Distribution and storage in networks

Remigiusz Modrzejewski

UNIVERSITÉ DE NICE - SOPHIA-ANTIPOLIS
ÉCOLE DOCTORALE DES SCIENCES ET TECHNOLOGIES DE
L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

Docteur en Sciences

de l'Université de Nice Sophia Antipolis
Mention : Informatique

Defended by

Remigiusz MODRZEJEWSKI

Distribution and Storage in Networks

COATI Project
(I3S (CNRS/UNS), Inria)
Advisor:
Jean-Claude BERMOND

Jury:

Reviewers:		
Annie GRAVEY	-	TELECOM Bretagne (Brest, France)
Joseph G. PETERS	-	Simon Fraser University (Vancouver, Canada)
Laurent VIENNOT	-	Inria (Paris, France)
Examinators:		
Laurent LEFÈVRE	-	Inria (Lyon, France)
Jean-Claude BERMOND	-	CNRS (Sophia Antipolis, France)
Stéphane PÉRENNES	-	CNRS (Sophia Antipolis, France)
Frédéric GIROIRE	-	CNRS (Sophia Antipolis, France)
Invited:		
Jérôme GALTIER	-	Orange Labs (Sophia Antipolis, France)

To my beloved wife and my family.

Acknowledgments

First of all, I want to thank all the persons who worked with me on this text. My reviewers: Annie Gravey, Joseph Peters and Laurent Viennot. My advisor, Jean-Claude Bermond, as well as Stéphane Pérennes and Frédéric Giroire who helped me forge it chapter by chapter. Thanks to all the coauthors of the studies the thesis is based on.

I thank my wife, my parents and my sister for all the support that kept me going for these years. I also thank all the MASCOTTE/COATI team for making the doctoral studies so pleasant. Thanks to Issam and Julio for all the times I would be lost without your help. Special thanks to Patricia, without whom I would never succeed with any office.

Abstract

In this thesis we study multiple approaches to efficiently accomodating for the future growth of the Internet. The exponential growth of Internet traffic, reported to be as high as 41% in peak throughput in 2012 alone, continues to pose challenges to all interested parties. Therefore, to accommodate this growth, smart management and communication protocols are needed.

The basic protocols of the Internet are point-to-point in nature. However, the traffic is largely broadcasting, with projections stating that as much as 80-90% of it will be video by 2016. This discrepancy leads to an inefficiency, where multiple copies of essentially the same messages travel in parallel through the same links. In this thesis we study multiple approaches to mitigating this inefficiency.

The contributions are organized by layers and phases of the network life. We look into optimal cache provisioning during network design. Next, we move to managing an existing network. We look into putting devices to sleep mode, using caching and cooperation with Content Distribution Networks. In the application layer, we look into maintaining balanced trees for media broadcasting. Finally, we analyze data survivability in a distributed backup system, which can reduce network traffic by putting the backups closer to the client than if using a data center.

Our work is based on both theoretical methods, like Markov chains and linear programming, as well as empirical tools, like simulation and experimentation.

Abstract

Dans cette thèse, nous étudions divers problèmes dont l'objectif est de gérer la croissance d'internet plus efficacement. En effet celle-ci est très vive : 41% pour le pic en 2012. Afin de répondre aux défis posés par cette évolution aux divers acteurs du réseau, des protocoles de gestion et de communication plus intelligents sont nécessaires.

Les protocoles de l'Internet furent conçus comme des protocoles point à point. Or, la part de la diffusion de média dans le trafic est prépondérante et en nette hausse, et des projections indiquent qu'en 2016 80-90% du trafic sera engendré par de la diffusion vidéo. Cette divergence entraîne des inefficacités, car des multiples copies d'un message transitent par un lien. Dans cette thèse, nous étudions comment remediér à cette inefficacité.

Nos contributions sont organisées selon les couches et les phases de déploiement du réseau. Nous étudions le placement de caches lors de la conception du réseau. Ensuite, pour la gestion d'un réseau, nous regardons quand placer des appareils en veille, en utilisant un mécanisme de cache et en coopération avec des réseaux de distribution. Puis, au niveau de la couche application, nous étudions un problème de maintenance d'arbres équilibrés pour la diffusion de média. Enfin, nous analysons la probabilité de survie des données dans un système de sauvegarde distribuée.

Notre travail se fonde à la fois sur des méthodes théoriques (Chaînes de Markov, Programmation Linéaire), mais aussi sur des outils empiriques tels que la simulation et l'expérimentation.

Contents

CHAPTER 1

Introduction

In this thesis we study multiple approaches to optimizing the current and future Internet. In this introduction we motivate these approaches, mention the techniques used and finally enumerate our main contributions.

1.1 Motivation

The impact of the Internet on our lives has been becoming more and more evident in recent years. Nowadays, people are using it in work, in free time and in the commute between them. It is gradually replacing printed press, radio and television. This results in an exponential-like growth in network traffic, that is likely to last in the foreseeable future. According to a report by Cisco [Cis13], the peak global throughput has increased by 41% through the year 2012 alone. Sustaining such a growth, while minimizing investments and energy consumption, requires new approaches to how the networks comprising the Internet are used and operated.

In parallel to the increase of traffic volume, we see a shift in its nature. An increasing part of the traffic is media broadcasting. In fact, according to projections in the same Cisco report, video traffic alone will constitute 69 percent of all consumer traffic in 2017, up from 57 percent in 2012. Together with file sharing, this should approach 90% of all traffic. These kind of flows share the property that they are not concerned by which

server serves the client. This motivates the main questions of the thesis, which are concentrated on the study of content dissemination and peer-to-peer systems. Particularly, it is beneficial to all involved parties if a client is served from a location as close as possible. We study three different classes of such close locations:

- a mirror server located closer in the network, what is the case for Content Distribution Networks (CDN),
- a cache located at a nearby network device (in-network caching),
- or another client sharing his own resources, as in Peer-to-Peer networks (P2P).

They are described in more detail in Section 1.4 and studied in various chapters of this thesis, utilizing a set of techniques described in Section 1.6.

In this thesis I study multiple models of communication over Internet. I also look into multiple phases of the network life, from conception of physical layer, dimensioning, management, to using it in a more distributed way.

1.2 Network transmission taxonomy

Due to convergence of communications, computer networks transmit any kind of files and streams. Observing the nature of network traffic, in general one can divide the volume into three main categories: video streaming, file sharing (includes video files) and everything else. The historical and projected traffic amounts are plotted in Figure 1.1. In this section, we classify the bulk of the traffic into a few more categories and briefly describe them in terms of: volume of traffic, delay sensitivity and whether they may be cached, relocated or multicasted.

Conversations First, there is a broad category of communication between users. This may include emails, instant messages, video chat as well as many specific applications, e.g. computer games. All messages in this category have two given endpoints and are unique, transmitted only once. All the other characteristics vary from application to application. Email is usually low traffic and very tolerable towards transmission

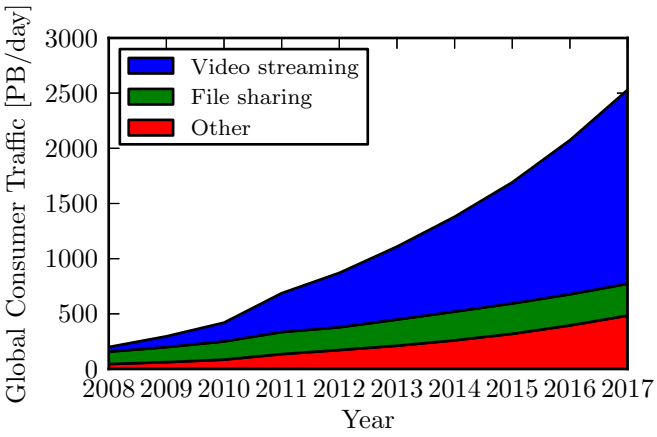


Figure 1.1: Traffic evolution according to [Cis13], historical data taken from [Cis09, Cis10, Cis11, Cis12].

delays and failures. Computer games are low traffic, but very sensitive towards delay. Video chat is high traffic and somewhat sensitive towards delay. Other applications can be any mixture of above. Flows belonging into this category generally cannot be cached nor relocated and they would not benefit from any form of multicasting.

Web This category contains the huge interlinked collection of objects, known as the World Wide Web. While it may be used as a front-end to the other categories, its main purpose is publishing. Traffic requirements depend on the type of viewed content, from tiny in case of plain text to huge in case of rich multimedia. Delay tolerance is medium, real time transmission is not required, but quick delivery is crucial for client satisfaction. Many objects are static and common across web sites, like logos or other images, and can be cached. However, the transmissions connected to a single location are usually relatively small and predicting the next location visited by the user is not a trivial problem. This has led to numerous studies on pre-fetching, surveyed in [Wan99].

Live streaming Live streaming can be seen as television over internet. A source broadcasts live media and clients display it after a short delay. This delay accommodates for buffering, transmissions and eventual re-

transmissions. It also serves as a *deadline* – it is useless for a client to receive a fragment of the stream delayed by more, as its playback time already passed and it will never be used again. Along this delay bound, these kind of flows often have very big bandwidth. The fact that we have a big number of clients interested in receiving exactly the same content at the same time makes a perfect match for multicasting.

On-demand streaming On-demand is another type of streaming, where user chooses a media file from a previously offered collection. While transmission requirements are roughly the same as in live streaming, the user may choose to pause and resume the playback at will. Optimizing this kind of streaming raises more challenges. The sizes of collections are usually much larger than number of live channels. Additionally, two users watching the same file may be too far apart in playback time to treat them as watching the same thing.

File sharing A big part of Internet's bandwidth is used by file sharing. In this kind of application users typically share single big files. As download times are often counter in hours, there is not much pressure on delays. However, as the user typically wants to receive the file as soon as possible, there is demand for practically unlimited bandwidth.

Cloud computing One use case for computer networks, that has been gaining on importance in the recent years, is *cloud computing*. This is a broad category, containing any kind of tasks performed server-side, controlled by a remote operator. These might be as different as multimedia editing, distributed computing or simply data storage. The main motivation is moving computing resources from the client, which can become simpler, towards centralized facilities, where economies of scale can be leveraged.

Delay tolerance depends on the actual application and cloud computing flows rarely refer static data that could be cached. However, as the operator is already remote to the servers performing the tasks and is usually oblivious to their location, the servers themselves can be placed at a possibly close location to the client.

Category	Traffic volume	Delay tolerance	Optimization
Conversations	Variable	Low	—
Web	Variable	Medium	Caching
Live streaming	High	Low	Multicasting
On-demand streaming	High	Low	Caching
File sharing	High	High	Caching, relocating
Cloud computing	Variable	Variable	Relocating

Table 1.1: Summary of the network flow classes, their properties and natural optimizations.

Table 1.1 summarizes the above classification. Note that the only class without a natural way of optimization, from network perspective, are conversations. On the other hand, there is little redundancy in this class. Thus, we can state that in most cases if there is an inefficiency, we can attempt to address it.

1.3 Content popularity and caching

As discussed in previous section, the majority of transmissions over Internet are expected to be video streaming. In both live and on-demand streaming, the same content is received by multiple users. For live streaming this opens the possibility of multicasting, either by IP multicast or peer-to-peer networks, studied in Chapter 4. For on-demand streaming caching can be employed. It plays important roles in Chapters 2 and 3.

In general, caching means storing a subset of a collection of objects in another place, from where retrieval is significantly cheaper than from said collection. Caches are ubiquitous in all areas of computing. A remarkable example are CPU caches. A small amount of static RAM located on the CPU, usually a few megabytes, mirrors some parts of the main memory, usually a few gigabytes of dynamic RAM. If data accessed by the processor is present within a cache, we say it is a *cache hit* and the access takes a few nanoseconds. Otherwise, we face a *cache miss* and the access is directed towards the main memory, what is measured hundreds of nanoseconds. Therefore the probability of the required data

being present in the cache, called *cache hit ratio*, is crucial for the overall efficiency of the system. An in-depth explanation of caching in hardware can be found in [JNW10].

In networking, a well known usage of caching are web proxies. They are servers usually located in the same network as their clients. Proxies essentially cache any web content. They are either enabled explicitly in client's browser configuration, or the network is configured to redirect requests to a proxy, possibly without client's knowledge. In the latter case we say the proxy is *transparent*. Another notable example of a cache is a server of a Content Distribution Network, as described in Section 1.4. Such a server is located at a network close to the client, but in case of a miss it needs to forward the request to the original content provider, which may be very far. In this thesis we look into *in-network* caches, also described in Section 1.4.

The interest in caching in networking is to allow many clients to obtain some data from a nearby cache, thus saving multiple redundant long-range transmissions. This implies that the effectiveness of caches depends on popularity, understood as the number of clients requesting for an object (web page, song, movie, etc.) over some time. It is often stated in the literature that it follows a power-law. This means that there are very few objects that are very popular and a lot objects that are not popular.

Zipf's law is proposed to described popularity of objects in the Internet. In was proposed in [Zip32], in order to study natural languages. It states that the frequency of any word is inversely proportional to its rank in the frequency list. More formally, the popularity f of object ranked k is:

$$f(k, \beta) = \frac{1}{k^{\beta}} \tag{1.1}$$

where β is a positive real parameter. A probability distribution of accessing a given object is obtained by simply dividing the above by the sum for all objects. This distribution was found to be a good fit for Web traffic in [BCF+99], where it was found that the value of β falls within the range $[0.6, 0.8]$, depending on the collection and viewers population. More recent studies, specializing on video traffic, tend to confirm this, e.g. [CDL08, GHM13]. However, values observed can be as low as 0.56 in [GALM07] and as high as 1.5 in [CKR+07].

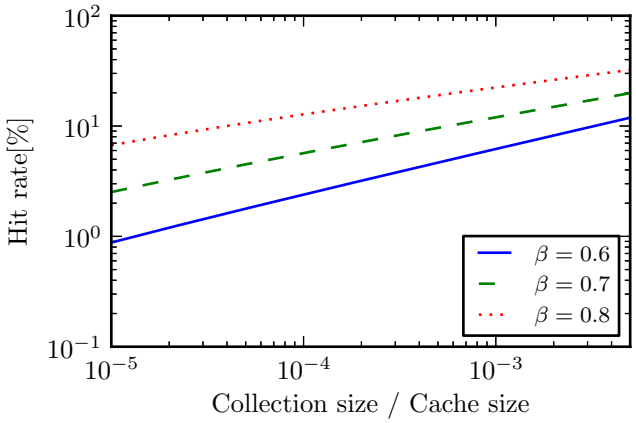


Figure 1.2: Hit rates in function of cache size relative to the size of the collection it caches, for three different values of the distribution's exponent β. Computed for $n = 10^7$ objects in collection. Note the log-log scales.

Assume that we know the object popularity for a period of time, we store in the cache the most popular content for that time and the period is long enough to make any initial cache misses insignificant. Then, we obtain the formula for the hit rate of a cache mirroring s objects of a collection of n objects:

$$h(s, n, \beta) = \frac{\sum_{k=1}^{s} k^{-\beta}}{\sum_{k=1}^{n} k^{-\beta}}, \tag{1.2}$$

which is plotted in Figure 1.2. It shows the hit rates obtained, with conservative values of β, by a single cache that can store up to $1/200$ of the collection. For example, if the collection is 10PB[1] of video clips 100MB each, with $\beta = 0.8$, we would obtain 22.4% hit ratio with a 10TB cache, 12.8% with a 1TB and 6.7% with a 10GB one. An important observation in the plot is that, even within this conservative range, small variations of the β parameter have a huge effect on cache efficiency.

[1] PB = 10^3TB = 10^6GB = 10^9MB = 10^{15}B

1.4 Content distribution models

The majority of current internet traffic is delivered using a protocol stack built on the Internet Protocol (IP), which takes its name from being the one used to deliver messages between hosts that may be connected to different networks. Below it we have the link layer, which governs communications of devices sharing a link. Above there is the transport layer, which ensures continuity of host-to-host communication, mainly by the Transmission Control Protocol (TCP) protocol, and the application layer, which engulfs any communication abstracting over the layers below.

The TCP/IP stack is conversational by design. On the other hand, most of the data flows through today's networks are either content distribution or, starting recently and gaining momentum, *cloud based* services. This mismatch creates a range of opportunities to introduce more efficient architectures for the future Internet. In this section we briefly characterize the main ones.

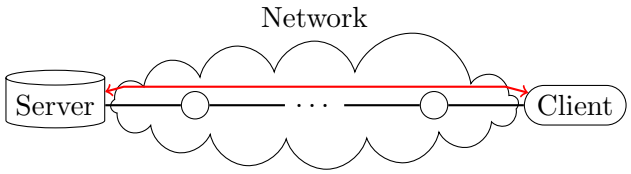


Figure 1.3: A communication flow between a server and a client, passing a network comprised of multiple routers.

Client-server

TCP/IP assumes communication between two points. These usually are a single client and a single server. This is depicted in Figure 1.3. Distinction between both endpoints comes down to the fact, that it is the client who initiates the communication. Thus, he must know the address of the server. This reflects how actual users use network services. Even if ultimately they want to send a message to another user, usually they will do so, e.g., by the service provided by email servers.

Inefficiency arises if the communication is one-to-many by nature. Extreme, but increasingly significant, example of such communication is

media broadcasting. In this case, multiple copies of essentially the same messages flow in parallel through the network, often sharing and probably congesting the same links. This is shown in Figure 1.4. Furthermore, media broadcasting requires high and ever-growing bandwidth. Therefore, mitigating this redundancy is particularly important.

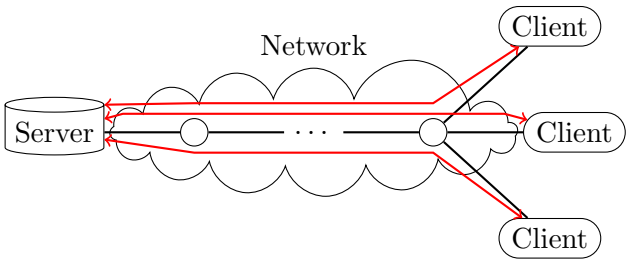


Figure 1.4: Multiple communication flows between a server and its clients, passing in parallel through the same routers in the network.

