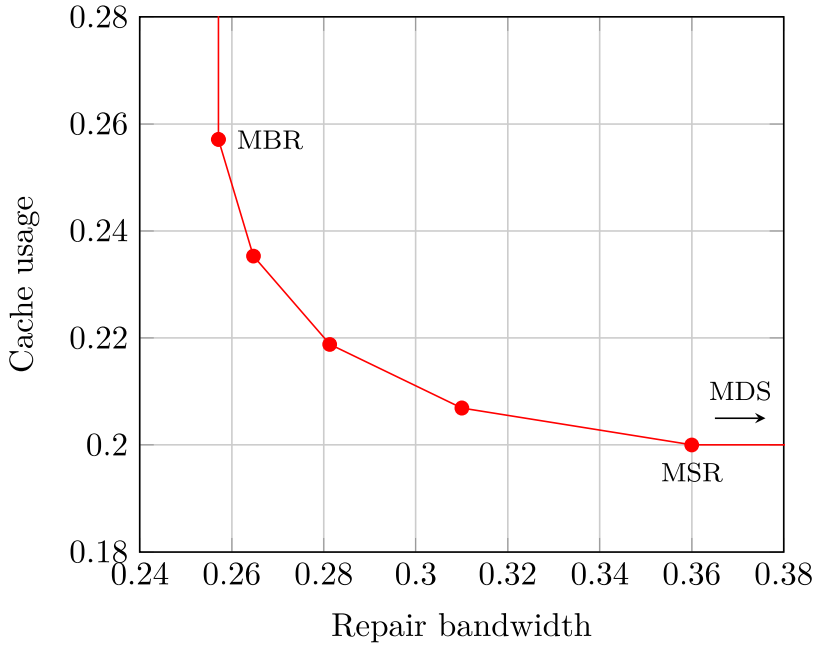


Figure 3.3: The repair bandwidth versus cache usage trade-off for regenerating codes, assuming code length $n = 90$, 10 caches, code dimension $k = 5$, file size $s = 1$, and repair locality 9. MDS codes achieve the point $(1, 0.2)$, which is indicated by the arrow.

the whole codeword has to be decoded and re-encoded, resulting in a repair bandwidth larger than that for an MDS code.

CHAPTER 4

Optimizing Content Allocations

When we are considering distributed coded caching in wireless networks, we are concerned with allocating coded content to the caches to maximize the amount of requested data that can be downloaded from the caches, i.e., is offloaded from the backhaul link. Assume that we wish cache files from a file library with N files, where, without loss of generality, we assume that each file is of size $s = 1$. Furthermore, assume that we may cache content in n SBSs or mobile devices, each with limited cache size C. In the following, we will consider data cached at SBSs, although all results are equally valid for the scenario of caching in mobile devices. Each file i to be cached is partitioned into k_i packets and encoded into n coded packets using an ECC. Specifically, we will consider encoding the files using an MDS code as an illustrative example (see Section 3.2). Each SBS caches one (unique) coded packet and hence cache an amount

$$x_i \triangleq \frac{1}{k_i} \tag{4.1}$$

of file i. Note that we modify the code dimension k_i to vary the amount of file i cached at each SBS. Using (3.1), the code rate for file i is

$$R_i \triangleq \frac{k_i}{n}. \tag{4.2}$$

Consequently, for fixed n, modifying the code rate R_i controls the amount of file i cached at each SBS. Due to the limited cache size C, we have the following cache size constraint,

$$\sum_{i=1}^{N} x_i \leq C. \tag{4.3}$$

Recall that, for an MDS code, any k_i coded packets suffice to decode file i (see Section 3.2). Hence, it is necessary and sufficient that a user downloads an amount

$$k_i x_i = 1 \tag{4.4}$$

to recover file i. A user requesting file i and within communication range of b SBSs therefore downloads an amount

$$\min\{1, bx_i\}, \tag{4.5}$$

from the caches, where the $\min\{\cdot,\cdot\}$ function arises due to (4.4). The average amount of content downloaded from the SBSs is

$$\sum_{i=1}^{N} p_i \sum_{b=0}^{n} \gamma_b \min\{1, bx_i\}, \tag{4.6}$$

where $\gamma_b \triangleq f_B(b)$ is the probability that a user is within communication range of b SBSs (see Section 2.2) and p_i is the probability that the user requests file i (see Section 2.4). Maximizing the average amount of content downloaded from

the caches corresponds to the following constrained optimization problem

$$\underset{x_i}{\text{maximize}} \sum_{i=1}^{N} p_i \sum_{b=0}^{n} \gamma_b \min\{1, bx_i\}, \tag{4.7}$$

$$\text{subject to} \sum_{i=1}^{N} x_i \leq C, \tag{4.8}$$

$$x_i \in \left\{0, \frac{1}{n}, \frac{1}{n-1}, \ldots, 1\right\}, \ i = 1, 2, \ldots, N. \tag{4.9}$$

Variations of this problem, recognized as the nonlinear resource allocation problem [49], arise frequently in the analysis of distributed coded caching networks (see, e.g., [10], [11]).

Integer Relaxation

The main difficulty in solving the problem (4.7)–(4.9) lies in the fact that x_i can only take on a discrete set of values, due to $k_i \in \mathbb{Z}$. This is most commonly addressed by simply relaxing (4.9) to

$$0 \leq x_i \leq 1, \ i = 1, 2, \ldots, N, \tag{4.10}$$

in which case (4.7) yields an upper bound on the average amount of data downloaded from the caches. By realizing that the sum of min functions is a concave function [50, Section 3.2.1], the problem is efficiently solved using a range of applicable numerical solvers.

Epigraph Formulation

The problem (4.7), (4.8), and (4.10) can be simplified further. By making use of the epigraph formulation [50, Section 3.1.7], an equivalent problem is

$$\underset{x_i, t_{i,b}}{\text{maximize}} \sum_{i=1}^{N} p_i \sum_{b=0}^{n} \gamma_b t_{i,b}, \tag{4.11}$$

$$\text{subject to} \sum_{i=1}^{N} x_i \leq C, \tag{4.12}$$

$$0 \leq x_i \leq 1,\ i = 1, 2, \ldots, N, \tag{4.13}$$

$$t_{i,b} \leq 1,\ i = 1, 2, \ldots, N,\ b = 0, 1, \ldots, n, \tag{4.14}$$

$$t_{i,b} \leq b x_i,\ i = 1, 2, \ldots, N,\ b = 0, 1, \ldots, n, \tag{4.15}$$

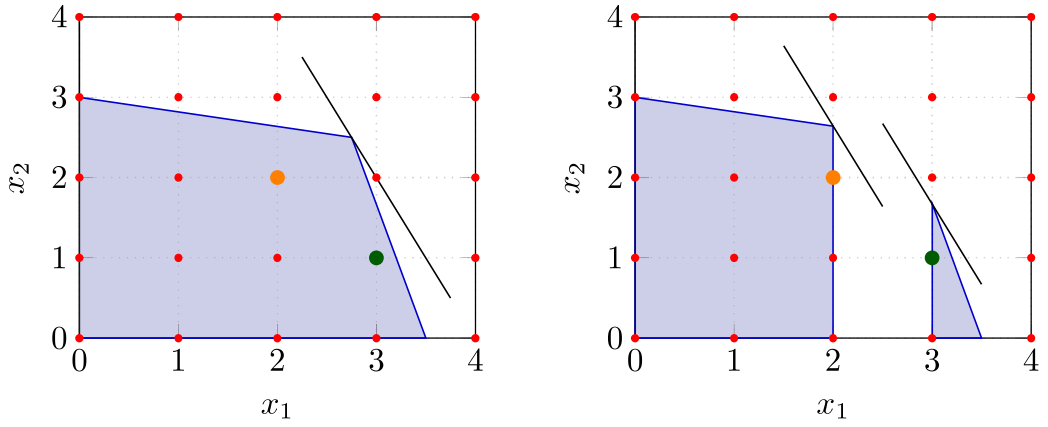
which we identify as a linear program (LP). Although there is no analytical solution to LPs in general, very efficient algorithms exists to solve such problems numerically, e.g., the simplex method [51].

4.1 Mixed-Integer Linear Programs

Applying the epigraph formulation directly to the problem (4.7)–(4.9) yields a mixed-integer linear program (MILP). Although such problems are known to be NP-complete in general, algorithms exist, e.g., branch-and-bound (BB), that frequently reduce the time required to solve MILPs as compared with exhaustive search, despite the lack of guarantees for such performance.

Branch-and-Bound

The BB algorithm, invented in the 1960s [52], is a commonly used algorithm to solve NP-hard optimization problems. The algorithm recursively splits the search space into smaller spaces, called *branches*, thereby creating new optimization problems. A feasible solution to the original problem is stored and the best known solution is referred to as the *incumbent*. As a better feasible solution is found, the incumbent is replaced. The algorithm attempts to find *bounds* precluding the branch from containing the solution, e.g., an upper bound on the objective function which is smaller than the incumbent



(a) ILP **(b)** P1 and P2

Figure 4.1: The BB algorithm applied to solve an ILP. The blue area indicates the search space, red dots the feasible values for the ILP, and black lines are level curves for the objective function. The orange and green points show the incumbent from P1 and the solution to the ILP, respectively.

for a maximization problem. If such bounds can be found, the branch is cut. The algorithm proceeds until there is a single remaining branch, whereby the incumbent is returned as the solution to the optimization problem.