IP multicast

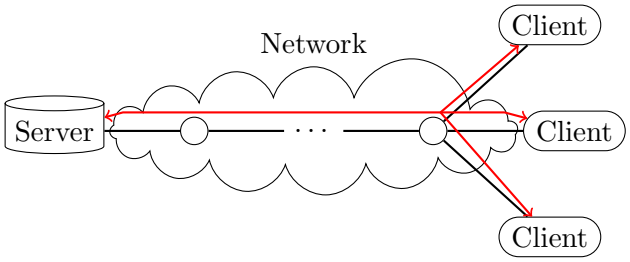


Figure 1.5: A single flow is multicasted at IP routers towards all interested clients using IP multicast.

The first solution to this inefficiency is IP multicast. It was proposed in [DEE88]. It is implemented in standard IP routers. Every broadcast channel is assigned a *multicast address*. Clients interested in receiving it subscribe using the *Internet Group Management Protocol*, or the *Multicast Listener Discovery* component of IPv6. A router, seeing such a subscription, will forward any messages related to this channel towards the client. This is illustrated in Figure 1.5. If it is not receiving it yet, it will also signal to its default route that it has clients interested. The

broadcaster is simply sending single messages to the multicast address, it is not responsible for multicasting or retransmissions.

This solution is obviously limited to live broadcasting. Due to lack of applications, following a mismatch between protocol and popular needs, economic reasons and security issues, IP multicast is not widely deployed. In practice it is restricted to specific services, like providing traditional television inside an operator's network. IP multicast traversing multiple networks is rare.

Content distribution networks

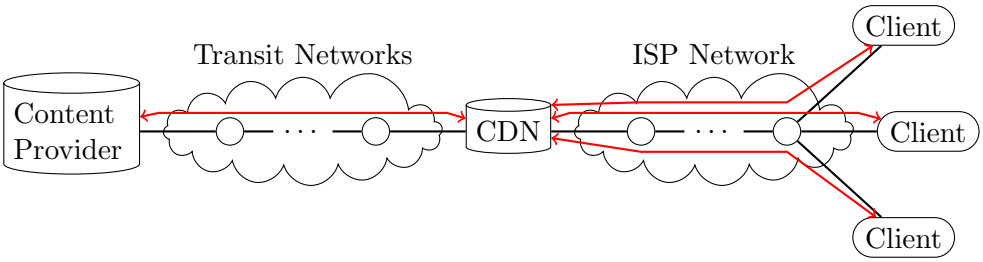


Figure 1.6: Multiple clients request the same content from a Content Provider. Only one copy of the content is passed, traversing possibly multiple transit networks, to a CDN server. This server, located on an edge of an ISP network, distributes the content to all the clients connected to it.

One response to aforementioned inefficiency, widely deployed in the wild, are Content Distribution Networks (CDNs). This solution leverages a particular socio-economical phenomena: in case of media distribution, both clients and providers are willing to pay anybody who can ensure swift transmission, not only to network operators giving them connectivity. In fact, clients pay, probably indirectly by watching advertisements, for the received content to big Content Providers (CPs). CDNs install themselves as a man-in-the-middle. They are paid by the CPs and serve to clients content previously obtained from the CPs. Aggregating multiple CPs, they can afford to put their servers in the edges of many networks.

When a client wants to access some content of a CP, he is redirected towards the nearest CDN server. If it is the first request for this content

in some time, the CDN server obtains a copy from original CP, stores it and provides to the client. On subsequent requests the copy stored by the server will be used, unless it is deleted due to too long time between requests, thus eliminating the need for parallel long-haul communications. This is shown in Figure 1.6.

Arguably the most notable CDN is Akamai, founded in 1998. By their own claim in [Aka13], they serve *15-30% of the world's Internet traffic on a daily basis*. This is achieved using a *global network of more than 85,000 servers in 70 countries*. Less is known about other major CDNs, like Level 3 or Limelight. Some ISPs maintain their own CDNs. These, like IP multicast for live streaming, usually serve their own on-demand streaming offerings. The Cisco report [Cis13] estimates that CDNs currently account for 34% of Internet traffic. That number should rise to 51% by the year 2017.

Content Distribution Networks play a major role in Chapter 3.

Peer-to-Peer

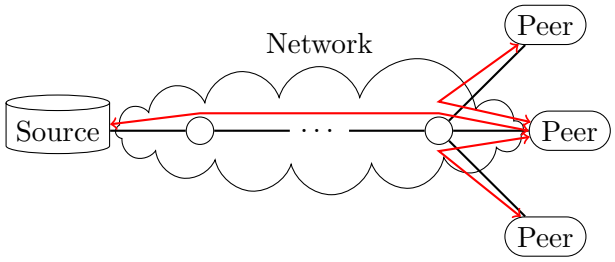


Figure 1.7: A source providing a single peer, who then shares with multiple local peers.

If the content provider cannot, or is unwilling to, employ a CDN, then efficient content distribution can be undertaken by the clients. The basic principle of Peer-to-Peer (P2P) networks is that most of its users are clients and servers at the same time. Hence, they are called peers. After receiving a fragment of media, a peer is expected to pass it to others, as shown in Figure 1.7. The obvious exception is the original source of the media, or more generally any peer that is not interesting in receiving, but has some data to serve.

P2P networks got some popularity in multiple areas. Arguably the one with biggest mindshare is the file sharing network Bit Torrent, proposed in [Coh03]. A popular solution for amateur video broadcasting is SopCast, investigated in [LFK+09]. A commercial success in China was achieved by the P2P broadcaster PPTV, formerly PPLive, studied in [HLL+07]. As they claim in [PPL13], *PPTV has more than 260 million users*. A number of P2P storage systems have been proposed [DR01, BTcC+04, KBC+00]. However, the only well-known commercial system, Wuala, has switched to a purely client-server architecture. According to [MBM12], this was dictated by a significant drop in data center prices, making this easier design economically feasible. According to the Cisco report [Cis13], just P2P file sharing constitutes 23% of current Internet traffic.

Note that many popular P2P networks do not explicitly optimize for locality. However, prioritizing peers with high throughput, like in Bit Torrent, indirectly favors peers which are closer network-wise. Additionally, many popular clients implement the *Local Peer Discovery* extension, see [Bit13].

Peer-to-Peer networks are analyzed in Chapters 4 and 5.

In-network caching and Content Centric Networking

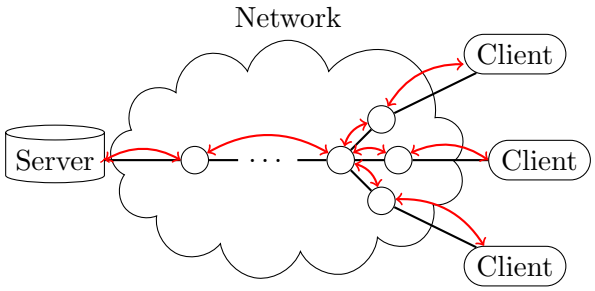


Figure 1.8: Object dissemination over a network of caching routers. Each router on a path between a client and source server stores the object in its cache. When other clients request the same object, it needs to be disseminated only from the closest router common to their paths to the source server.

Finally, network operators can battle the inefficiency by deploying in-network caches. A scenario, where each router is augmented with a

cache and on a missed request passes it to the next cache on the path towards a server, is depicted in Figure 1.8. Unlike in IP multicast, caching is not restricted to live streaming. Any popular objects can be stored in-network, to be accessed at client's convenience. The motivation for an ISP to undertake such an investment is twofold. First, caches reduce the latency, making its clients happier. Second, they reduce long-haul traffic, thus saving money to the operator. While both of these are achieved, to some extent, by the third-party operation of CDNs (which essentially can be seen as caches themselves), there are some advantages to operator's caches. First, they can be provider-agnostic, therefore optimizing the savings of the operator, disregarding the nature of the source of the content. However, note there may be possible copyright issues, as providers want to be in control of who and when can access their content. Second, placement and dimensioning can be tailored to benefit the particular network (Chapter 2 studies this problem). Third, it is the ISP who controls the operation, allowing it to respond to current network conditions (studied in Chapter 3).

In-network caching is attracting research interest thanks to Content Centric Networking (CCN), proposed in [JST+09]. It uses the concept of *nearest replica routing*, where requests are propagated towards the nearest cache containing the data, instead of going along the shortest route towards a known host. This broadcast-centric approach allows for massive reuse of media chunks, virtually eliminating redundant parallel transfers seen before. CCN's network layer can be used as an alternative to the Internet Protocol, as well as be layered over it (or over UDP) for easier deployment. Also, unlike TCP which concerns connection, CCN ensures that the content is received intact and as requested.

1.5 Metrics studied

In different studies we are interested in optimizing or evaluating different metrics. These need to be defined and modeled in a clear and simple way. In this section we briefly describe them.

One concern that has been growing throughout the recent years, because of rising prices of electricity and worries about global warming, is energy consumption. According to [LVHV+12], the Information and Communication Technology sector already consumes 2% of global elec-

tricity and experiences 10% of yearly increase. In Chapters 2 and 3, both concerned in network layer, we optimize power consumption.

When it comes to telecommunication networks, we see huge improvements in energy efficiency achieved by device manufacturers. In [LKWG11] it is shown that power consumed per unit of data transmitted is halved every four or five years. However, as we know from [Cis13], the amount of data transmitted is at least tripled over the same period. Furthermore, the trend towards decreasing per bit energy efficiency can slow down significantly. As discussed in an ICC 2013 keynote [Win13], per-fiber capacities are approaching the Shannon limit computed in [EFKW08]. On the other hand, there is no clear reason to believe traffic growth will come to a standstill. Thus, we arrive at the conclusion that improving network power efficiency is of global importance.

Three ways of achieving this are considered in this thesis. The first is shortening the routes travelled by traffic, either by caching or choosing a server that is closer to the client. The second is putting unused components into sleep or low-power modes. This has been first proposed in [GS03]. The third way is aggregating the traffic. It has been shown, in the influential paper of Chabarek [CSB+08], that the energy consumption of network equipment is not proportional to the volume of traffic. Thus, using fewer devices with higher load may lead to significant savings. This may happen both in network deployment, or by putting more devices to sleep mode.

For Peer-to-Peer broadcasting in Chapter 4, we look into the time between a failure and finishing repair of the tree. The time is expressed in number of turns, where a turn is the time needed for each node in the tree to perform a single operation. For the simulations we can look into other metrics, like the average delay or fraction of media received correctly. Both the values are improved by the algorithm. The delay is the time between the source sending some content and the nodes receiving it. It depends on the distance of the node from the source; thus balancing the tree minimizes delay. When a node has too many children, it cannot sustain streaming to every one of them. Thus nodes that have overloaded ancestors do not receive all the media. Our algorithm improves this as well.

In Chapter 5, we look into Peer-to-Peer backup systems. To ensure data survival, such a system employs a continuous self-repair process.

Whenever a fragment of the data is lost, it is being reconstructed from redundant data in the network. To achieve it, peers need to upload the data. Thus, the system continuously uses bandwidth and we evaluate its usage. If available bandwidth is too low to accommodate all the losses, we arrive at a probability of losing some data. This probability is the most important characteristic for such a system.

When solving frequency assignment problem in Appendix A, we want to minimize either the number of radio channels needed or the interferences between nodes. Radio channels, modeled as the colours in a graph colouring, are a monetary cost to obtain. Interference, induced by other nearby devices using the same channel, has to be kept below a threshold to keep transmission reliable.

1.6 Techniques used

Over the course of this thesis we faced different problems, calling for different solutions. The main techniques used, ordered from the more theoretical to more empirical, are:

- Queueing and Markov chain analysis, in Chapter 5, described in Section 5.1
- Integer Linear Programming approaches, in Chapter 3 and Appendix A, described in Section 3.1
- Rounding or fractional relaxations of mixed integer linear programs, in Chapter 3, described in Section 3.2
- Branch-and-bound methods, in Appendix A
- Discrete event simulation, in Chapter 5
- Cycle based simulation, in Chapter 4
- Experiments, using commercial software[2] on a testbed platform[3], in Chapter 5

[2] `http://www.ubistorage.fr/`
[3] `https://www.grid5000.fr/`

A sizeable part of the work presented here does not fall into this classification. For example the analysis in Chapter 2 relies only on basic probability and algebra, leading to a straightforward exact algorithm. In Chapter 4 we use a potential function approach, similar to the ones used to prove the convergence to a Nash Equilibrium in game theory. In Appendix A it is a case-by-case analysis.

1.7 Contributions

The remainder of this thesis is organized around my contributions. What follows in this section are their short descriptions. Following the customs in our team, the alphabetic order of authors is employed for every paper other than *Energy Efficient Content Distribution in an ISP Network* [MCT+13].

Chapters begin with preliminary sections, setting up some context for the contribution. The bodies of chapters mainly correspond to research report versions of the respective publications. These are more detailed than the published articles.

Chapter 2: Energy Efficient Cache Provisioning

We look into saving the energy in the network design phase, by minimizing the requirements for deployed devices. The main contribution itself is lead by some general insights on caching and deriving power models of networks. Then, we study the problem of reducing power consumption in an Internet Service Provider (ISP) network by designing the content distribution infrastructure managed by the operator. We propose an algorithm to optimally decide where to cache the content inside the ISP network. We evaluate our solution over two case studies driven by operators feedback. Results show that the energy-efficient design of the content infrastructure brings substantial savings, both in terms of energy and in terms of bandwidth required at the peering point of the operator. Moreover, we study the impact of the content characteristics and the power consumption models. Finally, we derive some insights for the design of future energy-aware networks.

The results of this chapter have been accepted for publication in GLOBECOM 2013 [MCT+13].

Chapter 3: Energy Efficient Routing

In this chapter, we move to the management of an already deployed network. We consider saving the energy by aggregating traffic and putting some devices to sleep or low power modes. We study the impact of using in-network caches and content delivery network (CDN) cooperation on an energy-efficient routing. We formulate this problem as Energy Efficient Content Distribution and propose an integer linear program (ILP) and an efficient heuristic algorithm to solve it. The objective is to find a feasible routing, so that the total energy consumption of the network is minimized subject to satisfying all the demands and link capacity. We exhibit the range of parameters (size of caches, popularity of content, demand intensity, etc.) for which caches are useful. Experimental results show that by placing a cache on each backbone router to store the most popular content, along with well choosing the best content provider server for each demand to a CDN, we can save about 20% of power in the backbone, while 16% can be gained solely thanks to the use of caches.

The results of this chapter have been accepted for publication in ICC 2013 [AGL+13].

Chapter 4: Maintaining Balanced Trees For Structured Distributed Streaming Systems

In this chapter, we move to content distribution in the application layer. As discussed before, peer-to-peer networks reduce the broadcasting redundancy by allowing clients to share the content among themselves. We deal with some concerns about robustness of such a setup. We propose and analyze a simple localized algorithm to balance a tree. The motivation comes from live distributed streaming systems in which a source diffuses a content to peers via a tree, a node forwarding the data to its children. Such systems are subject to a high churn, peers frequently joining and leaving the system. It is thus crucial to be able to repair the diffusion tree to allow an efficient data distribution. In particular, due to bandwidth limitations, an efficient diffusion tree must ensure that node degrees are bounded. Moreover, to minimize the delay of the streaming, the depth of the diffusion tree must also be controlled. We propose here a simple distributed repair algorithm in which each node carries out local operations based on its degree and on the subtree sizes of its children.

In a synchronous setting, we first prove that starting from any n-node tree our process converges to a balanced tree in $O(n^2)$ turns. We then describe a more restrictive model, adding a small extra information to each node, under which we adapt our algorithm to converge in $\Theta(n \log n)$ turns. Finally, we exhibit by simulation that the convergence is much faster (logarithmic number of turns in average) for a random tree.

The results of this chapter have been accepted for publication in SIROCCO 2013 [GMNP13].

Chapter 5:Analysis of the Repair Time in Distributed Storage Systems

In the final chapter, we move from content distribution to distributed applications. One such application, with big bandwidth requirements, are online backups. A conservative approach to this task employs data centers. However, these usually are far away from the users. Instead, it is possible to use storage located at the perimeters of other nearby users of a distributed system. This, again, raises questions about reliability. To that end, these storage systems introduce redundancy to preserve the data in case of peer failures or departures. To ensure long-term fault tolerance, the storage system must have a self-repair service that continuously reconstructs lost fragments of redundancy. The speed of this reconstruction process is crucial for the data survival. This speed is mainly determined by available bandwidth, a critical resource of such systems. We propose a new analytical framework that takes into account the correlation of concurrent repairs when estimating the repair time and the probability of data loss. Mainly, we introduce queuing models in which reconstructions are served by peers at a rate that depends on the available bandwidth. The models and schemes proposed are validated by mathematical analysis, extensive set of simulations, and experimentation using the Grid'5000 test-bed platform.

The results of this chapter have been published in Globe 2013 [GGM$^+$13].

Appendix A: Weighted Improper Colouring

Appendix A contains work that is not concerned by reducing redundancy in network traffic. Instead, it is motivated by frequency assignment in satellite networks. Thus, it concerns link layer. In wireless networks, a

node interferes with other nodes, the level of interference depending on numerous parameters: distance between the nodes, geographical topography, obstacles, etc. We model this as a new graph colouring problem. We find some general bounds and optimal solutions for infinite grids. We model the problem using integer linear programming, propose and test heuristic and exact Branch-and-Bound algorithms on random cell-like graphs.

The results of this chapter have been published in IWOCA 2011 [ABG+11] and JDA 2012 [ABG+].

1.8 Bibliography

[ABG+] J. Araujo, J-C. Bermond, F. Giroire, F. Havet, D. Mazauric, and R. Modrzejewski. Weighted improper colouring. In *Journal of Discrete Algorithms volume 16*, pages 53–66.

[ABG+11] J. Araujo, J-C. Bermond, F. Giroire, F. Havet, D. Mazauric, and R. Modrzejewski. Weighted improper colouring. In *22nd International Workshop on Combinatorial Algorithms*, pages 1–18, 2011.

[AGL+13] J. Araujo, F. Giroire, Y. Liu, R. Modrzejewski, and J. Moulierac. Energy efficient content distribution. In *IEEE International Conference on Communications 2013 (ICC 2013)*, 2013. Accepted.

[Aka13] Akamai Technologies, Inc. The akamai internet, June 2013. URL: `http://www.akamai.com/html/riverbed/akamai_internet.html`.

[BCF+99] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134. IEEE, 1999.

[Bit13] BitTorrent, Inc. Basic bittorrent features, June 2013. URL: `http://www.bittorrent.com/help/manual/`

appendixa0206#Basic_BitTorrent_Features.Enable_Local_Peer_Discovery.