The BB algorithm is best illustrated through an example. Consider the integer linear program (ILP)

$$\underset{x_1,x_2}{\text{maximize}}\ 2x_1 + x_2 \tag{4.16}$$

$$\text{subject to}\ 2x_1 + 11x_2 \leq 33, \tag{4.17}$$

$$10x_1 + 3x_2 \leq 35, \tag{4.18}$$

$$x_1, x_2 \in \mathbb{Z}, \tag{4.19}$$

shown in Fig. 4.1a. The integer relaxed version of the ILP, i.e., (4.16)–(4.18) is optimal for $(x_1, x_2) = (2.75, 2.5)$. However, since this point clearly violates the integer constraint, the problem is split (branched) into two new problems, one for which $x_1 \leq 2$ and one for which $x_1 \geq 3$. We refer to the first subproblem

as P1,

$$\underset{x_1,x_2}{\text{maximize}}\ 2x_1 + x_2 \tag{4.20}$$

$$\text{subject to } 2x_1 + 11x_2 \leq 33, \tag{4.21}$$

$$x_1 \leq 2, \tag{4.22}$$

$$x_1, x_2 \geq 0, \tag{4.23}$$

maximized for $(x_1, x_2) = (2, 2.64)$, and the other as P2,

$$\underset{x_1,x_2}{\text{maximize}}\ 2x_1 + x_2 \tag{4.24}$$

$$\text{subject to } 10x_1 + 3x_2 \leq 35, \tag{4.25}$$

$$x_1 \geq 3, \tag{4.26}$$

$$x_2 \geq 0, \tag{4.27}$$

maximized for $(x_1, x_2) = (3, 1.67)$. The branching into problems P1 and P2 is illustrated in Fig. 4.1b. P1 is branched into $x_2 \leq 2$ (P11) and $x_2 \geq 3$ (P12), where P12 attains the value 3, stored as the incumbent, for $(x_1, x_2) = (0, 3)$. Since this is the best integer solution found, it is the new incumbent and a lower bound on the optimal solution to the ILP. The subproblem P11 attains the optimal value 6, the new incumbent, for the point $(x_1, x_2) = (2, 2)$ and the branch P12 is cut. Continuing to branch P2 in a similar manner, it is readily found that the point $(x_1, x_2) = (3, 1)$, attaining the optimal value of 7, is the new incumbent. The P1 branch is cut as the value of the objective function is upper bounded by 6. Hence, the solution related to P2 is the global optimum as there are no subproblems left to solve. The branching process is illustrated in Fig. 4.2.

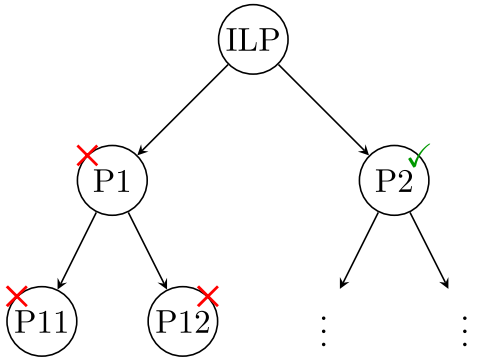


Figure 4.2: The BB algorithm for the example ILP. The suboptimal problems P11 and P12, and consequently P1, are cut, which is indicated by red crosses. P2 is optimal which is highlighted by the green checkmark.

CHAPTER 5

Reinforcement Learning

ML can be broadly defined as the study of algorithms that improve through experience. Together with supervised and unsupervised learning, RL is one of the main approaches to ML. The idea behind RL is to map a situation, or state, to an action in order to maximize a reward. The states, actions, and rewards can be formally defined through a Markov decision process (MDP).

5.1 Markov Decision Process

An MDP is a mathematical framework used to model sequential decision making. In this framework, we refer to the entity making the decisions as the *agent*, which interacts with an *environment* at discrete times $t = 0, 1, 2, \ldots$. The environment provides a state representation $S_t \in \mathcal{S}$ to the agent and the agent decides on an action $A_t \in \mathcal{A}$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions, also referred to as state and action space, respectively. In response to the action, the environment responds with a new state $S_{t+1} \in \mathcal{S}$ and a reward $R_{t+1} \in \mathcal{R}$, where $\mathcal{R}$ is the set of rewards. The agent decides on a new action $A_{t+1} \in \mathcal{A}$ and the process continues. An illustration of the interaction between the agent and the environment is shown in Fig. 5.1. The dynamics

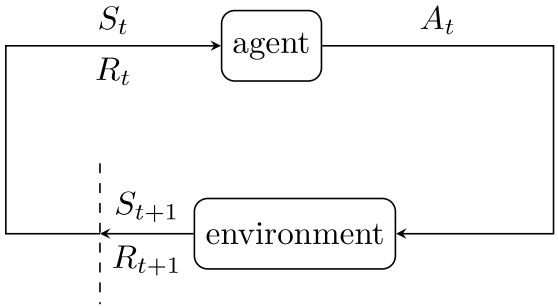


Figure 5.1: An illustration of the interaction between the agent and the environment in the MDP framework.

of the MDP are defined by

$$p(s', r|s, a) \triangleq \Pr(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a) \tag{5.1}$$

with state-transition probabilities

$$p(s'|s, a) = \sum_{r \in \mathcal{R}} p(s', r|s, a). \tag{5.2}$$

The goal of the agent is to maximize long-term rewards. Specifically, the agent strives to maximize the expected *discounted return*

$$\mathbb{E}[G_t] \triangleq \mathbb{E}\left[\sum_{\ell=0}^{\infty} \gamma^\ell R_{t+\ell+1}\right], \tag{5.3}$$

where γ is the discount rate. A *policy* $\pi(a|s)$ is a mapping from a state s to an action a, i.e.,

$$\pi(a|s) \triangleq \Pr(A_t = a|S_t = s). \tag{5.4}$$

For a policy π, the *value function*

$$v_\pi(s) \triangleq \mathbb{E}_\pi[G_t|S_t = s] \tag{5.5}$$

describes the expected discounted return that would be obtained if the agent is in state s and follows policy π from that state. Similarly, the *action-value function*

$$q_\pi(a, s) \triangleq \mathbb{E}_\pi[G_t|A_t = a, S_t = s] \tag{5.6}$$

is the expected discounted return when being in state s, taking action a, and following policy π afterward. An optimal policy π_* provides at least as much expected discounted return as other policies π, for all states, i.e.,

$$v_{\pi_*}(s) \geq v_\pi(s), \text{ for all } s \in \mathcal{S}. \tag{5.7}$$

We define the optimal value and action-value functions by

$$v_*(s) \triangleq \max_\pi v_\pi(s) \tag{5.8}$$

and

$$q_*(a, s) \triangleq \max_\pi q_\pi(a, s), \tag{5.9}$$

respectively. The optimal value and action-value functions abide by a recursive relationship, called the Bellman optimality equation. This relationship states that, for an optimal value function

$$v_*(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*\left(S_{t+1}\right) | A_t = a, S_t = s\right] \tag{5.10}$$

and for an optimal action-value function [53]

$$q_*(s, a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*\left(S_{t+1}, a'\right) | A_t = a, S_t = s\right]. \tag{5.11}$$

Using (5.10), an optimal policy π_* is readily obtained once the optimal value function v_* is known. Similarly, knowing the optimal action-value function q_* immediately gives an optimal policy using (5.11). Several algorithms exist to find optimal policies, e.g., dynamic programming, Monte Carlo methods, and temporal-difference (TD) learning [53]. Here, we will focus on a widely used TD control algorithm known as Q-learning [54].

5.2 Q-Learning

Q-learning is a model-free, off-policy algorithm, i.e., it does not require a model of the environment (model-free) and the learned (target) policy is separated from the (behavior) policy used to take actions (off-policy). The reason for maintaining two policies is that we are trying to learn the action-value function, conditioned on a subsequent optimal behavior, while we somehow have

to *find* this optimal behavior, i.e., we have to *explore* the environment [53]. A popular implementation is the ϵ-greedy method which dictates that, with probability ϵ, an action is chosen uniformly at random, and with probability $1 - \epsilon$, the optimal action is chosen, i.e.,

$$\arg\max_a q(s, a). \tag{5.12}$$

For the chosen action $A_t = a$ and observed next state $S_{t+1} = s'$ with corresponding reward $R_{t+1} = r$, the Q-learning update rule is

$$q(s, a) \longleftarrow q(s, a) + \alpha \left[r + \gamma \max_a q(s', a) - q(s, a) \right], \tag{5.13}$$

where $\alpha \in (0, 1]$ is the step size. The update rule bears striking similarities with (5.11) without the expectation. Hence, the Q-learning update attempts to modify the action-value function to come closer and closer to satisfying the Bellman optimality equation. Indeed, q converges to q_* with probability 1 under sufficient exploration and small enough step size [53].