[BTcC+04] Ranjita Bhagwan, Kiran Tati, Yu chung Cheng, Stefan Savage, and Geoffrey M. Voelker. Total recall: System support for automated availability management. In *Proc. of the USENIX NSDI*, pages 337–350, 2004.

[CDL08] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and social network of youtube videos. In *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pages 229–238. IEEE, 2008.

[Cis09] Cisco Systems, Inc. Cisco visual networking index: Forecast and methodology, 2008-2014, June 2009. URL: http://www.cisco.com/web/BR/assets/docs/whitepaper_VNI_06_09.pdf.

[Cis10] Cisco Systems, Inc. Cisco visual networking index: Forecast and methodology, 2009-2014, June 2010. URL: http://large.stanford.edu/courses/2010/ph240/abdul-kafi1/docs/white_paper_c11-481360.pdf.

[Cis11] Cisco Systems, Inc. Cisco visual networking index: Forecast and methodology, 2010-2015, June 2011. URL: http://www.df.cl/prontus_df/site/artic/20110602/asocfile/20110602113637/white_paper_c11_481360_1_.pdf.

[Cis12] Cisco Systems, Inc. Cisco visual networking index: Forecast and methodology, 2011-2016, May 2012. URL: http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=2037.

[Cis13] Cisco Systems, Inc. Cisco visual networking index: Forecast and methodology, 2012-2017, May 2013. URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html.

[CKR+07] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[Coh03] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[CSB+08] Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, David Tsiang, and Steve Wright. Power awareness in network design and routing. In *INFOCOM'08: the 27th IEEE Conference on Computer Communications.*, pages 457–465, 2008.

[DEE88] S DEERING. Host extension for ip multicasting. *RFC 1112*, 1988.

[DR01] Peter Druschel and Antony Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 75–80. IEEE, 2001.

[EFKW08] René-Jean Essiambre, Gerard J Foschini, Gerhard Kramer, and Peter J Winzer. Capacity limits of information transport in fiber-optic networks. *Physical review letters*, 101(16):163901, 2008.

[GALM07] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM, 2007.

[GGM+13] F. Giroire, S. K. Gupta, R. Modrzejewski, J. Monteiro, and S. Pérennes. Repair time in distributed storage systems. In *6th International Conference on Data Management in Cloud, Grid and P2P Systems*, pages 99–111, 2013.

[GHM13] Fabrice Guillemin, Thierry Houdoin, and Stéphanie Moteau. Volatility of youtube content in orange networks and consequences. In *2013 IEEE International Conference on Communications (ICC2013)*, June 2013. Accepted.

[GMNP13] F. Giroire, R. Modrzejewski, N. Nisse, and S. Pérennes. Maintaining balanced trees for structured distributed streaming systems. In *20th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2013)*, 2013. Accepted.

[GS03] Maruti Gupta and Suresh Singh. Greening of the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26. ACM, 2003.

[HLL+07] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W Ross. A measurement study of a large-scale p2p iptv system. In *IEEE Transactions on Multimedia*, volume 9, pages 1672–1687, 2007.

[JNW10] Bruce Jacob, Spencer Ng, and David Wang. *Memory systems: cache, DRAM, disk*. Morgan Kaufmann, 2010.

[JST+09] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[KBC+00] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, et al. OceanStore: an architecture for global-scale persistent storage. *ACM SIGARCH Computer Architecture News*, 28(5):190–201, 2000.

[LFK+09] Y. Lu, B. Fallica, F.A. Kuipers, R.E. Kooij, and P.V. Mieghem. Assessing the quality of experience of sopcast. *International Journal of Internet Protocol Technology*, 4(1):11–23, 2009.

[LKWG11] C. Lange, D. Kosiankowski, R. Weidmann, and A. Gladisch. Energy consumption of telecommunication networks and related improvement options. *Selected Topics in Quantum Electronics, IEEE Journal of*, 17(2):285–295, 2011.

[LVHV+12] S. Lambert, W. Van Heddeghem, W. Vereecken, B. Lannoo, D. Colle, and M. Pickavet. Worldwide electricity consumption of communication networks. *Optics Express*, 20(26):513–524, 2012.

[MBM12] Thomas Mager, Ernst Biersack, and Pietro Michiardi. A measurement study of the wuala on-line storage service. In *Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on*, pages 237–248. IEEE, 2012.

[MCT+13] R. Modrzejewski, L. Chiaraviglio, I. Tahiri, F. Giroire, E. Le Rouzic, E. Bonetto, F. Musumeci, R. Gonzalez, and C. Guerrero. Energy efficient content distribution in an isp network. In *IEEE Global Communications Conference 2013 (Globecom 2013)*, 2013. Accepted.

[PPL13] PPLive, Inc. Pptv, June 2013. URL: `http://www.pptv.com/aboutus/en/`.

[Wan99] Jia Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, 1999.

[Win13] Peter Winzer. Optical transport is going mimo. In *2013 IEEE International Conference on Communications (ICC2013)*, June 2013. Keynote address.

[Zip32] George Kingsley Zipf. Selected studies of the principle of relative frequency in language. 1932.

CHAPTER 2

Energy Efficient Cache Provisioning

In this chapter we look into saving energy by optimizing the dimensioning of the network infrastructure. Dimensioning is the phase of design after deciding the connection structure, when the numbers and capacities of deployed devices are decided. The network is augmented by in-router caches. The caches are organized into a hierarchy, what has been previously discredited in the literature. Thus, in the first preliminary to the chapter, we discuss why hierarchy of caches can be beneficial, when taking into account traffic aggregation. In the second preliminary, we discuss the multiple possible approaches towards constructing power models. This is relevant both for this and the next chapter.

2.1 Preliminary: modelling content flow over a network

The real ISP network design that we take into consideration is divided into the core network, which ensures long-range connectivity, and multiple metropolitan networks, which cover geographical regions to give access to the clients. The core is a two-connected graph of some tens of nodes. Metropolitan network is comprised of two core routers, that are its connection to the core network, an optical ring consisting a number

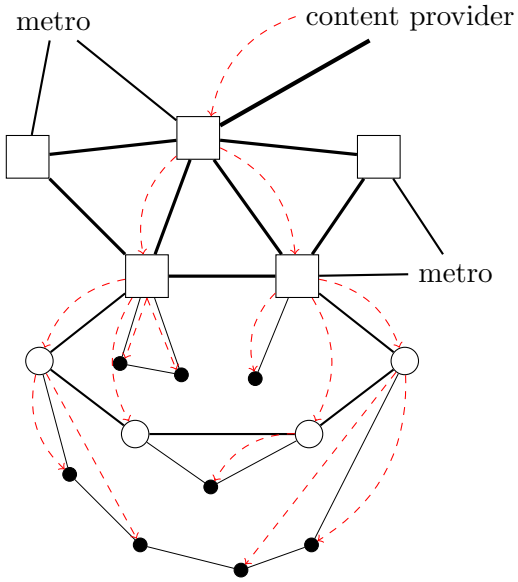


Figure 2.1: Example network comprising a core and three metropolitan networks, out of which one is displayed. Core routers are depicted with squares, metropolitan routers with circles and access nodes with solid dots. An example routing from a single content provider towards the visible access nodes is overlaid in red dashed lines.

of edge routers and some access nodes connected to each edge router, usually also in some two-connected arrangement. Optical bypass is often used in the metropolitan networks, creating a logical topology where nodes that are distant in the physical topology can have a direct connection in the network layer.

However, the setting becomes much simpler when considering media coming from a single content provider. Note that most media comes from big providers and enters the considered network through some peering point. By looking at shortest routes between this point and all clients, we obtain a tree over which the data is disseminated. This is depicted in Figure 2.1.

Looking at the trees obtained for the networks studied in this work, we found some distinctive levels. For example in the France Telecom network, we have a level of core routers that connect only to core routers, core routers that are edge to metropolitan networks, some more metropoli-

tan routers and access nodes. The fan-out of nodes within a level does not vary too much. Thus, we simplify the network as a rooted tree, where all nodes that have the same distance from root have the same degree. Note that while we have a tree for each possible content provider, all the trees are independent and have the same structure. Therefore, we treat them as a single aggregated tree, without affecting the results.

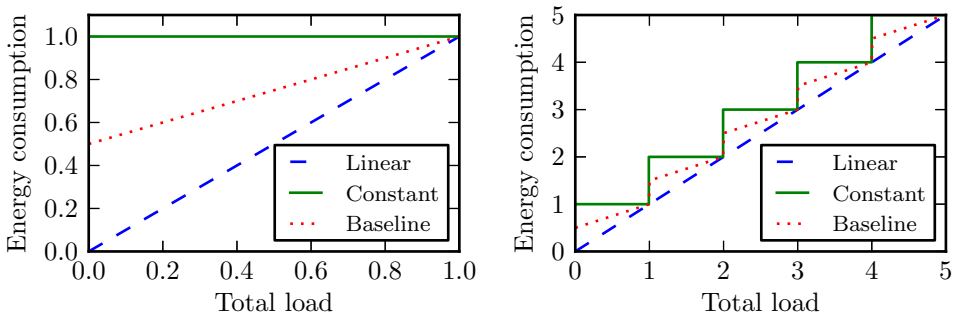
2.2 Preliminary: modelling energy consumption in a network

Expressing power consumption of a system as complex as a computer network in terms simple enough to be an optimization metric is not a trivial task. In a study focused on minimizing the number of active devices of single kind, it may be abstracted simply as the number of devices running. One example of such study is [GMM12], where the optimization metric is simply the number of links turned on.

However, assuming a device's power usage is constant is a simplification, which may be imprecise for some device types. Some modern electronics are known to switch to lower power consumption modes when under moderate load. One well known example of such a solution is CPU Throttling. Some devices may also enter low power mode on short inactivity. Overall, this promises that in future devices we will see power consumption approach proportionality to the load.

Nowadays a middle ground model is closer to reality. Whenever a device is turned on, it consumes a *baseline power*. This power is committed to spinning disks, fans and overall upkeep of an idle system. Figure 2.2 shows a comparison of these 3 models. Note that if we consider multiple devices sharing the same load, when the number of devices turned on is kept to a minimum, we approach the linear model, as shown on plot 2.2b. Thus, when considering a provisioning problem like in this chapter, the linear model can be an acceptable approximation.

Once the model considers multiple device types, or absolute figures on energy consumed are required, a need arises for knowing the actual characteristics. This is where databases like Powerlib [VHI12] come in handy. However, the data presented therein contains only capacities and producer rated peak power consumption. This allows using either constant or linear models, as described in the previous paragraph. A number



(a) One device

(b) Five devices

Figure 2.2: Three power models of a device, shown for one and five devices. The baseline power consumption is assumed to be half of the peak.

of measurement studies aimed at getting more insight were conducted, e.g. [VHILR+12] or [VLM+09a]. These studies show that baseline power usages, for a handful of current devices, tend to be over 80% of the maximum.

However, each such study is limited only to devices available to its authors; there are methodology differences between separate works. Furthermore, some of the numbers get outdated quickly. For example Solid State Drives are still in explosive growth phase, where each next generation is faster, bigger, cheaper and consumes less power. When looking into total server energy consumption for a unit of transfer, numbers found in the literature [VLM+09b, GAKG11] are over $200^{J}/_{Gb}$. However, for the results of this chapter we obtained current numbers from an innovative company [1], which turned out to be around $20^{J}/_{Gb}$. Therefore it is important to consider the power model carefully for each separate problem and attempt to obtain the most current data possible.

2.3 Preliminary: algorithmic approach

The general problem that underlies this section, Copy Placement, can be defined as follows.

[1] `http://www.cloudflare.com/`

Input We are given a digraph $G = (V, A)$ modelling the network, a set of files F and a set of demands D. Each file can be served from a source according to a function $s : F \to V$. Each demand $d \in D$ is characterized by the requesting vertex, a file identifier and request rate, $d = (v_d, f_d, r_d), v_d \in V, f_d \in F, r_d \in \mathbb{R}$. We are also given the cost functions for: transmission of a file over an arc $t : A \to \mathbb{R}$, placement of a copy of a file in a vertex $p : V \to \mathbb{R}$ and access to a copy of a file in a vertex $a : V \to \mathbb{R}$.

Output A solution consists of copy placement C and routing R. Copy placement assigns to each file $f \in F$ a subset of nodes $C_f \subset V$ in which copies of f are placed. Routing determines for each demand $d \in D$ a directed path in the digraph $R_d \subset A$, that begins in $w_d \in C_{f_d} \cup \{s(f_d)\}$ and ends in v_d.

Metric The solution should minimize the cost, determined by:

$$\sum_{f \in F} \sum_{v \in C_f} p(v) + \sum_{d \in D} \sum_{e \in R_d} r_d t(e) + \sum_{d \in D} \begin{cases} r_d a(w_d) \text{ if } w_d \neq s(f_d) \\ 0 \text{ otherwise} \end{cases} \tag{2.1}$$

Above problem is, in general digraphs, hard to compute. In fact we can show inapproximability with a simple reduction from the Set Cover problem, defined as follows.

Set Cover Given an universe $U = \{u_1, u_2, \ldots, u_n\}$, a collection of subsets of U, $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$, and a cost function $c : \mathcal{S} \to \mathbb{R}$, find a minimum cost subcollection of $\mathcal{S}$ that covers all elements of U.

Now, we show how to transform an instance of the Set Cover problem into an instance of Copy Placement. Take two sets of vertices, $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_k\}$, corresponding respectively to elements of $\mathcal{S}$ and U and a source vertex s. Put an arc from any vertex in X to one in Y if the corresponding element of Y belongs to the corresponding subset from $\mathcal{S}$ and an arc from s to every vertex in X. An example of such graph is shown in Figure 2.3. Let there be a single file served from the vertex s and requested once from each vertex in Y, with rate equal to one. Let $M > \sum_{S \in \mathcal{S}} c(S)$. For the cost functions, let $t(a) = M$ for every arc a from s and 0 for the other ones, $a(v) = 0$ for

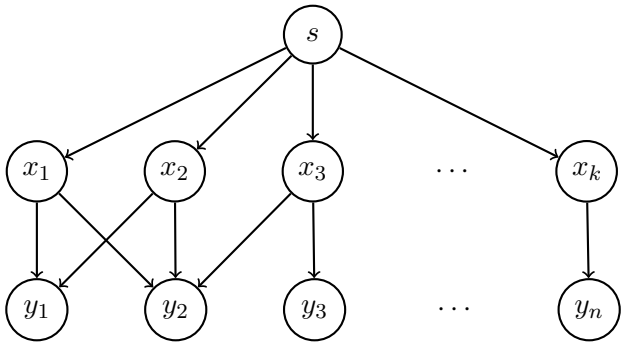


Figure 2.3: An example Copy Placement digraph obtained by transforming a Set Cover instance, where $\mathcal{S} = \{\{u_1, u_2\}, \{u_1, u_2\}, \{u_2, u_3\}, \ldots, \{u_n\}\}$.

any $v \in V$, and p be equal to the cost of corresponding elements of $\mathcal{S}$ for vertices in X and M for other vertices.

Consider an optimal solution to the instance of Copy Placement. For any demand $d \in D$, it can be routed either from v_d or s for a cost of M, or from a vertex $x \in X$ for a, possibly shared, cost of $p(x) < M$. Thus, in an optimal solution, each demand is routed from a vertex in X, that is a neighbour of requesting vertex. The solution is determined by placing copies of the file in a minimum cost subset of X that dominates Y. Choosing the corresponding subsets from $\mathcal{S}$ directly gives us an optimal solution of Set Cover with the same cost. Following the results in [AMS06], this can not be approximated within a ratio better than $O(\log |X|)$.

As explained in Section 2.1, for content distribution we can model network as a tree. The data is requested by the leaves and can be served by any node on the path between requesting leave and root. This has been solved in [LGI+99] using dynamic programming. However, the solution takes $O(|V|^3)$ time to place a single file. Thus, we make an additional assumption, that is in line with what we observed in Section 2.1: nodes on the same level (at the same distance from root) have an equal degree. An example of such a tree is shown in Figure 2.4. Data is requested according to a static distribution and we statically place data in the optimal levels. If a file i is stored at level j, the total cost of serving it to all the leaves is:

$$n_j o_j + r_i d_j, \tag{2.2}$$

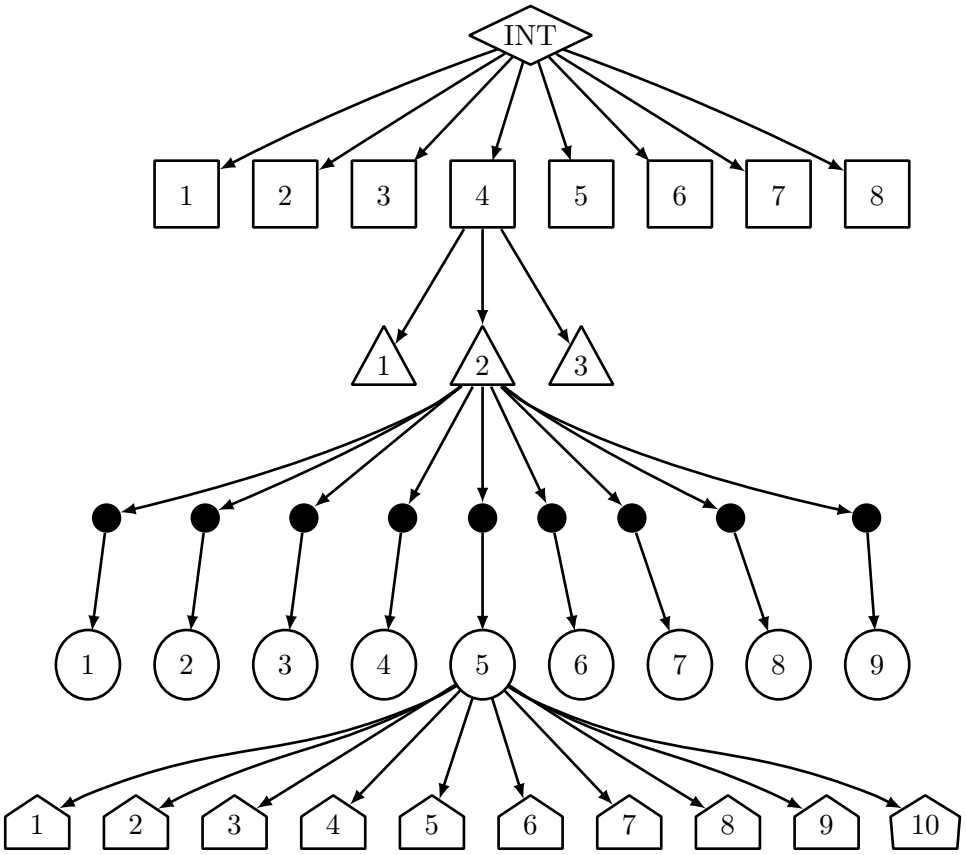


Figure 2.4: A tree representing the France Telecom network. All devices residing at the same distance from root of the tree have the same degree. For clarity of presentation, only a subset is drawn. The last level of the tree, representing consumers' premises, is omitted as there are 5000 such nodes per DSLAM node (the last level shown).

where n_j is the number of nodes on level j, o_j is the overhead cost to place the data in a node on level j, r_i is the total number of requests for file i and d_j is the cost of delivery from a device on level j to a leaf. In practice, d_j is nearly proportional to the distance between j and leafs and o_j is nearly constant. Thus, as we show in Section 2.8, the optimal placement depends on a file's popularity. We can efficiently compute for each level a request rate interval, such that files with rate within this interval are

optimally stored in each node in this level. This means that we can optimally place any number of files in time $O(h^2)$, where h is height of the tree. We define a simple algorithm to perform that placement. The main contribution of this study is determining practical values of all the factors of equation 2.2, obtaining the results and sensitivity analysis.