5.3 Policy Gradient Methods

For finite MDPs, i.e., the state and action spaces have a finite number of elements, the action-value function estimates $q(s, a)$ can simply be maintained in a table. If the state and/or action spaces are not finite, e.g., when the state or action space, or both, are continuous, this is no longer possible. Furthermore, the optimization

$$\max_a q(s', a) \tag{5.14}$$

in (5.13) is too costly, considering that it would have to be computed for each step the agent takes in the environment. Hence, we have to resort to other methods. One solution to this problem is to consider learning a parameterized policy

$$\pi_{\boldsymbol{\theta}}(a|s) \triangleq \Pr(A_t = a | S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}), \tag{5.15}$$

which is assumed to be differentiable in the parameter vector $\boldsymbol{\theta}$ for all $a \in \mathcal{A}$ and $s \in \mathcal{S}$. If the performance is measured by some function $J(\boldsymbol{\theta})$, a policy gradient method is attempting to maximize the performance by modifying the

vector $\boldsymbol{\theta}$ in the direction of the gradient, i.e., by gradient ascent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta}_t). \tag{5.16}$$

We can also parameterize a value function, e.g., the action-value function q. Regardless whether we are trying to learn a value function or not, we will refer to the method as policy gradient. However, more frequently, they are called *actor-critic methods* where the actor describes the learned policy and the critic is the learned value function [53].

Deep Deterministic Policy Gradient

A policy gradient method that is closely related to Q-learning is the deep deterministic policy gradient algorithm, which can handle continuous state and action spaces [55]. Using this algorithm, we assume a deterministic policy, denoted by $\pi_{\boldsymbol{\theta}}(s)$, where the parameter vector $\boldsymbol{\theta}$ corresponds to a collection of weights of a neural network (NN), referred to as the *actor network*. The goal is to learn the vector $\boldsymbol{\theta}$ such that the policy approximates well the optimal action, i.e.,

$$\pi_{\boldsymbol{\theta}}(s) \approx \arg\max_a q(s, a). \tag{5.17}$$

The deterministic policy hence outputs an action for a given state. Note that this distinguishes from the policy in (5.15) which provides a probability distribution over actions. Furthermore, assuming that the action-value function $q(s, a)$ is differentiable in the continuous actions a, we can avoid the costly operation (5.14) by using the approximation

$$\max_a q(s, a) \approx q(s, \pi_{\boldsymbol{\theta}}(s)). \tag{5.18}$$

The parameterized action-value function, denoted by $q_{\boldsymbol{\phi}}(s, a)$, is obtained in a similar fashion, where $\boldsymbol{\phi}$ are the weights of a NN, known as the *critic network*. The aim is to learn the vector $\boldsymbol{\phi}$ such that

$$q_{\boldsymbol{\phi}}(s, a) \approx q(s, a). \tag{5.19}$$

Apart from the aforementioned networks, we maintain two additional NNs in order to stabilize the algorithm; a *target actor network* with weights $\hat{\boldsymbol{\theta}}$ and a *target critic network* with weights $\hat{\boldsymbol{\phi}}$.

The action is

$$a = \pi_{\boldsymbol{\theta}}(s) + \epsilon, \tag{5.20}$$

where ϵ is Gaussian distributed with zero mean and variance σ^2 and controls the amount of exploration. The next state s' and the reward r is observed and the tuple (s, a, s', r) is stored in a set $\mathcal{M}$, referred to as the *memory replay buffer*. The update of the action-value function is similar to the Q-learning update (5.13). Specifically, the weights of the critic network are updated by stochastic gradient descent

$$\nabla_{\boldsymbol{\phi}} \frac{1}{|\mathcal{B}|} \sum_{(s,a,s',r)\in\mathcal{B}} \left(q_{\boldsymbol{\phi}}(s, a) - r - \gamma q_{\hat{\boldsymbol{\phi}}} \left(s', \pi_{\hat{\boldsymbol{\theta}}}(s') \right) \right)^2, \tag{5.21}$$

where $\mathcal{B}$ is a set of tuples sampled randomly from $\mathcal{M}$. Following (5.18), the weights of the actor network are updated by gradient ascent

$$\nabla_{\boldsymbol{\theta}} \frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}} q_{\boldsymbol{\phi}}(s, \pi_{\boldsymbol{\theta}}(s)). \tag{5.22}$$

Finally, the weights of the target networks are updated

$$\hat{\boldsymbol{\phi}} \longleftarrow \rho\hat{\boldsymbol{\phi}} + (1 - \rho)\boldsymbol{\phi}, \tag{5.23}$$

$$\hat{\boldsymbol{\theta}} \longleftarrow \rho\hat{\boldsymbol{\theta}} + (1 - \rho)\boldsymbol{\theta}, \tag{5.24}$$

where ρ is a parameter close to 1.

CHAPTER 6

Summary

In this chapter, we outline the contributions and conclude the findings of the appended papers. Furthermore, we give an outlook on possible future research directions.

6.1 Contributions

In this thesis, we study the caching of content in wireless networks. Specifically, we assume that content is cached across SBSs and mobile devices in order to minimize the amount of data that has to be downloaded over a congested backhaul link. As users requesting content are assumed to access a subset of the caches, we consider encoding the content using ECCs. In general, we investigate the impacts on backhaul traffic of data loss due to caching devices departing the wireless network and consequent data repair. We furthermore evaluate the backhaul traffic reductions in highly dense D2D networks. Finally, we analyze how to optimally manage the distributed caching of coded content over time. In the following, we summarize the contributions of the papers that are appended in Part II of the thesis.

Paper A

In this paper, we study the caching of coded content in mobile devices that join and leave a cell in a wireless network according to a stochastic process. Specifically, we assume that devices arrive and depart from the network according to a Poisson birth-death process. We utilize spare memory capacity on the mobile devices to cache coded packets of a frequently requested file. Users in the network request the file according to a Poisson request process and download coded packets from caching devices in the network using D2D communication or from the BS, at the expense of a higher communication cost. As caching devices depart the cell, we have to restore the initial state of reliability in the network to be able to continue to serve users requesting the file. The goal is to minimize the *network load*, defined as the weighted sum of data transmitted from the BS and caching devices due to file downloads and data repair. The main novelty of this work is the analysis of a scheme performing repairs at periodic times. Several erasure correcting codes are evaluated to find the ones minimizing the network load, depending on the repair period. We compare the network performance when devices arriving to the cell have empty caches or carry coded packets, such that the devices can immediately participate in the caching network. When mobile devices arriving to the cell already cache coded packets, the network load is reduced compared to when devices arrive with empty caches. Furthermore, we analyze the performance as the BS and caching devices collaborate to deliver and repair content. We show that coded caching can reduce the network load, depending on the repair period. We demonstrate that MDS codes are optimal when repairs are not considered, and regenerating codes can offer significant reductions in terms of network load, provided that repairs are performed frequently enough. Finally, we show that non-instantaneous, so called lazy, repairs are sometimes desirable.

Paper B

In Paper B, we extend Paper A by considering a more practical device mobility and communication model, where the D2D communication is user-centric, i.e., mobile devices within a certain distance can communicate. We prove that, for an area without boundaries, the steady-state distribution of the number of devices within communication range is well approximated by the

Poisson distribution. The result is interesting because it provides an analytical framework that is tractable, despite the assumption of more realistic mobility and communication models. Furthermore, we consider a library of files with different popularity, i.e., probability to be requested. As content repair is not considered in this paper, we assume that content is encoded using MDS codes, which was shown to minimize the communication cost for such scenarios in Paper A. We reformulate the optimization of code rates to minimize the communication cost into a form that is tractable for numerical solvers. We study how to spread coded content in the wireless network, i.e., how many devices should be involved in the caching of a particular file. We develop bounds on the communication cost that facilitate the analysis of very large and dense networks. Finally, we show that coded caching in mobile devices offers significant gains as compared to uncoded or no caching.

Paper C

The request processes in Papers A and B were all assumed to be memoryless, i.e., we assumed exponentially distributed inter-request times. In Paper C, we lift this assumption and consider request processes with memory, i.e., we model requests as a renewal process. For such request processes, we investigate how to manage coded content, cached in a distributed fashion across SBSs, optimally over time. In particular, we build on the work by Goseling and Simeone where fractions of files in a single cache may be evicted at periodic times, and generalize it to a distributed coded caching scenario. We consider the network load, which we define as a weighted sum of data rates due to content download and cache updates. We formulate an optimization problem minimizing the network load, show that this problem is convex, and demonstrate that our problem formulation is a generalization of the previously studied single cache and static caching cases. We investigate what gains can be expected, both with respect to optimally managed (but uncoded) caching, as well as static caching. We furthermore analyze how the density of SBSs, the burstiness of the request process, the frequency of cache updates, as well as the update cost, impacts the network performance. Finally, we prove the optimality of static caching under a Poisson request process and show that the single cache case has a simple greedy solution.

Paper D

In Paper C, we assumed perfect knowledge about the distribution of the inter-request times for each file. Furthermore, the steady-state distribution of the number SBSs within communication range of a user placing a request was assumed to be perfectly known. In Paper D, we build on our previous work and suggest an RL approach to optimize the content allocation and cache management without any prior knowledge of request statistics or SBS distribution. In order to investigate the network load reductions for a scenario where the request process is heterogenous in space, we suggest a multi-agent reinforcement learning (MARL) algorithm allowing each SBS to autonomously make cache management decisions. We show that the reductions in network load increase as the request process deviates from being spatially homogeneous. The MARL approach furthermore allows us to evaluate the potential benefits of updating content cached at SBSs individually, reducing the total amount of data that have to be sent to the caches.