2.4 Publication

The remainder of this chapter corresponds to *Energy Efficient Content Distribution in an ISP Network* by R. Modrzejewski, L. Chiaraviglio, I. Tahiri, F. Giroire, E. Le Rouzic, E. Bonetto, F. Musumeci, R. Gonzalez and C. Guerrero, which is accepted for publication in the proceedings of IEEE Global Communications Conference 2013.

2.5 Introduction

The electricity consumption of the Information and Communication Technology (ICT) sector represents today almost 2% of the world electricity [LVHV+12], having observed an annual increase of 10% from 2007 to 2012. In this context, data centers and backbone networks will experience the highest energy consumption growth rates in the forthcoming years [LKWG11], due to the increase of traffic, especially for multimedia content. As an example, Cisco estimates that the sum of all forms of videos will represent 86% of the global consumer traffic by 2016 [Cis12]. In order to mitigate this trend, different solutions have been proposed in the literature for the design and the management of energy-efficient backbone networks (see [BBDC11] for an overview).

Recently, the problem of reducing power consumption in a backbone network by moving the contents accessed by users has attracted attention of the research community. In particular, in [CM10] we have studied the problem of reducing power consumption of an Internet Service Provider (ISP) and a Content Provider (CP) jointly, showing that considerable energy savings can be obtained when the CP and the ISP cooperate to minimize the total power consumption. In [LRH10] authors propose an architecture based on Content Centric Networking (CCN) to reduce the power consumption. Additionally, in [VLM+09b] an architecture based on home gateways forming a distributed data center infrastructure

managed by the ISP is proposed and evaluated. Finally, the energy trade-offs of an architecture based on immersive video centric services are evaluated in [LGAK12]. All these works prove that a huge amount of power is saved when the ISP takes control of the content and caches it considering the energy consumed to move the information across the network.

In this work we study the problem of reducing power consumption in an ISP network by considering the design of a content distribution infrastructure managed by the ISP. Our aim is to study where to cache content inside the network in order to reduce the overall power consumption of the system composed by the network elements and the installed storage. In current ISP networks a huge amount of traffic is exchanged between the users and the data centers owned by large CPs, such as as Google, Yahoo, Amazon, and Limelight. Normally, the data centers of large CPs are located close to the peering points of the ISP [GALM08].Therefore, the traffic originated from the data centers has to traverse a number of hops in the ISP network before reaching the users. We therefore study the optimal content caching inside the ISP, rather than sending the content from the data centers to the users. In our scenario, we consider a hierarchical logical topology composed of different levels (e.g. core, metro, and access), and we optimize the energy consumption by choosing the best level where to put each content.

The benefits of the energy-efficient design of content distribution architectures inside the ISP are multiple. First of all, it is possible to reduce jointly the electricity costs of the storage and of the network, as their energy is explicitly taken into account during the design phase. Secondly, the ISP reduces the amount of traffic that is exchanged across the network. This in turn may decrease the maintenance costs incurred by the ISP, since network elements are upgraded less frequently and less new switching devices need to be installed. Third, the monetary costs for sending/receiving information from outside the network are also reduced, since less bandwidth is required.

2.6 Related Work

There have been several works tackling the problem of web object efficient caching. Most of these works were not focusing on energy savings but

rather reducing the latency, the network traffic and/or the server load.

In [LDGS98], authors studied the optimal placement (for reducing either latency or network traffic) of M multiple web proxies among N potential sites under a given traffic pattern. They considered a simple path of $N+1$ nodes where the extremity of it corresponds to the original web server and the other nodes correspond to the potential sites. In case of no data replication, all the requests coming from the different sites would need to go until the main web server before being fulfilled. Dispatching some web proxies efficiently among the N potential sites can reduce the latency and the traffic load. Authors give an algorithm to find the best solution in $O(N^2M)$ time. In [LGI+99] they extended this algorithm to the case where the topology is a rooted tree, the main server is in the root and potential sites are all the other tree nodes. Again they show that the best solution can be found using a polynomial algorithm with time complexity $O(N^3M^2)$.

[BRS08] focuses on a more general case. Instead of having only one web object as in the previously cited papers, several objects should be simultaneously taken into account by the optimization. They considered also a set of caches that have a limited capacity and a cost for storing each of those objects. Finally they assume a set of clients that have demands for different objects and to each client they assign a cost for getting a specific object when it is stored in a specific cache. For efficiently solving this NP-complete problem, on a general topology, they presented a 10-approximation algorithm.

We want to mention also that some works investigated the possibility of managing the caches in a distributed scheme. In [TC02], authors propose a novel caching scheme that integrates both object placement and replacement policies and which makes caching decisions on all candidate sites in a coordinated fashion.

The closest papers to our work are [SK09, JNWC11, MLG+11]. In [SK09] authors detail an analytical model for caching considering the cost for transporting information and the cost for storing the content. However, the model is derived for a simple scenario (a metro network), with at most three levels in the topology as possible locations to cache content. Moreover, the evaluation of savings in terms of energy is not performed. In [JNWC11] authors propose a model for caching that integrates energy costs. The evaluation is performed considering five possible levels

for caching. Finally, in [MLG+11] an ILP model and two simple heuristics for the energy-efficient content distribution are detailed. However, authors do not consider the energy consumed for sending the content to the possible locations inside the network and a limited number of levels is also assumed.

In contrast to previous work, in this paper we go one step further by: a) defining a model with a generic number of levels and not only restricted to specific values or specific segments of the network, b) proposing an optimal algorithm to decide where to cache the content and compute the total energy consumption, c) evaluating the results over two case studies. Moreover, we consider the impact of the topology properties on the content caching, and we derive some insights for the design of future energy-aware networks.

The rest of the paper is organized as follows. In Sec. 2.7 we describe the problem. Sec. 2.8 details the algorithm we propose to solve the problem. Results are presented in Sec. A.5. Finally, Sec. 2.10 concludes our work.

2.7 Problem Description

We assume that the network is organized in a hierarchical structure composed of different levels. In particular, we assume a tree-like network to represent the collection of paths between each user and the Internet peering node. Nodes are grouped according to a hierarchy, and each level of the tree corresponds to a different level of the hierarchy. The content data is delivered towards the clients following a path on the tree from the root, i.e., the Internet peering point. A storage cache can be located at each node of the network, providing a potential facility for storing data. Moreover, caches are organized in a hierarchical structure: if a requested content is not available in a given cache, the request is forwarded to the parent cache of the hierarchy, without any collaboration among the caches located in the same level of the tree. Finally, we do not impose a given cache size, i.e., the cache size is an output of our approach.[2]

[2]In our scenarios the obtained cache size is always lower than the maximum capacity of current storage devices.

The content distribution procedure is divided in the following steps: a) the content is fetched from the peering point to the storage caches located at a given level of the tree, b) the content is cached for a fixed amount of time, c) during this period the content is retrieved by users, based on its popularity. We then associate an energy cost to each of these steps, and we compute the total energy consumption. Our aim is then to find the optimal amount of data to cache at each level of the tree in order to minimize the overall energy consumption.

Focusing on power requirements, we consider the cost of keeping the content stored in the cache, the cost of reading/writing the content from/to the cache and the cost of sending the content through one hop of the tree. We assume that the cost for traversing one hop is different for each level, due to the different switching devices deployed in each segment of the network [LRKH11]. In order to model the power consumption of each device, we assume a linear dependency with traffic volume, following the assumptions of previous works [BAHT09, FGK$^+$10, MLG$^+$11]. In particular, the cost of transporting information is expressed in terms of energy per bit, i.e., the total power consumption divided by the average throughput.

More formally, the set of levels in the network is $\mathcal{L} = \{1, \ldots, L\}$, $L = |\mathcal{L}|$ being the number of levels. The peering point is located at level 1, while the users are connected to level L (e.g., the DSLAMs). We denote the total number of switching devices located at level $j \in \mathcal{L}$ as N_D^j. Let us define the storage cost for a single cache as C_S. C_R is the cost of reading/writing content on one cache. C_H^j is the cost of traversing one node located at level j in the network. Moreover, we consider the characteristics of the content. We assume that the content is represented by videos watched by users. τ is the total throughput of videos requested by users. Let us denote the average video size as A and the popularity window duration as I. Thus, the total number of videos V_W watched during the popularity window is:

$$V_W = \frac{\tau I}{A} \tag{2.3}$$

Let us define V_S as the total number of videos provided by the CP. We divide the videos into classes according to their popularity, N_C being the number of classes. The set of classes is denoted as $\mathcal{K} = \{1, ..., N_C\}$. Class 1 is the most popular while class N_C is the least popular. We assume

that, on average, each class has the same number of videos, which we denote as $V_C = \frac{V_S}{N_C}$.

For each class $k \in \mathcal{K}$ we adopt the Zipf-based popularity model of [CRC+08] and compute the number of videos watched per class as

$$V_W^k = V_W \frac{k^{-\beta}}{\sum_{k=1}^{N_C} k^{-\beta}} \tag{2.4}$$

β being the parameter of the Zipf distribution.

We then compute the energy consumed for disseminating class k when it is stored on the caches located at level j. In particular, we first compute the energy consumed for fetching the content into the caches, and to keep the content stored:

$$\phi(j) = AV_C N_D^j \left(\sum_{z=1}^{j-1} C_H^z + C_R + C_S I \right) \tag{2.5}$$

The first term inside parentheses is the cost of traversing $(j-1)$ hops. The second term is the cost of writing the content on the cache. The third term is the cost of keeping the content stored, which is multiplied by the popularity window duration I since this cost has to be always accounted for the whole time period. All the costs are then multiplied by the amount of information that it is stored in level j, i.e., $A \times V_C \times N_D^j$. Note that $\phi(j)$ does not depend directly on the popularity of the class but only on the level j chosen for caching.

We then compute the energy consumed for retrieving the content as:

$$\varphi(j,k) = AV_W^k \left(C_R + \sum_{z=j}^{L} C_H^z \right) \tag{2.6}$$

In particular, we consider the cost of reading the content and the cost of sending the content from the caches at level j to users. The retrieved information corresponds to the videos that are watched during the popularity window duration, i.e, $A \times V_W^k$. Differently from $\phi(j)$, $\varphi(j,k)$ depends on both the class popularity and the level where the content is cached.

The total energy consumed for disseminating class k on level j is:

$$E_k^j = \begin{cases} \phi(j) + \varphi(j,k), & j > 0 \\ AV_W^k \sum_{z=1}^{L} C_H^z, & j = 0 \end{cases} \tag{2.7}$$

Note that level 0 is the special case where the data is served from the original source, i.e., caching is not exploited within the considered network. In this case, the total energy consumption is the cost of sending the watched videos directly from the peering point to the users.

The best level to store the videos of class k is then:

$$h_k = \underset{j \in \mathcal{L}}{\operatorname{argmin}} E_k^j \tag{2.8}$$

Note that the best level for each class is computed independently from the other classes. We therefore repeat this procedure for each class k.

The total energy consumption with caching is computed as:

$$T = \sum_{k \in \mathcal{K}} E_k^{h_k} \tag{2.9}$$

Which we can compare to the energy consumption without caching:

$$T' = \sum_{k \in \mathcal{K}} E_k^0 \tag{2.10}$$

By comparing T with T', we can estimate if caching is effective or not in saving energy. However, computing Eq.(2.8) for each class is not feasible, since the iteration over the levels has to be repeated for all the classes, resulting in a time complexity of $\mathcal{O}(L \times N_C)$. To solve this issue, we have proposed a new algorithm in order to efficiently compute T.

2.8 GCT Algorithm Description

We first detail the properties that we have exploited to design our algorithm. In particular, since there is no limit on the storage, we can choose the best level for every video class independently from the others. Moreover, for level j that is optimal for the video class k, we have necessarily $E_k^j \leq E_k^{j'}$ for any j' different from j. This implies:

$$\varphi(j,k) - \varphi(j',k) \leq \phi(j') - \phi(j). \tag{2.11}$$

In addition to that, when two video classes are stored in the same level, less videos will be retrieved from this level for the less popular class.

Namely, if a class k' is less popular than another class k, then: $\varphi(j,k') \leq \varphi(j,k)$ and $\varphi(j',k') \leq \varphi(j',k)$. This leads to the following inequality:

$$\varphi(j,k') - \varphi(j',k') \leq \varphi(j,k) - \varphi(j',k). \tag{2.12}$$

The two previous equations imply:

$$\varphi(j,k') - \varphi(j',k') \leq \phi(j') - \phi(j). \tag{2.13}$$

And hence we get the following property.

Property 1. *Let k and k' be two video classes such that k-class videos are more popular than k'-class ones ($k' > k$) and let j be the optimal level for k. Then for every level j' lower than j ($j' < j$) we have $E^j_{k'} \leq E^{j'}_{k'}$.*

The intuition of the Green Content Threshold (GCT) algorithm is to restrict the evaluation of Eq. (2.8) to specific k, which we call thresholds. A threshold is defined as the last class to be stored at level j, before starting storing in another level x (with $x < j$). The rule for deciding when to pass from one level to another one is based on the energy consumption E^j_k (recall that level 0 correspond to the case without caching). In particular, we find the class index $k = \bar{k}(j,x)$ that verifies, for arbitrary levels j and x the following equality:

$$E^j_k = E^x_k \tag{2.14}$$

For some classes, the caching of their videos in level j is preferred to caching them in level x in term of energy efficiency; and for some other classes it is the opposite. $\bar{k}(j,x)$ is the index that separates both set of classes. In fact, Eq. (2.14) being verified by $\bar{k}$ implies:

$$\varphi(x,\bar{k}) - \varphi(j,\bar{k}) = \phi(j) - \phi(x). \tag{2.15}$$

On the other side $k \leq \bar{k}$ iff:

$$\varphi(x,\bar{k}) - \varphi(j,\bar{k}) \leq \varphi(x,k) - \varphi(j,k). \tag{2.16}$$

And this leads to the following property.

Property 2. *Let x and j be two levels such that $x < j$ and let $\bar{k}$ be the solution to $E^j_k = E^x_k$. Then $k \leq \bar{k}$ iff $E^x_k \geq E^j_k$.*

Algorithm 1 Pseudo-Code of the best threshold selection procedure.

Input: threshold matrix $\overline{K}$, number of levels L, number of classes N_C;
Output: array of best thresholds B

```
curr_level = L;
B [0]=N_C- 1;
while curr_level != 1 do
   min_thre=inf;
   for upper_level = 1:curr_level-1 do
      curr_thre = K̄ [curr_level,upper_level];
      if curr_thre < min_thre then
         min_thre=curr_thre;
      end if
   end for
   B [curr_level]=min_thre;
   curr_level=curr_level-1;
end while
```

To compute $\bar{k}(j,x)$ we solve Eq.(2.14) using Eq.(2.7), obtaining:

$$\bar{k}(j,x) = \begin{cases} \left[\frac{V_W(\sum_{z=x}^{j} C_H^z)}{(\Delta_f(j,x)) \sum_{k=1}^{N_C} k^{-\beta}} \right]^{\frac{1}{\beta}}, & j > 0 \\ \left[\frac{V_W(\sum_{z=x}^{j} C_H^z - C_R)}{f(j) \sum_{k=1}^{N_C} k^{-\beta}} \right]^{\frac{1}{\beta}}, & j = 0 \end{cases} \tag{2.17}$$

with $f(j) = N_D^j \left(\sum_{z=1}^{j} C_H^z + C_R + C_S I \right)$ and $\Delta_f(j,x) = f(x) - f(j)$.

The matrix with elements $\bar{k}(j,x)$ is denoted as $\overline{K}$. Each element of this matrix represents a threshold class for moving from one level to another one.

The algorithm that we propose is then divided into three steps: a) computation of the thresholds matrix, b) best threshold selection, and c) computation of total power consumption.

The first step is performed by computing $\overline{K}$ with Eq.(2.17) for each $j \in \mathcal{L}$ and each $x < j$. In the next step, we select the best threshold for each level, by adopting the procedure reported in Alg.1. The function takes as input the matrix $\overline{K}$, the number of levels L, and the number of classes N_C. The array of best thresholds B is produced as output. The

algorithm searches the best thresholds from the lower levels (i.e., the access nodes) to the upper ones. In particular, the minimum threshold is selected by evaluating $\bar{k}(j,x)$ from the current level to each upper level (lines 5-10): in fact, thanks to property 1, we can ignore levels that are lower than the current level. Moreover, we know, thanks to property 2, that for every video that has a popularity rank lower than the minimum threshold it is better to cache it in current level than in any higher level. The procedure is repeated until the last level is reached (line 3). It is clear that when the values of $\overline{K}$ are computed optimally, this algorithm is optimal.