6.2 Conclusion

In summary, this thesis has thoroughly examined the significant reductions in backhaul traffic offered by the caching of content across SBSs and mobile devices using ECCs. When the repair problem is considered in D2D caching networks, we conclude that the choice of ECC is of considerable importance to minimize the backhaul traffic. The repair frequency must be fairly high for the caching of content across mobile devices to bring any benefit over downloading all the requested content over the backhaul, although the exact frequency depends on the code and network parameters. If mobile devices arrive to the wireless network with cached content, the requirements on the repair frequency are less stringent. Interestingly, an infinite repair frequency, i.e., instantaneous repairs, is not necessarily optimal. When considering the number of mobile devices to involve in the caching of a particular (encoded) file, we observe that as many devices as possible should be incorporated.

Requests for files arrive in bursts, an effect which is accurately captured by a renewal request process with memory. For such request processes, the periodic eviction of coded packets from the caches markedly reduces the amount of data downloaded over the backhaul compared to when content is cached statically. The backhaul traffic reductions become more significant with the in-

creased burstiness of the request process. For homogeneous request processes, it appears that synchronously updating the cached content is beneficial over asynchronous updates. Conversely, asynchronous updates are necessary under heterogenous request processes to minimize the backhaul traffic.

6.3 Future Work

In Paper C, we study the performance of cache management policies where coded content may be evicted from the caches at periodic times. New requests trigger content to be replenished, or updated, to the caches. This data replenishment resembles the content repair studied in Paper A where the use of particular ECCs can reduce the amount of data that have to be transmitted to repair lost content. Such ECCs could hence be applied when caching coded content according to the cache management policies from Paper C. It would be very interesting to investigate to what extent this potential reduction in update data traffic impacts the total network performance.

In order to study the distributed cache management from Paper C under asynchronous cache updates, we devise an RL approach in Paper D. Although, assuming a spatially heterogenous request process, our RL algorithm is able to learn caching policies that significantly outperform caching policies obtained using the optimization in Paper C assuming a homogenous request process and known request statistics and SBS distribution, we remain uncertain about the theoretical best performance that can be expected using such caching policies. Therefore, it is worthwhile to pursue further analysis of coded distributed caching policies under spatially heterogenous and bursty request processes to characterize the performance that can be achieved or bounds thereof.

Finally, all our research has focused on the caching of bulky content, e.g., video files, which has been shown to be the main driver of wireless network traffic. Video content is static in the sense that it is produced once and then watched many times. How old it is does not matter, only its popularity and how frequently it is requested. Although the trend of ever increasing wireless traffic due to video is expected to continue for several years, new applications begin to emerge where the freshness of the data is important. For example, much of the training for speech recognition in virtual assistants like Apple's Siri or Amazon's Alexa is performed at servers (i.e., the cloud) operated by the companies. The configuration is cached at the user device and can be

outdated as more recent configurations become available. Consequently, much of our previous work could be revisited and analyzed using a metric like age of information [56].

References

[1] *Cisco annual internet report*, White Paper, Cisco, Mar. 2020.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?", *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[3] *Ericsson mobility report*, White Paper, Ericsson, Nov. 2020.

[4] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks", *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[5] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers", *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 16–22, Aug. 2016.

[6] J. G. Andrews, "Seven ways that HetNets are a cellular paradigm shift", *IEEE Commun. Mag.*, vol. 51, no. 3, pp. 136–144, Mar. 2013.

[7] F. Boccardi, R. W. Heath Jr., A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G", *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[8] J. C. McCallum, *Historical memory prices*, https://jcmit.net/memoryprice.htm, Accessed: Jan. 2021.

[9] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks", *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, Jun. 2018.

[10] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers", *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[11] V. Bioglio, F. Gabry, and I. Land, "Optimizing MDS codes for caching at the edge", in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, 2015.

[12] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[13] S. N. Chiu, D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*. Wiley, 2013.

[14] S. L. Miller and D. Childers, *Probability and Random Processes*. Elsevier, 2004.

[15] B. Błaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks", in *IEEE Int. Conf. Commun. (ICC)*, London, UK, 2015, pp. 3358–3363.

[16] S. Tamoor-ul-Hassan, M. Bennis, P. H. J. Nardelli, and M. Latva-aho, "Caching in wireless small cell networks: A storage-bandwidth tradeoff", *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1175–1178, Jun. 2016.

[17] J. Pääkkönen, C. Hollanti, and O. Tirkkonen, "Device-to-device data storage for mobile cellular systems", in *Proc. IEEE Global Commun. Work. (GLOBECOM)*, Atlanta, GA, 2013.

[18] R. G. Gallager, *Stochastic Processes: Theory for Applications*. Cambridge University Press, 2013.

[19] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks", in *Mobile Computing*, ser. The Kluwer International Series in Engineering and Computer Science, vol. 353, Boston, MA: Springer, 1996, pp. 153–181.

[20] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks", *IEEE Trans. Mobile Comput.*, vol. 2, no. 3, pp. 257–269, Jul. 2003.

[21] P. Gupta and P. R. Kumar, "The capacity of wireless networks", *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[22] M. Abdulla and R. Simon, "Characteristics of common mobility models for opportunistic networks", in *Proc. 2nd ACM Workshop Performance Monitoring Measurement Heterogeneous Wireless Wired Networks*, Chania, Greece, 2007, pp. 105–109.

[23] J. Pedersen, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Optimizing MDS coded caching in wireless networks with device-to-device communication", *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 286–295, Jan. 2019.

[24] G. F. Newell, "The M/G/∞ queue", *J. SIAM Appl. Math.*, vol. 14, no. 1, pp. 86–88, Jan. 1966.

[25] A. Ullah and D. E. A. Giles, *Handbook of Empirical Economics and Finance*. CRC Press, 2011.

[26] A. Greiner-Petter, M. Schubotz, F. Müller, C. Breitinger, H. Cohl, A. Aizawa, and B. Gipp, "Discovering mathematical objects of interest—a study of mathematical notations", in *Proc. Web Conf.*, New York, NY, 2020, pp. 1445–1456.

[27] D. H. Zanette, "Zipf's law and the creation of musical context", *Musicae Scientiae*, vol. 10, no. 1, pp. 3–18, Mar. 2006.

[28] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications", in *Proc. IEEE 18th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, New York, NY, 1999, pp. 126–134.

[29] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of YouTube videos", in *Proc. Int. Workshop Quality Service*, Enschede, The Netherlands, 2008, pp. 229–238.

[30] C. Costa, I. Cunha, A. Borges, C. Ramos, M. Rocha, J. Almeida, and B. Ribeiro-Neto, "Analyzing client interactivity in streaming media", in *Proc. Int. Conf. World Wide Web*, New York, NY, 2004, pp. 534–543.

[31] M. Rausand and A. Høyland, *System Reliability Theory: Models, Statistical Methods, and Applications*. Wiley-Interscience, 2004.

[32] J. Goseling and O. Simeone, "Soft-TTL: Time-varying fractional caching", *IEEE Netw. Lett.*, vol. 1, no. 1, pp. 18–21, Mar. 2019.

[33] E. Ozfatura and D. Gündüz, "Mobility-aware coded storage and delivery", *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3275–3285, Jun. 2020.

[34] R. Wang, J. Zhang, S. H. Song, and K. B. Letaief, "Mobility-aware caching in D2D networks", *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5001–5015, Aug. 2017.

[35] A. Piemontese and A. Graell i Amat, "MDS-coded distributed caching for low delay wireless content delivery", *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1600–1612, Feb. 2019.

[36] A. Ferragut, I. Rodriguez, and F. Paganini, "Optimizing TTL caches under heavy-tailed demands", *SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 101–112, Jun. 2016.

[37] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results", *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, Sep. 2002.

[38] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance", in *Proc. 24th Int. Teletraffic Congr.*, Kraków, Poland, 2012.

[39] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi, "Check before storing: What is the performance price of content integrity verification in LRU caching?", *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 59–67, Jul. 2013.

[40] M. Dehghan, L. Massoulié, D. Towsley, D. S. Menasché, and Y. C. Tay, "A utility optimization approach to network cache design", *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1013–1027, Jun. 2019.

[41] M. Silberstein, L. Ganesh, Y. Wang, L. Alvisi, and M. Dahlin, "Lazy means smart: Reducing repair bandwidth costs in erasure-coded distributed storage", in *Proc. Int. Conf. Syst. Storage (SYSTOR)*, Haifa, Israel, 2014.

[42] W. E. Ryan and S. Lin, *Channel codes: Classical and modern*. Cambridge University Press, 2009.

[43] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland Pub. Co., 1977.

[44] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields", *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[45] G. Garrammone, "On decoding complexity of Reed-Solomon codes on the packet erasure channel", *IEEE Commun. Lett.*, vol. 17, no. 4, pp. 773–776, Apr. 2013.

[46] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems", *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[47] J. Pedersen, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Distributed storage in mobile wireless networks with device-to-device communication", *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4862–4878, Nov. 2016.

[48] D. Papailiopoulos and A. Dimakis, "Locally repairable codes", *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[49] K. M. Bretthauer and B. Shetty, "The nonlinear resource allocation problem", *Oper. Res.*, vol. 43, no. 4, pp. 670–683, Aug. 1995.

[50] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009.

[51] G. B. Dantzig, "Origins of the simplex method", Dep. Oper. Res., Stanford University, Tech. Rep., 1987.

[52] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems", *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[53] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[54] C. J. C. H. Watkins, "Learning from delayed rewards", PhD thesis, University of Cambridge, UK, 1989.

[55] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning", in *Proc. Int. Conf. Learning Rep. (ICLR)*, San Juan, Puerto Rico, 2016.

[56] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources", *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1827, Mar. 2019.