We then detail how the total energy consumption is computed from the best thresholds. We first derive the energy consumption consumed at level j. This term includes the energy consumption corresponding to the classes that are assigned to the current level j, i.e., $b_{j-1} < k \leq b_j$ (b_{j-1} and b_j being elements of the array of the best thresholds B), which can be expressed as:[3]

$$\sum_{k=(b_{j-1}+1)}^{b_j} E_j^k = [b_j - b_{j-1} + 1]\phi(j) + \sum_{k=(b_{j-1}+1)}^{b_j} \varphi(j,k) \tag{2.18}$$

In particular, from the definition of $\varphi(j,k)$ the last term can be expressed as:

$$\sum_{k=(b_{j-1}+1)}^{b_j} \varphi(j,k) = AV_W N_D^x \left(C_R + \sum_{z=j}^{L} C_H^z \right) \frac{\sum_{k=(b_x+1)}^{b_j} k^{-\beta}}{\sum_{k=1}^{N_C} k^{-\beta}} \tag{2.19}$$

The exponential terms can be expressed as:

$$\sum_{k=a}^{c} k^{-\beta} = \zeta(\beta, a) - \zeta(\beta, c+1) \tag{2.20}$$

ζ being the Hurwitz zeta function [Vor03]. To compute the total energy consumption T, the algorithm solves Eq.(2.18)-(2.20) for all the levels, and the sum of the energy consumption is stored in T.

GCT has a time complexity that depends on the time needed to evaluate $\sum_{k=a}^{c} k^{-\beta}$ for $0 \leq a \leq c \leq N_C$. When this time is bounded by $\mathcal{T}$, the

[3] A similar expression can be derived for the $j = 0$ case (with $b_{-1} = 0$).

complexity is $\mathcal{O}(L^2 + L \times \mathcal{T})$. Since there are efficient ways to approximate ζ, we can have a good approximation of $\sum_{k=a}^{c} k^{-\beta}$ and of GCT with a low time complexity. In particular, if there exists a ρ-approximation of ζ which has a time complexity $\mathcal{T}_\rho$, then we can get a ρ-approximation on the optimal solution (the minimum energy consumption induced by all classes) that have a time complexity $\mathcal{O}(L^2 + L \times \mathcal{T}_\rho)$ which is better than the original approach since normally $L << N_C$ and $\mathcal{T}_\rho << N_C$.

2.9 Results

We have implemented the GCT algorithm in Python. In particular, we have adopted the `mpmath` library for an efficient computation of the Hurwitz Zeta function ζ.[4] We have then evaluated GCT over two realistic networks of national ISPs, namely France Telecom (FT) and an ISP in Morocco. The main features of the networks, together with the setting of the main parameters, are reported in Tab. 2.1. Both networks are composed of six levels in total (core, core-regional, metro-core, metro, access-metro, access), and with a different number of switching devices deployed at each level.

Focusing on power requirements, the cost of storage C_S is taken from [OCZ]. The cost of reading/writing the cache is provided privately by CloudFlare Inc [Clo], based on their global network of content caches. This cost may be a slight overestimation of what the cost in our model should be, as it already accounts for storage energy consumption. Furthermore, computational overhead falls with the size of objects cached and we propose to cache videos, which would be in the order of hundreds of megabytes, whereas for CloudFlare only 0.4% objects exceed 1MB. The costs of network hops are based on equipment datasheets, measurements and operational conditions published in [VHILR+12] and provided by France Telecom.

We then consider the characteristics of the content. In particular, we assume that for the FT scenario traffic forecasts are provided for the year 2020. We refer the reader to [RLB12] for a detailed description on how these forecasts are obtained. On the contrary, for the Moroccan scenario we set values in accordance to the current traffic measured over

[4] The complexity of the implemented Hurwitz function is in the order of $\mathcal{O}(p^{2+\epsilon})$, p being the precision (the number of significant bits) and ϵ a small number.

Table 2.1: Main parameters for the considered scenarios.

Parameter	FT	Moroccan
L [units]	6	6
N_D^j [units]	[1 8 24 216 216 2160]	[1 20 20 20 200 10000]
C_S [W/Gb]	$9.375 \cdot 10^{-4}$	$9.375 \cdot 10^{-4}$
C_R [J/Gb]	24.3	24.3
C_H^j [J/Gb]	[12.5 25 30 35 200 300]	[12.5 25 30 35 200 300]
τ [Gb/s]	$8 \cdot 10^3$	10^3
I [days]	7	7
A [Gb]	15	0.6
β [units]	0.8	0.8
V_S [units]	$120 \cdot 10^6$	$120 \cdot 10^6$
N_C [units]	V_S	V_S

the network. As a consequence, the total video throughput τ is eight times higher in the FT scenario compared to the Moroccan one. To this extent, we have also considered different values for the average video size A, assuming for the FT scenario a value that corresponds to a high definition video provided today on optical disks. Moreover, we have assumed a popularity duration of one week,[5] and a value for the exponent of the Zipf distribution from [CRC+08] for both scenarios. Finally, we assume a number of videos of a typical video CP, and one video for each class.

Evaluation Metrics We describe the metrics adopted to evaluate the performance of our algorithm. We first derive the energy saving as:

$$S = \frac{T' - T}{T'} \tag{2.21}$$

We then compute the percentage of bandwidth that is saved at the

[5]Even though the popularity of the watched videos can actually change during this time period, the popularity of the most viewed videos is almost constant. As an example, [CRC+08] shows that the popularity of the 50 most viewed videos does not consistently vary over the days. In this work, we are interested in the most popular videos, as these contents are cached inside the ISP network.

Table 2.2: Summary of results for the two network scenarios

Metric		FT	Moroccan
Energy savings (S)		8.7%	11.0%
Yearly monetary savings [k€]		769	122
Bandwidth savings (Ψ)		18.2%	30.2%
Cache Size [GB]	Λ^1	0	0
	Λ^2	0	0
	Λ^3	32546	0
	Λ^4	0	23510
	Λ^5	35878	5581
	Λ^6	2041	46
Cache Bandwidth [Mbps]	Θ^1	0	0
	Θ^2	0	0
	Θ^3	7907	0
	Θ^4	0	4550
	Θ^5	2946	721
	Θ^6	290	6

peering point when caches are exploited:

$$\Psi = \frac{\tau I - \sum_{k:\{h_k=0\}} AV_W^k - \sum_{k:\{h_k>0\}} AV_C N_D^{h_k}}{\tau I} \tag{2.22}$$

In particular, τI is the total amount of information flowing through the peering point without caching; $\sum_{k:\{h_k=0\}} AV_W^k$ is the amount of watched videos not stored inside the network in the case with caching; $\sum_{k:\{h_k>0\}} AV_C N_D^{h_k}$ is the total amount of information initially fetched inside the caches.

Finally, we consider the cache size for a device in level j, defined as:

$$\Lambda^j = \sum_{k:\{h_k=j\}} AV_C \tag{2.23}$$

General Analysis Tab. 2.2 reports the results for the two scenarios obtained with the GCT algorithm. We first consider the energy savings compared to the case in which caching is not exploited. Energy savings of almost 9% and 11% are possible for the FT and the Moroccan scenarios, respectively. By assuming that caches are refreshed once a week for

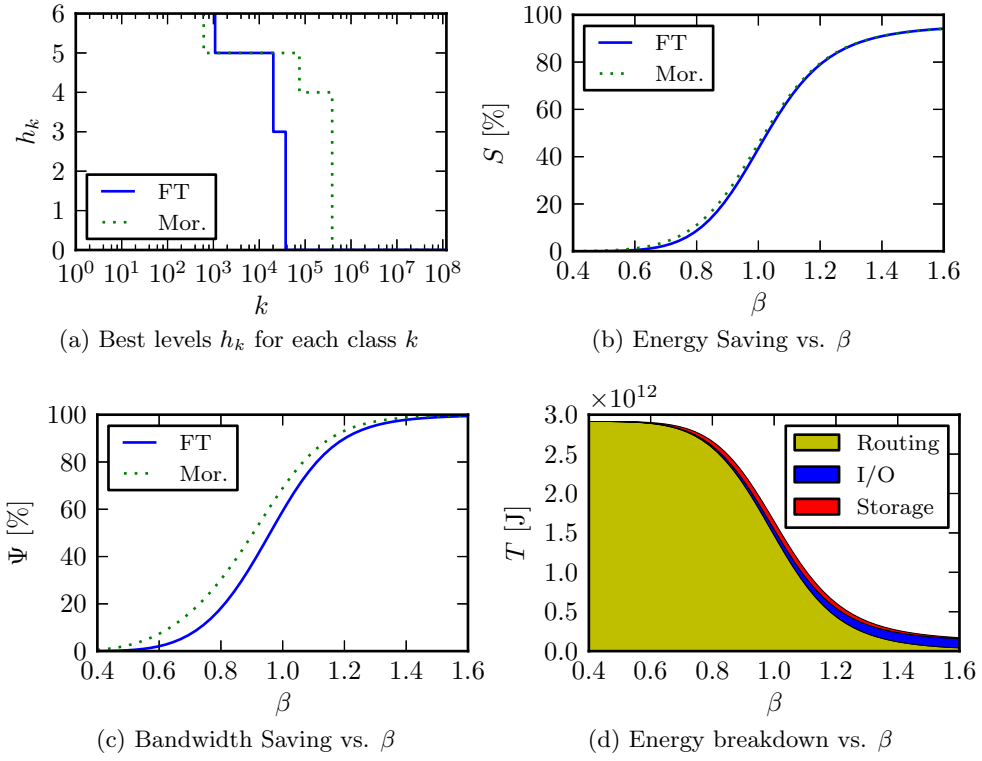
an entire year, we have estimated monetary savings[6] of more than 700 k€ for FT, and more than 100 k€ for the Moroccan network. Moreover, the savings in terms of bandwidth saved at the peering point are even larger, reaching 18% for the FT scenario and 30% for the Moroccan one.

The table also reports the cache size Λ^j per device for each level j. Interestingly, Λ^j is at most 36 TB, a value that can be covered by a commercial array of disk drives. Moreover, the capacity requirements tend to decrease moving closer to users, with at most 2041 GB of storage required at the access level for the FT network and only 46 GB for the Moroccan one. This is due to the fact that the number of switching devices per level increases when moving from the core to the access, hence the cost of increasing cache size is much higher than in higher levels and outweighs the gains faster. Finally, the table reports the average required bandwidth Θ per level. In this case, up to 7.9 Gbps and 4.5 Gbps are required for the FT and the Moroccan networks, respectively.

To give more insight, Fig. 2.5a reports the best level h_k for each class k for the two scenarios. k ranges between 1 and V_C. The levels on the left are the most popular ones and hence, to minimize the costs of moving the information frequently from the cache to users, it is better to store these classes in the closest level to users, i.e., the access part of the network. Moving from left to right, the popularity decreases, and therefore the classes are stored in the inner levels of the topology (metro and core). At last, very unpopular classes are assigned to level 0, i.e., they are not cached at all. Interestingly, the percentage of the total number of stored classes is around 1.7% and 0.5% for the Moroccan and the FT networks, respectively. Thus, we can conclude that with the considered power and popularity models, the ISP needs to store a little amount of content information to achieve energy and bandwidth savings. This is an encouraging result showing that caching not only has benefits on QoS and customer experience, but it can also lead to a better management of the ISP power consumption.

Impact of Content Characteristics We then consider how much the characteristics of the content impact the energy and the bandwidth savings. We first vary one parameter per time, keeping the others with the default values reported in Tab. 2.1.

[6] We have assumed an electricty cost of 0.21€/kWh.



(a) Best levels h_k for each class k

(b) Energy Saving vs. β

(c) Bandwidth Saving vs. β

(d) Energy breakdown vs. β

Figure 2.5: Best levels for each class k (a) and impact of the variation of β (b-d).

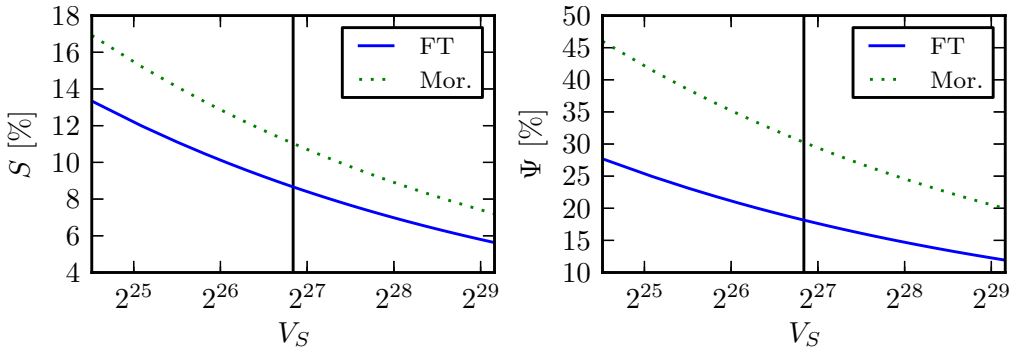
We start considering the variation of β, as reported in Fig. 2.5b and Fig. 2.5c. With low values of β (left part of the figures) all the classes tend to have a similar popularity. Intuitively, there is not a huge benefit in terms of energy in storing these classes inside the network, since the cost of storing this amount of information would be huge. On the contrary, when β takes higher values (right part of the figures), the variation on the popularity also increase. This means that few classes have a very high popularity, while most of the content is seldom accessed by users. This in turn imposes to store the most popular classes close to users, and therefore both the energy and the bandwidth savings steadily increase. At last, when β =1.6, the bandwidth savings are almost equal to 100%, while energy savings are more than 90%. This corresponds to the case in which the most popular contents are cached in the last level of the

network (i.e., the access part), and the main cost incurred by the ISP is to transfer these contents from the caches to the users. Thus, we can conclude that β greatly influences the performance in the network.

To give more insight, Fig.2.5d reports the breakdown of energy for the FT scenario considering: a) the energy consumed to route the content inside the ISP network, either from the peering point to the cache or from the cache to users ('Rout.' label), b) the energy consumed for reading/writing the content from/to the caches ('I/O' label), c) the energy consumed for keeping the content stored ('Stor.' label). For β =0.4 the caching is not exploited, and therefore the largest amount of energy is due to the routing, i.e., the cost of moving information inside the network. However, as β increases, the routing energy steadily decreases, since caches are more frequently used. This in turn implies that the energy consumed for reading and writing the content on the caches increases. However, the total energy consumption is always decreased, producing high energy savings. Finally, we can distinguish three different zones for characterizing the evolution of the energy consumed by storing. The first one ($\beta \approx 0.4$) corresponds to the case in which contents are not stored in the ISP, and therefore their storage costs is zero. On the contrary, when $\beta \approx 1.6$ most of users watch a very limited number of videos, whose storing cost is almost zero again. However, for intermediate popularity values ($\beta \approx 1.0$) the storing energy is not negligible, since a greater amount of videos is frequently watched by users.

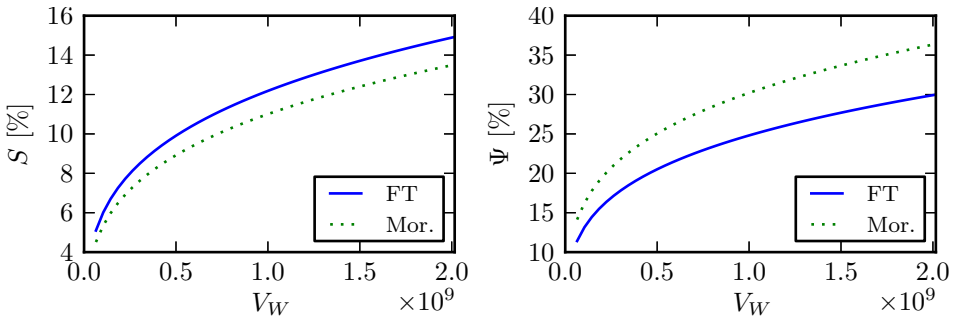
We then consider the impact of the number of videos in the collection V_S. Fig. 2.6 reports the energy and bandwidth savings. The reference values of Tab. 2.1 are reported as vertical lines. When V_S decreases, the savings tend to steadily increase. In this case, V_S is decreasing, while the actual number of watched videos V_W is kept constant. Thus, the gain introduced by caches increases, i.e., few videos frequently viewed by users. On the contrary, when V_S increases, the saving tend to decrease, since the efficiency of adopting the caches is reduced.

In the following, we consider the variation of the total number of videos watched V_W, reported in Fig. 2.7. When V_W increases (right part of the figures), both the energy and the bandwidth saving increase. This is due to the fact that as V_W increases the caches are more frequently accessed by users, and therefore the introduced gain in terms of energy and bandwidth is higher. Clearly, when the users seldom access the



(a) Energy Saving (b) Bandwidth Saving

Figure 2.6: Impact of the total number of videos V_S.



(a) Energy Saving (b) Bandwidth Saving

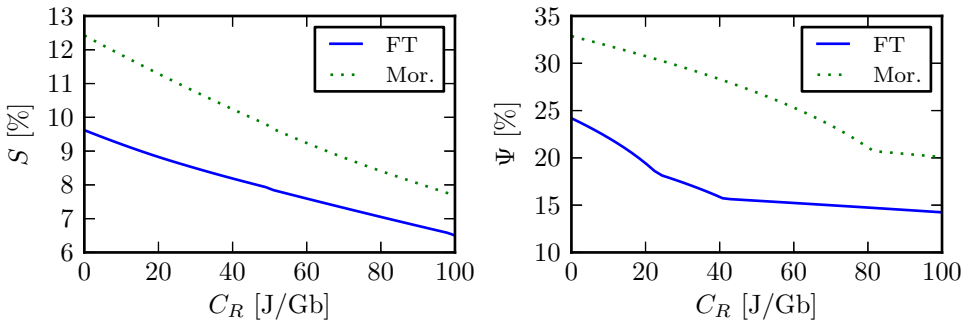
Figure 2.7: Impact of the total number of videos watched V_W.

content (left part of the figures) there is no need to put caches.

Finally, we have applied a variance-based sensitivity analysis [SRA+08] to precisely characterize the relative impact of the parameters. In particular, we have considered how much the variance of the energy saving S is impacted by the variation of the content parameters. To this end, we have considered the first order index and the total effect index. The first order index takes into account how much the variance of a single parameter influences the variance of the output. On the contrary, the total effect index takes into account the effects of varying the parameter on the model's output, including all the variances from interaction with the other parameters. To compute both indexes, we have adopted

Table 2.3: Variance Decomposition of S for the Content Parameters (FT Scenario)

Parameter	First Order Index	Total Effect Index
β	0.9950	0.9988
τ	0.0004	0.0007
V_S	0.0001	0.0002
A	0.0000	0.0000



(a) Energy Saving (b) Bandwidth Saving

Figure 2.8: Impact of the cost for reading and writing C_R.

a Monte Carlo method. In particular, we have generated a pool of 18 million samples, in which each parameter take a random value in the interval $[1/2, 2]$ w.r.t. the standard values reported in Tab. 2.1.

Tab. 2.3 reports the computed indexes considering the total energy savings. The largest contribution to the output variance of the first order index is due to β, while the other parameters play a minor role. Moreover, the average video size does not impact the energy savings since this parameter is simplified when computing S. The table also reports the values of the total effect index. These values are very similar to the ones of the first order index. This shows that simultaneous varying multiple input variables does not have a strongly amplified (multiplicative) effect on energy savings when compared to varying them separately.

Impact of Power Consumption Models We first consider the variation of the cost for reading and writing C_R. Fig. 2.8 reports the saving in terms of energy and bandwidth. As expected, the savings are increasing when C_R decreases. Intuitively, the lower is the cost for

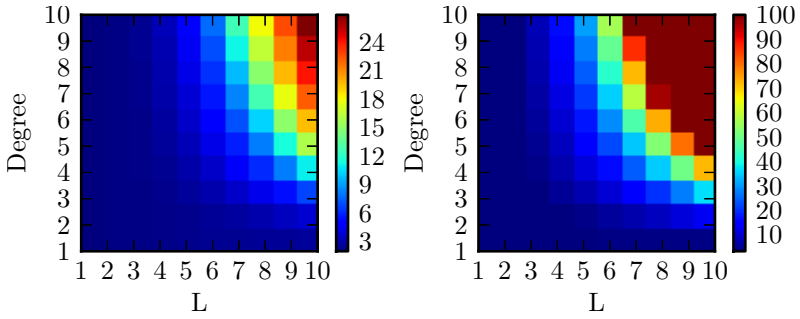
Table 2.4: Variance Decomposition of S for the Power Consumption Parameters (FT Scenario)

	First Order Index	Total Effect Index
C_S	0.2738	0.2699
C_R	0.0303	0.0265
C_H^1	0.1942	0.2009
C_H^2	0.3062	0.3125
C_H^3	0.0367	0.0436
C_H^4	0.0152	0.0210
C_H^5	0.0066	0.0143
C_H^6	0.1251	0.1259

reading and writing information, the higher is the gain introduced by caching. In particular, when $C_R \approx 0$ energy savings of more than 9% and 12% are possible for the FT and the Moroccan networks, respectively. Thus, we can expect that, if the energy efficiency of caches improves faster than the one of transport equipment, the benefit introduced by caching will be greater in the future.

Finally, we have performed the variance decomposition analysis also for the power consumption parameters, adopting the same procedure as in the previous subsection. Tab. 2.4 summarizes the main results for the FT scenario. Interestingly, the energy savings are greatly impacted by the cost of storing the content C_S. This is due to the fact this term has to be counted for all the time periods and for all the caches, thus at the end its contribution is not negligible. Thus, it is very important to deploy energy efficient storage inside the ISP to obtain energy savings. Moreover, the energy costs in the first levels are also impacting the energy savings, since most of traffic reduction occurs in these levels.

Impact of Network Properties To give more insight, we have considered a network with the same degree for all levels, and we have studied the impact of the variation of the degree and the number of levels. In this way, we are able to study the impact of caching over a set of topologies. In particular, the degree of level j is defined as the average number of links connecting a device in level j with the devices in level $(j+1)$. Moreover, we assume a number of video requests proportional to the degree and the number of levels. Specifically, we have assumed 5000



(a) Energy Saving [%] (b) Bandwidth Saving [%]

Figure 2.9: Impact of the number of levels and the average degree.

users connected for each device in the last level (i.e., the access one)[7], each of them watching 3 videos of size $A = 15$ Gb during a popularity window duration I of one week. Additionally, we have set $V_S = 120 \times 10^6$. Focusing on power requirements, we have set $C_S = 9.375 \times 10^{-4}$ W/Gb, $C_R = 24.3$ J/Gb,$C_H = 25$ J/Gb for each level j, respectively. Fig. 2.9 reports the energy and the bandwidth savings. Interestingly, the savings increase with the number of levels and the average degree. In particular, energy savings of more than 20% are possible for very large networks. Moreover, bandwidth savings quickly approach 100%. The fact that energy and bandwidth savings increase with the degree and the number of levels L is due to two main reasons: i) increased cost of moving information inside the network when the number of levels is increased, and ii) total number of watched video increased, while total number of stored videos kept constant.

2.10 Conclusions

We have studied the energy-efficient design of a content architecture in a ISP network, by exploiting caches managed by the ISP. Our results indicate that caching brings substantial savings in terms of energy and bandwidth.

[7]Note that while the number of user per device is constant, the total number of users scales with the degree and the number of levels L.

As next steps, we will consider the joint management of the content distribution architecture. In particular, our aim is to study the traffic variation over time and to compute the best set of caches powered on to satisfy a given traffic demand, while leaving the remaining caches powered off. Another possible direction is to introduce cooperation between neighboring caches to serve users and to reduce the amount of stored information. To better fit real topologies, we can study the case where devices on the same level of the tree have different degrees. Finally, we plan to study the impact of considering more than one peering node and the impact of introducing realistic traffic matrices inside the ISP.

2.11 Addendum: cache hierarchies and the filter effect

In this chapter we have studied a hierarchy of caches with popularity following a power law. Most of the gain is achieved by storing only the few most popular objects. When multiple caches are queried in a row, the big gain is achieved in the first one, leaving the others with much less popular objects. This is called the *filter effect* and has been studied in [Wil02]. Figure 2.10 shows results of a trace-based simulation. Three caches of the same size have been put one after another, storing any passing object and evicting the least recently used one, asking another

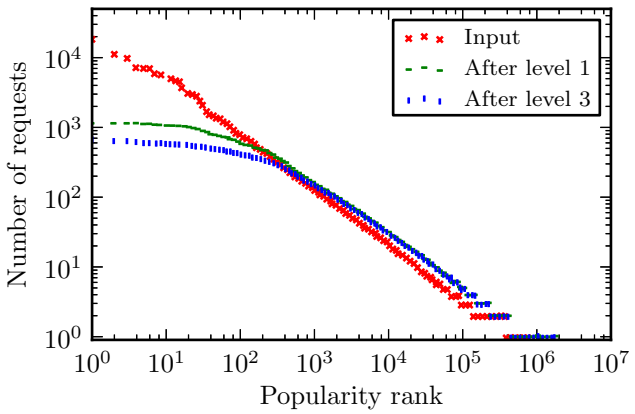


Figure 2.10: Object popularity in a hierarchy of caches, as observed in [Wil02].

cache, or the original source in case of third cache, in case of a cache miss. What we can see from the figure, is that the first cache observed an order of magnitude more requests for the most popular objects. This is reflected by the hit rates, which were respectively 19.66%, 2.05% and 0.94%. Intuitively, this can raise doubts if the results presented in this study do not stand in contradiction.

However, the above example differs from what we presented in the study. In the example there are three levels of caches, a single source of traffic and a single receiver. In a more practical setup, cache hierarchy would not be a path. A cache on a higher level should aggregate requests that were missed by several lower level caches. Therefore, the traffic seen by a cache grows exponentially with the distance in the hierarchy from the clients.

As explained in Section 1.4, the hit ratio does not depend on the volume of traffic. However, as we have seen in this chapter, just the raw number of cache hits can be enough to make such an aggregated cache beneficial (aggregating overhead costs makes up for the lower hit rate).

The same effect cannot be exploited easily in Chapter 3, due to assumption of all-to-all traffic in a core network. Therefore, it is assumed there that at most one cache is queried for a request and after a miss the original source is reached.

2.12 Bibliography

[AMS06] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177, 2006.

[BAHT09] J. Baliga, R. Ayre, K. Hinton, and R.S. Tucker. Architectures for energy-efficient iptv networks. In *Optical Fiber Communication Conference*, San Diego, California, 2009.

[BBDC11] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti. Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures. *Communications Surveys & Tutorials, IEEE*, 13(2):223–244, 2011.

[BRS08] Ivan Baev, Rajmohan Rajaraman, and Chaitanya Swamy. Approximation algorithms for data placement problems. *SIAM J. Comput.*, 38(4):1411–1429, August 2008.

[Cis12] Cisco. Visual networking index: Forecast and methodology, 2011-2016, 2012.

[Clo] Cloud Flare Inc. URL: `https://www.cloudflare.com/`.

[CM10] L. Chiaraviglio and I. Matta. Greencoop: cooperative green routing with energy-efficient servers. In *Proceedings of the 1st ACM International Conference on Energy-Efficient Computing and Networking*, pages 191–194, Passau, Germany, 2010.

[CRC+08] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain. Watching television over an IP network. In *ACM IMC '08*, pages 71–84, Vouliagmeni, Greece, 2008.

[FGK+10] Anja Feldmann, Andreas Gladisch, Mario Kind, C Lange, G Smaragdakis, and F-J Westphal. Energy trade-offs among content delivery architectures. In *IEEE CTTE '10*, pages 1–6, 2010.

[GAKG11] Kyle Guan, Gary Atkinson, Daniel C. Kilper, and Ece Gulsen. On the energy efficiency of content delivery architectures. In *IEEE International Conference on Communications Workshops (ICC)*, pages 1 –6, Kyoto, Japan, june 2011.

[GALM08] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *Passive and Active Network Measurement*, pages 1–10, Cleveland, USA, 2008. Springer.

[GMM12] F. Giroire, D. Mazauric, and J. Moulierac. Energy efficient routing by switching-off network interfaces. *Energy-Aware Systems and Networking for Sustainable Initiatives*, pages 207–237, 2012.

[JNWC11] C. Jayasundara, A. Nirmalathas, E. Wong, and C.A. Chan. Energy efficient content distribution for VoD services. In *Optical Fiber Communication Conference*, pages 1–3, Los Angeles, USA, 2011.

[LDGS98] B. Li, X. Deng, M.J. Golin, and K. Sohraby. On the optimal placement of web proxies in the internet: The linear topology. pages 485–495, 1998.

[LGAK12] J. Llorca, K. Guan, G. Atkinson, and D.C. Kilper. Energy efficient delivery of immersive video centric services. In *IEEE INFOCOM*, pages 1656–1664, Orlando, USA, 2012.

[LGI+99] Bo Li, M.J. Golin, G.F. Italiano, Xin Deng, and K. Sohraby. On the optimal placement of web proxies in the internet. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1282 –1290 vol.3, mar 1999.

[LKWG11] C. Lange, D. Kosiankowski, R. Weidmann, and A. Gladisch. Energy consumption of telecommunication networks and related improvement options. *Selected Topics in Quantum Electronics, IEEE Journal of*, 17(2):285–295, 2011.

[LRH10] U. Lee, I. Rimac, and V. Hilt. Greening the internet with content-centric networking. In *Proceedings of the 1st ACM International Conference on Energy-Efficient Computing and Networking*, pages 179–182, Passau, Germany, 2010.

[LRKH11] U. Lee, I. Rimac, D. Kilper, and V. Hilt. Toward energy-efficient content dissemination. *Network, IEEE*, 25(2):14–19, 2011.

[LVHV+12] S. Lambert, W. Van Heddeghem, W. Vereecken, B. Lannoo, D. Colle, and M. Pickavet. Worldwide electricity consumption of communication networks. *Optics Express*, 20(26):513–524, 2012.

[MLG+11] U. Mandal, C. Lange, A. Gladisch, P. Chowdhury, and B. Mukherjee. Energy-efficient content distribution over

telecom network infrastructure. In *13th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, Stockholm, Sweden, 2011.

[OCZ] OCZ Technology Group. URL: `http://www.ocztechnology.com/ocz-revodrive-3-x2-pci-express-ssd.html`.

[RLB12] Olivier Renais and Jacques Le Briand. Tracks for transport network architecture optimization. In *NETWORKS 2012*, pages 1–6, Rome, Italy, 2012.

[SK09] L.B. Sofman and B. Krogfoss. Analytical model for hierarchical cache optimization in iptv network. *Broadcasting, IEEE Transactions on*, 55(1):62–70, 2009.

[SRA+08] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis: the primer*. Wiley-Interscience, 2008.

[TC02] Xueyan Tang and Samuel T. Chanson. Coordinated en-route web caching. *IEEE Trans. Comput.*, 51(6):595–607, June 2002.

[VHI12] W. Van Heddeghem and F. Idzikowski. Equipment power consumption in optical multilayer networks-sour ce data. Technical report, Technical Report IBCN-12-001-01 (January 2012), available at http://powerlib. intec. ugent. be, 2012.

[VHILR+12] Ward Van Heddeghem, Filip Idzikowski, Esther Le Rouzic, J Mazeas, Hubert Poignant, Suzanne Salaun, Bart Lannoo, and Didier Colle. Evaluation of power rating of core network equipment in practical deployments. In *IEEE Online conference on green communications (GreenCom)*, pages 126–132, 2012.

[VLM+09a] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulié, Christophe Diot, and Pablo Rodriguez. Greening the internet with nano data centers. In *Proceedings of the*

5th international conference on Emerging networking experiments and technologies, pages 37–48. ACM, 2009.

[VLM+09b] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulié, Christophe Diot, and Pablo Rodriguez. Greening the internet with nano data centers. In *ACM CoNEXT '09*, pages 37–48, Rome, Italy, 2009.

[Vor03] André Voros. Zeta functions for the riemann zeros. In *Annales de l'institut Fourier*, volume 53, pages 665–700, 2003.

[Wil02] Carey Williamson. On filter effects in web caching hierarchies. *ACM Transactions on Internet Technology (TOIT)*, 2(1):47–77, 2002.

CHAPTER 3

Energy Efficient Routing

This chapter, like the previous one, is also devoted to energy saving in a network augmented with caches. However, this time the network structure is already deployed and the saving is achieved by putting devices into sleep mode. To maximize this, we utilize traffic aggregation and cooperation with Content Distribution Networks. Before the study itself, we present a preliminary introduction to *Linear programming*. Then, we describe how to use it to design heuristic algorithms, in a technique called *Rounding*.

3.1 Preliminary: Linear programming

For some problems, optimal solutions can be obtained with Integer Linear Programming (ILP) models. It is a general framework that can be used to model many combinatorial problems.

A *Linear Program* (LP) comprises a linear *objective* function, a set of linear inequality *constraints* and a set of *variables*, upon which the objective and constraints are defined. The objective function can be either minimized or maximized. It is also possible for a program to have no objective, where its goal is determining whether the set of constraints is feasible, i.e. if any assignment of the variables satisfies all constraints. The constraints are inequalities stating that a linear combination of variables must be not greater than a given constant. If all the variables are

real numbers, we simply call the program linear. A LP can be written as:

$$\max\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}, \tag{3.1}$$

where A is a matrix and $\mathbf{c}$ and $\mathbf{b}$ are vectors of known coefficients and $\mathbf{x}$ is the vector of variables. However, if some variables are integers, we say we face a *Mixed Integer Linear Program* (MILP) (ILP if all the variables are integral). A MILP extends LP:

$$\max\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \forall_{x_j \in I} x_j \in \mathbb{Z}\}, \tag{3.2}$$

where I is its subset of variables meant to be integer.

An interesting property of linear programs is their duality. For any LP of the form presented in equation 3.1, called the *primal* problem, its *dual* problem is:

$$\min\{\mathbf{b}^T\mathbf{y} : A^T\mathbf{y} \geq \mathbf{c}, \mathbf{y} \geq \mathbf{0}\} \tag{3.3}$$

Notice that the dual of the dual problem is the original primal problem. The objective function of the dual problem, at any feasible solution, is always greater than the value of the objective function of the primal, at any feasible solution. Furthermore, if the primal has an optimal solution $\mathbf{x}^\star$, then its dual has an optimal solution $\mathbf{y}^\star$ given by:

$$\mathbf{c}^T\mathbf{x}^\star = \mathbf{b}^T\mathbf{y}^\star. \tag{3.4}$$

These properties are often used to find bounds on the objective function value. This can be useful for solving algorithms, or as a stopping criterion when a solution that is close enough to optimum is sufficient.

Solving MILP is NP-hard. It is trivial to express SAT using ILP: simply limit the variables to be 0 or 1 and transform each clause into a constraint saying that the sum of positive variables minus sum of negative variables must be not less than one (e.g. $(a \vee b \vee \neg c)$ becomes $a + b + (1 - c) \geq 1$). Indeed, binary programming is among the original 21 NP-complete problems put by Karp in [Kar72]. Still, due to wide application over practical problems, there is a big interest in solving these models. Many exact methods have been proposed: cutting plane, branch and bound, column generation and row generation to name a few. See [Sch98] for further reading. These methods are usually accessed through solvers – software packages which allow finding exact or approximate solutions

of specified MILP. A brief overview of currently available solvers can be found in [LL10].

The fundamental problem behind routing, multicommodity flow, is classically approached in this way, see [Min06] for a survey. It constitutes a broad body of special cases. Here, we look into a simple integral problem. We are given a graph $G = (V, E)$ and a set of single commodity flow requirements Φ. Each flow requirement $\phi \in \Phi$ has given endpoints and value, $\phi = (s_\phi, t_\phi, \varphi_\phi), s_\phi, t_\phi \in V$. First, there is a set of constraints called *flow conservation*, that basically reads that what flows in must flow out, unless we're in an endpoint:

$$\forall_{\phi\in\Phi}\forall_{v\in V} \sum_{i\in V} f^\phi_{i,v} - \sum_{j\in V} f^\phi_{v,j} = \begin{cases} -\varphi_\phi \text{ if } v = s \\ \varphi_\phi \text{ if } v = t \\ 0 \text{ otherwise} \end{cases} \tag{3.5}$$

Then, for each link, the sum of values of flows flowing through it cannot exceed *link capacity* c:

$$\forall_{\{u,v\}\in E} \sum_{\phi\in\Phi} (f^\phi_{u,v} + f^\phi_{v,u}) \leq c \tag{3.6}$$

Finally, if our flow requirements come in indivisible units, we set the variable units:

$$\forall_{u,v\in V,\phi\in\Phi} f^\phi_{u,v} \in \mathbb{N} \tag{3.7}$$

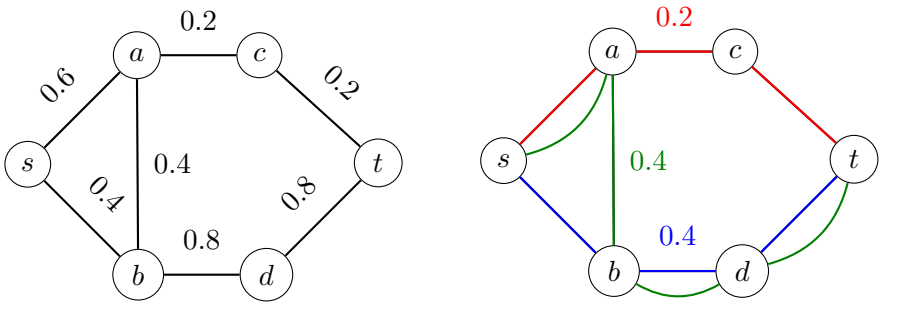
The program given without an objective tests whether the flow is feasible. If so, for each flow requirement ϕ we obtain $f^\phi_{u,v}$ – a matrix determining the flow itself.

Basing on the above we can obtain a number of useful variants. The capacity can be a constant (maybe given for each link), when doing routing over a given network, or some cost function, when doing network provisioning. Depending on the exact problem, there may be various optimization goals basing on the costs, some possible rewards, or even no goal when the only interest is finding a feasible routing. Later in this chapter, we extended this approach by taking into account Content Distribution Networks and in-network caches. Solving the ILP directly yields an exact solution, albeit the running time is exponential in the instance size. Limiting the time given to the solver may yield sub-optimal, but possibly acceptable solutions.

The usage of Integer Linear Programming is not limited to routing. A number of various graph problems are solved with it in [Coh11]. In Appendix A, we look into a graph coloring problem motivated by frequency assignment in satellite networks. It can be solved by an ILP, where for each vertex we limit interferences for a given color (frequency) up to a given threshold and minimize the total number of colors used.

3.2 Preliminary: rounding

The aforementioned Integer Linear Programs can be used as a basis for heuristics. This approach is generally called *rounding*. For the multicommodity flow problem *randomized rounding* has been introduced in [RT87]. The flows are of unit value and the optimization goal is minimizing the uniform link capacity C. Figure 3.1 displays two steps of a solution to a simple example. First, a fractional relaxation (a version of the program with integer variables are changed to fractional) of the ILP is computed. This can be achieved, using Karmarkar's algorithm proposed in [Kar84], in time $O(n^{3.5})$, where n is the number of variables in the ILP. Solving a relaxation leads to some flows being split among a number of edges, as shown on Figure 3.1a. The second step of the algorithm is decomposing such flows into a set of paths, see Figure 3.1b. Finally, we choose which path should the flow follow in an integer solution. The choice is random, weighted by the flow value of each of the paths. This simple, computable in polynomial time, procedure is guar-



(a) Fractional *st* flow

(b) The flow decomposed into trhee paths

Figure 3.1: An example path decomposition of a fractional *st* flow

anteed to give a solution with $C \leq C_o + \sqrt{3 \cdot C_o \cdot \ln \frac{|E|}{\varepsilon}}$ with probability at least $1 - \varepsilon$, where C_o is the optimal value, E is the edge set and ε a small positive real number.

A rounding heuristic is proposed for the network management problem dealt with in this chapter. In our specific problem, the routing is fractional and link capacities are given. Furthermore, for some flows, one endpoint can be chosen from among a subset of nodes, modeling demands towards Content Distribution Networks. The problem is NP-complete because of binary choice of turning devices on or off. Our rounding approach is greedy, instead of randomized. Having a solved relaxation, we turn on the most loaded devices and then iterate by solving a fractional relaxation with these additional constraints. As there is a polynomial number of relaxations to be solved, and a relaxation is solved in polynomial time, the whole algorithm has a polynomial time complexity.

While we formulate an ILP for the improper colouring problem in Appendix A, the heuristic is unrelated to it. Instead, it uses *potential interference*. Whenever a node is being coloured, it is computed what would be the interference for each possible colour and the colour with smallest value is chosen. In contrast to the previously described algorithm, this has randomness: the node to be coloured next is chosen randomly (from among the ones with highest interference) as well as the colour (from among the ones with lowest interference).

3.3 Publication

The remainder of this chapter corresponds to *Energy Efficient Content Distribution* by J. Araujo, F. Giroire, Y. Liu, R. Modrzejewski and J. Moulierac which has been submitted to the journal of Computer Communications, which is an extended version of the work of same title and authors accepted for publication in the proceedings of IEEE International Conference on Communications 2013.

3.4 Introduction

Energy efficiency of networking systems is a growing concern, due to both increasing energy costs and worries about CO_2 emissions. In [Web08] it

is reported that Information and Communication Technology sector is responsible for up to 10% of global energy consumption. 51% of that is attributed to telecommunication infrastructure and data centers. Thus, saving power is important. Backbone network operators study the deployment of energy-efficient routing solutions. The general principle is to aggregate traffic in order to be able to turn off a maximum number of devices [ZYLZ10, CMN11, BCMR12, GMM12].

On the other hand, in order to reduce network load and improve quality of service, content providers and network operators try to disaggregate traffic by replicating their data in several points of the networks, reducing the distance between this data and their users. Recent years have seen, along the growing popularity of video over Internet, a huge raise of traffic served by Content Delivery Networks (CDNs). These kinds of networks operate by replicating the content among its servers and serving it to the end users from the nearest one. CDNs deliver nowadays a large part of the total Internet traffic: estimation ranges from 15% to 30% of all Web traffic worldwide for the single most popular CDN [Aka]. Chiaraviglio et al. [CM10, CM11] have shown how the choice of CDN servers impacts the backbone energy consumption. More precisely, they aim at turning off network devices by choosing, for each demand from a client to a content provider, the best server of this CDN while being energy aware.

Here, we go further on this idea by also considering the usage of caches on each of backbone routers, while still taking into account the choice of CDN servers. It is important to mention that there have been several proposals for developing global caching systems [RK09], in particular recently using in-network storage and content-oriented routing to improve the efficiency of content distribution by future Internet architectures [PYRK08, Dan09, JST+09]. Among these studies, we mention that in this paper we do not assume any specific technology for future Internet architectures, nor anything else that would require major overhaul of how the Internet works. Thus, there is no content routing among our caches. We assume that a cache serves a single city, taking all of its contents from the original provider. We consider that caches can be turned on or off. Thus, there is a trade-off between the energy savings they allow, by reducing network load, and their own consumption.

We propose an Integer Linear Programming (ILP) formulation to re-

duce energy consumption by using caches and properly choosing content provider servers, for each demand. We implemented this formulation on the ILP solver CPLEX [CPL] version 12 and made experiments on real, taken from SNDlib [SND], and random, Erdős-Rényi [ER60], network topologies. We study the impact of different parameters: size of caches, demand intensity, network size, etc. In particular, we found that almost maximal energy gain can be achieved, in our scenarios, by caches of the order of 1 TB. Larger caches do not lead to significantly better gains. We discuss the increase of cache usage with network size. Experimental results show potential energy savings of around 20% by putting devices to sleep outside peak hours; introducing CDN to the network without caches gives 16% savings; introducing caches to network without CDN also gives around 16% savings. Furthermore, we observed that the impact of caches is more prominent in bigger networks. To be able to quantify this effect, we propose an efficient heuristic. This heuristic, called Spanning Tree Heuristic, allows us to obtain acceptable solutions in a time orders of magnitude shorter than solving the model directly with CPLEX. Furthermore, the heuristic accepts a parameter controlling a speed/quality trade-off. This trade-off is also studied in this paper.

The main take away of our work is thus that, by storing the most popular content in caches at each router and by choosing the best content provider server, we may save around 20% of power in the backbone.

The paper is organized as follows. We discuss the related work in Section 3.5. We present the problem and its formulation in Section 3.6. Section 3.7 describes how we built the instances used in the experimentations.Finally, we present the experiments we did and discuss the results in Section A.5.

3.5 Related Work

Reducing energy consumption of the backbone network has been approached before multiple times. A model where it was achieved by shutting down individual links is studied in [GMM12]. An interesting way of performing this in a distributed manner is shown in [BCMR12]. Energy efficient CDNs have also been studied recently. Authors in [MSS12] propose to reduce energy consumption in CDN networks by turning off CDN servers through considering user SLAs. In order to optimize the power

consumption of content servers in large-scale content distribution platforms across multiple ISP domains, a strategy is proposed in [GWS12] to put servers into sleep without impact on content service capability. Our work is different from these works, since they do not consider in-network caches.

Network caches have been used in global caching systems [RK09]. In recent years, several Information Centric Networking architectures, such as Cache and Forward Network (CNF) [PYRK08], Content Centric Networking (CCN) [JST+09] and NetInf [Dan09], have exploited in-network caching. Their objectives are to explore new network architectures and protocols to support future content-oriented services. Caching schemes have been investigated in these new Internet architectures [PYRK08, LSG12, PCP12]. Similar to our work, these works also use in-network caches, however they do not consider energy savings.

Energy efficiency in content-oriented architectures with an in-network caching had been studied recently in [GAKG11, SLW11, CGKA12]. In [GAKG11], authors analyze the energy benefit of using CCN by comparison to CDN networks. A further work [CGKA12] considered the impact of different memory technologies on energy consumption. Adding network caches that work transparently with current Internet architecture has been studied, with linear power models, in [MLG+11], where caches are added to backbone routers and in [JNWC11], where it is found that optimal placement during peak hours is in the access network. These works focus on the energy efficiency considering data delivery and storage, however, they do not take into account the energy savings by turning on/off network links. Authors in [SLW11] extend GreenTE [ZYLZ10] to achieve a power-aware traffic engineering in CCN network. It is different from our work, since we consider energy consumption of in-network caches that could be turned on or off, as well as a cooperation between network operators and content providers.

Most closely related to ours is the work from Chiaraviglio et al. [CM10, CM11], which enables the cooperation between network operators and content providers, to optimize the total energy consumption by an ILP formulation for both sides. In this paper, we also consider this cooperation to achieve such a total energy saving. Our work is an extension of this optimization problem formulation, through considering in-network caching.

3.6 Problem Modeling

What follows in this section is a discussion of model parameters, formal problem definition and a Mixed Integer Linear formulation used to solve it.

However, let us first informally recall the problem description and some assumptions we consider. Our goal is to save energy on a backbone network by aggregating traffic and turning off as many devices as possible. We consider that this network has a set of demands between pairs of routers and a set of demands to CDN servers in an overlay of this network. A demand to a CDN can be satisfied by any of its servers, which are placed in different routers of our backbone network. Thus, these demands have of course a single destination, but several possibilities of sources, one for each CDN server. Moreover, we consider that each backbone router of our network has a cache, with a limited amount of storage, that can only be used to satisfy demands to its router. Our goal is to satisfy all these demands, under the capacity contraints of CDN servers, caches and links, while minimizing the number of links and caches that are turned on.

Parameters

For in-network caches, it is still an open question: if and how they should be deployed. Therefore, we avoid making specific assumptions about the details. Once the question is answered, the model we propose can be updated to answer any possible specific concerns. However, the conclusions can change, if the actual parameters vary heavily from our estimates.

Cache hit rate A cache, located in each router, automatically caches the most popular content, potentially saving a fraction of any demand. Establishing this fraction is a non-trivial task. According to [HH10], content popularity follows a Zipf-like distribution. In their study, they computed the relation between cache size and hit rate for a trace of traffic towards YouTube. Note that this relation does not depend on the number of cache accesses, only on the relative size of the cache and all the content collection. This relation is shown on Figure 3.2, with the assumption that an average video is 100 megabytes. The figure shows results for two algorithms: *least recently used*, a classic caching algorithm,

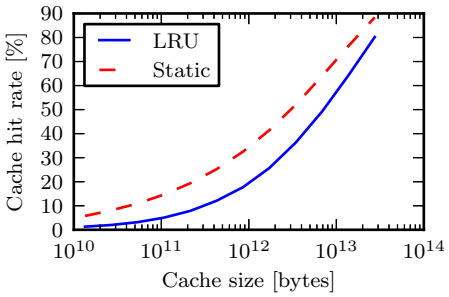


Figure 3.2: Cache hit ratio for YouTube trace, assuming average video size 100MB, following the results in [HH10].

and *static most popular*, a simple algorithm proposed by the authors. For example with a cache of around 800GB the expected hit rate is around 17.7% using LRU and around 32.5% using the static algorithm, thus saving an equivalent fraction of traffic.

As the situation changes frequently, both regarding to the volume of popular content and available storage, we leave this fraction as a parameter of the model: α – the maximal part of any demand that can be served from a cache. Network operator can establish it empirically, by means of measurements. Typically, we take $\alpha \in [0.2, 0.35]$.

Cache power usage In our model we deal with two types of equipment: links and caches. In practice, main energy drain of links are port cards and amplifiers. As can be seen in Powerlib [VHI12], power requirements of single port cards suitable for long haul networks are well over 100 Watts, while other backbone cards can be as few as a quarter of that. For the caches, the main concern is fast mass storage. This has improved recently, with current SSD models offering 1TB of storage accessed at 10Gbps consuming below 10 Watts of power, for example [OCZ].

Again, as the practical values in the time of implementation are hard to predict, we make this ratio another parameter of the model: β – the power consumption of a cache divided by the power consumption of a link. Typically we take $\beta \in [0.1, 1]$.

Problem definition

We use a typical model, from the perspective of a backbone provider, where aggregated traffic between cities is expressed as a demand matrix. We augment this matrix to represent not only cities, but also content providers. This is motivated by the fact that content providers generate traffic that can easily be equal to that of a city.

Let us first formally define the optimization problem we are dealing with. We call it Energy Efficient Content Distribution. In this problem, we model the network by a graph $G = (V, E)$, for which we have a link capacity function $c : E \to \mathbb{R}_+$ and city to city demands $\tilde{\mathcal{D}}_s^t, \forall s, t \in V$. Moreover, we are given a set of content providers P. For each content provider $p \in P$, the subset of vertices of $V(G)$ containing its servers is given by the function $\mathfrak{L}_p \subseteq V(G)$. Each server placed in location $l \in \mathfrak{L}_p$ of a content provider p has a capacity $C(\mathfrak{s}_p^l)$. We are also given demands from cities to content providers given by the function $\tilde{\mathfrak{D}}_s^p$, for every $s \in V, p \in P$. We consider that the data is replicated at each node of $\mathfrak{L}_p$. Finally, each node $v \in V(G)$ in the network has a cache of bandwidth capacity $b(v)$.

The goal of our problem is to find a feasible routing in G satisfying all the demands $\tilde{\mathcal{D}}_s^t$ and $\tilde{\mathfrak{D}}_s^p$ under the capacity constraints $c(u, v)$, $C(\mathfrak{s}_p^l)$ and $b(v)$ that minimizes the total energy consumption of the network. By total energy consumption, we mean the energy used by the links plus the energy used by the caches. For each cache, despite of a fixed energy cost of turning it on, we also consider an increased usage of energy in terms of its load.

Mixed Integer Linear Programming Formulation

First recall that our goal is to turn off links and caches in order to minimize the amount of energy used in the underlying network. Consequently, we use a variable x_{uv} to indicate if the link uv is turned on or off, for every $\{u, v\} \in E$. The model is normalized as to say that every link uses 1 unit of energy. We denote this unit l_c. We use a variable y_v to indicate if the cache at router v is turned on or off, for every $v \in V$. We say that a cache uses at most β units of energy. Finally, we recognize that mass memory access can constitute a significant energy cost. Thus, we use a variable z_v to indicate the load (fraction of used bandwidth) of

the cache in router v. We assume that an idle cache uses fraction γ of β and its power consumption grows linearly with load to reach β once fully utilized. The objective function is then written formally as:

$$\min \sum_{\{u,v\}\in E} x_{uv} + \sum_{v\in V} \beta\gamma y_v + \sum_{v\in V} \beta(1-\gamma) z_v.$$

Denote $\tilde{\mathcal{D}}$ and $\tilde{\mathfrak{D}}$ as the demands posed in the problem instance, respectively from other cities and content providers. A cache in a source router s, when turned on, allows to save a portion of any demand up to α, call these savings respectively $\mathcal{C}$ and $\mathfrak{C}$. We will consider *reduced demands*, denoted $\mathcal{D}$ and $\mathfrak{D}$, which are the *input demands* with the caching savings subtracted:

$$\begin{aligned}
\mathcal{D}_s^t &= \tilde{\mathcal{D}}_s^t - \mathcal{C}_s^t \quad &,\forall s,t \in V,\\
\mathcal{C}_s^t &\le \alpha\tilde{\mathcal{D}}_s^t \quad &,\forall s,t \in V,\\
\mathfrak{D}_s^p &= \tilde{\mathfrak{D}}_s^p - \mathfrak{C}_s^p \quad &,\forall s \in V, p \in P,\\
\mathfrak{C}_s^p &\le \alpha\tilde{\mathfrak{D}}_s^p \quad &,\forall s \in V, p \in P.
\end{aligned}$$

Then, we record the load of the cache:

$$\sum_t \mathcal{C}_s^t + \sum_p \mathfrak{C}_s^p = z_s b(s) \quad ,\forall s,t \in V, \forall p \in P.$$

Finally, the load cannot exceed the capacity and needs to be zero if cache is off:

$$z_s \le y_s \quad ,\forall s \in V.$$

Each possible source $s \in V$ demands from each provider $p \in P$ an amount of data flow $\mathfrak{D}_s^p \ge 0$. The provider has a set of servers of $\mathfrak{s}_p^l$ located in a subset of nodes of the network $l \in \mathfrak{L}_p \subseteq V$. Each of those servers sends $\mathfrak{S}_p^{l,s}$ flow units, to collectively satisfy the demand:

$$\sum_{l\in\mathfrak{L}_p} \mathfrak{S}_p^{l,s} = \mathfrak{D}_s^p \quad ,\forall s \in V, p \in P.$$

Each server $\mathfrak{s}_p^l$ has a constrained capacity $C(\mathfrak{s}_p^l)$, which limits the demands it can satisfy:

$$\sum_{s\in V} \mathfrak{S}_p^{l,s} \le C(\mathfrak{s}_p^l) \quad ,\forall p \in P, l \in \mathfrak{L}_p.$$

	Popularity	Server capacity	Server locations
CDN1	40	0.3	Berlin Hamburg Duesseldorf Frankfurt Muenchen Nuernberg
CDN2	20	0.45	Berlin Duesseldorf Frankfurt Muenchen
CDN3	15	0.6	Berlin Frankfurt
CDN4	15	0.5	Hamburg Frankfurt Muenchen
CDN5	10	0.2	Berlin Duesseldorf Frankfurt Hamburg Muenchen Nuernberg Osnabrueck

Table 3.1: Content Distribution Networks assumed for the *germany50* network.

Now we set flow constraints. By $f^s_{u,v}$ we denote the flow on edge $\{u, v\}$ corresponding to demands originating from s.

$$\sum_{v \in N_u} f^s_{v,u} - \sum_{z \in N_u} f^s_{u,z} =$$

$$= \begin{cases} -\sum_{p \in P} \mathfrak{D}^p_s - \sum_{t \in V} \mathcal{D}^t_s & u = s \\ \mathcal{D}^u_s + \sum_{\{p \in P | u \in \mathfrak{L}_p\}} \mathfrak{S}^{u,s}_p & \text{otherwise} \end{cases} \quad , \forall s, u \in V.$$

Finally we consider capacities of links, denoted $c(uv)$. The constraints involve both kinds of flows and the on/off status of the links:

$$\sum_{s \in V} (f^s_{u,v} + f^s_{v,u}) + \leq c(uv) x_{uv} \quad , \forall \{u, v\} \in L.$$

All variables are non-negative real numbers, except for x_{uv} and y_v which admit only values in $\{0, 1\}$.

Spanning tree heuristic

Since CPLEX was not able to solve the ILP model described in the last section for bigger instances, we describe here a polynomial-time heuristic to our problem. For instance, for a random example with 150 cities and 300 links, CPLEX was not able to produce any feasible solution within two hours, while the proposed algorithm can give a good solution in two minutes.

Our heuristic is an iterative algorithm that, at each step $i \geq 0$, computes an optimal (fractional) solution s_i for the relaxation of our model and fix value of some variables of the model corresponding to the usage of links and caches (i.e. the integral variables x_{uv} and y_v). When we say that we *fix* a variable x to a value $c \in \{0, 1\}$ at step i, we mean that we add a constraint $x = c$ to the model used to compute s_j, for all $j > i$.

At the first step 0, our heuristic computes a solution s_0 of relaxation of the model. Then, by setting the weight of each edge to be the value of its corresponding variable in s_0, a maximum spanning tree T of the input network graph G is computed and all the variables x_{uv} of all the edges $uv \in E(T)$ are fixed to one.

After this initialization step, the heuristic solves, at each step $i > 0$, the relaxation of the model (which will already have several variables with fixed values) to get an optimal solution s_i. Then, if some other variables x_{uv} or y_v have value $v \in \{0, 1\}$ in the solution s_i, these variables are fixed to this value v. Finally, at least one most loaded device is fixed to be turned on. To speed up the process, the heuristic has a parameter $\mathcal{S}$. At each step i, we also fix $\mathcal{S}$ fraction of the highest value variables x_{uv} or y_v whose values v are in $0 < v < 1$ to one. Once all the integer variables are fixed, the relaxation is solved one last time. This gives us a valid solution to the Integer Linear Program.

The heuristic has been implemented in Java and it can be downloaded as open source[1]. Note that we use CPLEX to solve the relaxations of the model at each step of the heuristic. The performance of this heuristic is analyzed in Section A.5.

On the complexity of the heuristic algorithm The model we propose has a polynomial number of constraints on the size of the input. It is well-known that its relaxation can then be solved in polynomial time. The number of devices whose variables have to be set to 0 or 1 by our heuristic is n caches (one at each node) plus m edges. The first iteration puts $n - 1$ edge variables to 1. When a variable is set to 0 or 1, it is not reconsidered during the algorithm execution. Hence, the number of relaxations solved, i.e. of steps of the heuristic, is bounded by $m + 1$.

Note that we indicated the number of iterations and the execution times in seconds for varied values of $\mathcal{S}$ in Section 3.8 and for networks of

[1] `https://github.com/lrem/GreenContentDistribution`

varying size in Section 3.8.

3.7 Instance generation

The Survivable fixed telecommunication Network Design (SND) Library contains a set of real network topologies, which we use as a base for most of our instances. In particular, we have decided to use three networks with considerably different sizes:

- *Atlanta* – $|V| = 15, |E| = 22$, unidentifiable cities
- *Nobel-EU* – $|V| = 28, |E| = 41$, major European cities
- *Germany50* – $|V| = 50, |E| = 88$, major German cities

We added the position of the Content Distribution Network servers. Usually, Content Distribution Networks locate their servers in Internet Exchanges and major Points of Presence, to minimize the network distance to the end users. Locations of such points are publicly known. Thus, for topologies with clearly identifiable cities, we have ready sets of candidate locations for CDN servers. Otherwise (Atlanta network), we put them manually at cities which minimize the route lengths.

We used a population model to build the traffic matrices of the demands between cities. Then, we augmented matrices with the demands towards content providers. Obtaining exact figures about CDN market shares and operational details is out of scope of this study. Still, we explored the publicly available information, e.g. [Aka], to come up with a list of the major providers. Each of the networks is assigned a *popularity*, which is based on market share either claimed by the company or media. The number of servers is heterogeneous and we try to arrange it into distinct classes in regard to popularity/server capacity proportion, i.e. there can be networks with many small servers, or few strong ones.

Table 3.1 exemplifies CDN specification used in the *germany50* network. Server capacity means what part of total demand towards a network can be satisfied by the infrastructure in a single location. For example, just two servers with capacity 0.5 can satisfy all demands towards CDN4.

A more detailed description of the instance generation can be found in the research report [AGL+12].

3.8 Results

In this section, we first validate our heuristic. We show that it is able to find good solutions for small and medium-sized networks by comparing with optimal solutions given by the model. We implemented the formulation on the ILP solver CPLEX version 12. We then show that the heuristic is fast and is able to quickly find solutions for large networks for which CPLEX was not able to find any feasible solutuion.

Then, we investigate the potential energy savings of our solution on realistic networks. We exhibit the impact of the cache, CDN and network parameters, such as cache size, number of CDN servers, or route lengths. Note that, as described in Section A.5, energy consumption is given in normalized energy units equal to energy used by one link, denoted l_c.

When directly solving the ILP, by default we set a limit on the execution time to five minutes per instance.

Validation of the Heuristic Algorithm

In order to validate the Spanning Tree Heuristic, we compare its performance to solving the integer model directly with CPLEX. First we show the differences in several examples. Then, we focus on showing the impact of the parameter $\mathcal{S}$, which governs the speed/quality trade-off, on three chosen examples.

Comparison of the heuristic and the ILP

Table 3.2 displays performance comparison between Spanning Tree Heuristic and solving the ILP directly by CPLEX version 12. It compares the values of objective function and wall clock time taken by the computation on an Intel i7-powered computer. The Δ columns mean, respectively, by what percentage the solution found by the heuristic is worse and how much time is saved by using it. The heuristic parameter $\mathcal{S}$ is set to 0.2. It is discussed in the next section.

First, notice that for very small networks it is feasible to solve the ILP optimally. This is exemplified by the 15-node Atlanta network. The optimal solution is found within two seconds. Interestingly, the running time grows for lower traffic. This is entirely because rising the lower bound, which has to be equal to the objective value to state the solution

Topology	$\|V\|$	Total energy[l_c]		Δ
		Model	Heuristic	
Atlanta (high traffic)	15	18.8$\star$	19.0	1%
Atlanta (medium traffic)	15	16.6$\star$	18.6	12%
Atlanta (low traffic)	15	14.1$\star$	14.4	2%
Nobel-EU (high traffic)	28	31.4$\star$	35.1	12%
Nobel-EU (medium traffic)	28	28.4	32.2	13%
Nobel-EU (low traffic)	28	27.9	30.2	8%
Germany50 (high traffic)	50	69.7	69.0	-1%
Germany50 (medium traffic)	50	54.2	61.6	14%
Germany50 (low traffic)	50	50.0	56.2	12%
Random	150	No solution	203.7	—

Topology	$\|V\|$	Computation time[s]		Δ
		Model	Heuristic	
Atlanta (high traffic)	15	1.5	0.6	60%
Atlanta (medium traffic)	15	5.2	0.6	88%
Atlanta (low traffic)	15	34.4	0.6	98%
Nobel-EU (high traffic)	28	1075	1.8	99.8%
Nobel-EU (medium traffic)	28	1800	1.3	99.9%
Nobel-EU (low traffic)	28	1800	1.1	99.9%
Germany50 (high traffic)	50	300	8.5	97%
Germany50 (medium traffic)	50	300	5.0	98%
Germany50 (low traffic)	50	300	2.9	99%
Random	150	7200	127.8	—

Table 3.2: Comparison of results given by the Spanning Tree Heuristic (labelled *Heuristic*) and by solving the model directly with CPLEX (labelled *Model*). The $\star$ symbol denotes optimal solutions.

is optimal, becomes much harder. Solutions given by the heuristic are close to the optimum, while the time needed to find them is much shorter. Still, for networks of this size, we would strongly recommend solving the model directly.

For networks up to 30 nodes it is still feasible to find optimal solutions. However, the cost of obtaining the solution is rather high, while closing the gap to the lower bound becomes impractical for low traffic. Thus, we limited the CPLEX execution time to half an hour. On the other hand, Spanning Tree Heuristic provides its solutions in under two seconds. Again, by choosing the heuristic, we accept only a slight increase in consumed energy. Precisely, to obtain a solution within 12% of the optimum, we save 99.8% of the computation time.

In medium-sized networks, such as Germany50, finding exact solution becomes impractical. Thus, we set a limit of 5 minutes to obtain near-optimal results. This allows Spanning Tree Heuristic to obtain a slightly better solution than the ILP, while taking only 3% of the running time, in the case of high traffic. In the other cases it is still not far quality-wise, while taking negligible time.

Finally, we take a big random instance. The topology is a two-connected Erdös-Renyi graph, with 150 nodes, an average degree of four and one CDN with fifteen servers. Each city issues demands only to seven other cities. The overall traffic level is medium (demand ratio 4.0), as these kind of instances are prone to bottlenecks, which could render higher traffic levels unrouteable. It is infeasible to directly obtain any integer solution of the model. After two hours CPLEX was not able to propose even a trivial solution (e.g. turning on all the devices). Spanning Tree Heuristic, in just above two minutes, gives a solution that is 35.8% over the trivial lower bound of a minimal connected network.

To conclude, we say that the Spanning Tree Heuristic is clearly the better choice for big networks. For small to medium-sized ones, its results are always reasonably good, while its running time is very short. Therefore, it is a viable choice whenever the computation time is an issue.

Speed/quality trade-off of the Spanning Tree Heuristic

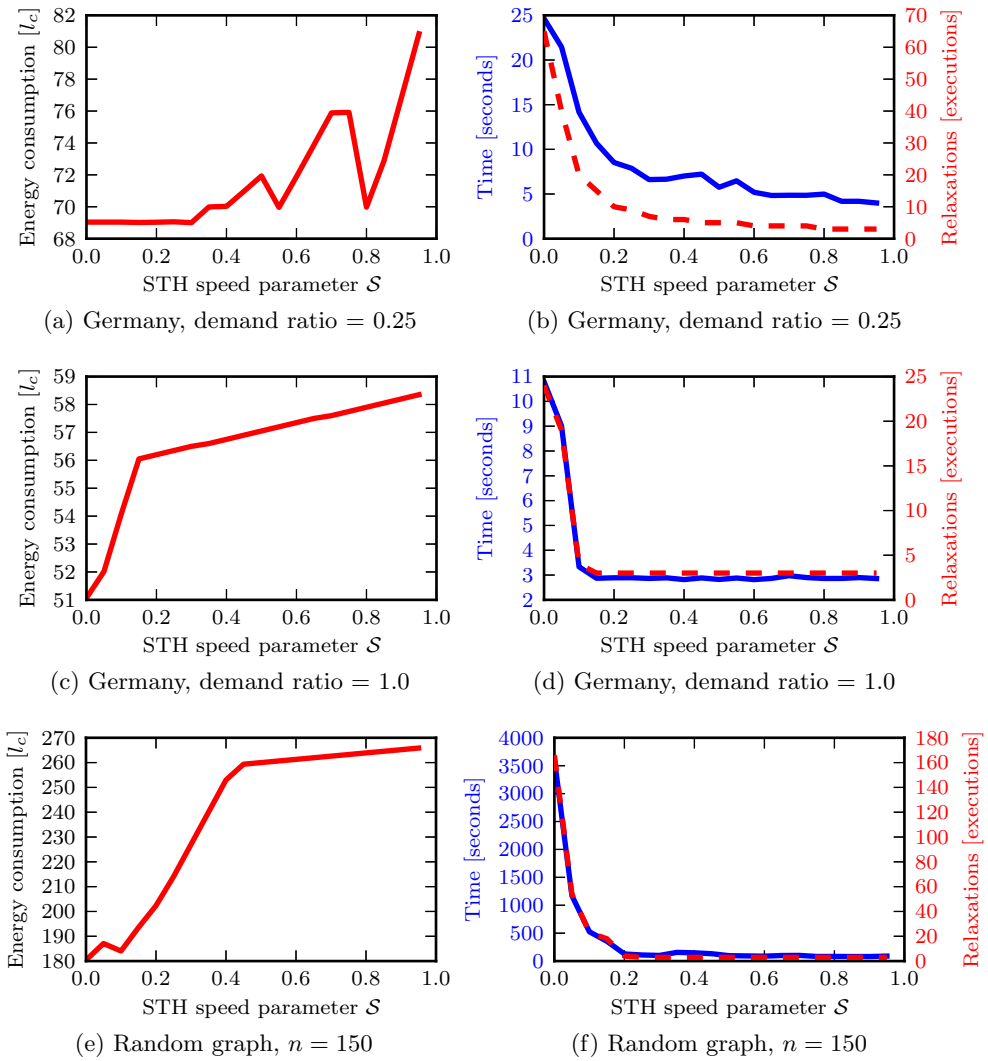
As stated in Section 3.6, the parameter $\mathcal{S}$ governs an execution speed vs quality of solution trade-off for the Spanning Tree Heuristic. We investigate its influence in this section.

First, recall that $\mathcal{S}$ determines the fraction of undecided variables to be fixed to an integer value within an iteration. Each iteration at least one variable is set to one, so setting $\mathcal{S}$ to zero means turning on devices one by one. It is easy to see how increasing $\mathcal{S}$ reduces the number of iterations. To comprehend how it can decrease the quality of obtained solution, imagine a simple example, that represents a fragment of an instance. Take two cities with two disjoint paths and one demand between them. Let the value of that demand be equal to bandwidth of a link. One valid solution of the relaxation can be splitting the demand in half and routing both halves along both paths. The optimal integer solution for this case is all the flow going through one route, the links of the other turned off. If $\mathcal{S} = 0$, then after the first step one link will be turned on. The only possible solution of the relaxation will route all the traffic through the path containing this link. Thus, the solution found by Spanning Tree Heuristic will be optimal. However, if $\mathcal{S} > 0$ and two links are turned on in the first step, then it is possible the two links will be on different paths. Thus, the integer solution will have some unnecessary links turned on. In the extreme case of $\mathcal{S} = 1$ all devices will always be turned on.

Figure 3.3 shows the impact on three examples. In all cases, the x axis determines the value of parameter $\mathcal{S}$. The left column plots the value of the objective function in integer solutions. The right column shows computational costs, both in terms of wall clock time in seconds (solid blue lines) and number of relaxations solved (dashed red lines).

First, look into an instance based on maximum traffic sustainable in the *Germany50* network. Solutions obtained are displayed on plot 3.3a. Recall that the value found by a solver for this instance was 69.7 energy units. Taking between 24 and 8 seconds, Spanning Tree Heuristic with $\mathcal{S} \leq 0.3$ obtains solutions with 69.0 units. This means it is in this case both faster by an order of magnitude and gives a marginally better solution. Note that even at $\mathcal{S} = 1$ not all devices are turned on. This is because, after freezing the spanning tree, some devices get turned off before all the undecided ones are turned on. Looking at plot 3.3b, we see that the number of relaxations solved and the running time are falling drastically for $\mathcal{S} \leq 0.2$. Then, they decrease more slowly, with 6 seconds at $\mathcal{S} = 0.5$ and 4 seconds at $\mathcal{S} = 1.0$.

Second, we assign to the same network a small load, that sill does not



(a) Germany, demand ratio = 0.25

(b) Germany, demand ratio = 0.25

(c) Germany, demand ratio = 1.0

(d) Germany, demand ratio = 1.0

(e) Random graph, $n = 150$

(f) Random graph, $n = 150$

Figure 3.3: Impact of the parameter $\mathcal{S}$, left column plots the energy consumption of obtained designs, while right column plots the computational cost.

allow for routing on a spanning tree (which would be a trivial case for the heuristic). With model given directly to a solver, we have obtained in 5 minutes a solution with value 50. Plot 3.3c shows that the best solution found by Spanning Tree Heuristic is still one unit worse and can deteriorate by almost eight further units. On the other hand, the

maximum time taken by Spanning Tree Heuristic is 10.8 seconds. For $\mathcal{S} = 0.1$ it is already 3.3 seconds, reaching 2.8 at $\mathcal{S} = 1$.

Finally, we present results for the same random graph as in the previous section. Looking at plot 3.3e, we see that there is significant but steady increase in energy consumption until $\mathcal{S} = 0.4$. At that point, the value objective function is nearly saturating, at 1.44 times the value for $\mathcal{S} = 0$. On the other hand, plot 3.3f shows that there is a sharp decrease in computational cost until $\mathcal{S} = 0.2$. As the objective value at that point is not far from the best known value, we deduce that this is a reasonable value of $\mathcal{S}$ for fast solving of big instances. Note that when solving the model directly, CPLEX 12 is not able to produce any integer solution within reasonable timespan of two hours. The only lower bound on the objective value we know comes from the fact, that the network needs to be connected. Thus, there are at least 149 links needed. This means that the heuristic, with $\mathcal{S} = 0$, is at most within 20.8% from the optimal solution.

As we have seen in the above examples, the Spanning Tree Heuristic is a good alternative to solving the model directly for big networks. Furthermore, even when it is possible to obtain a solution directly from the model, it may be possible that the heuristic provides a solution of similar quality in a shorter time.

Impact of cache parameters

In this section, we exemplify the impact of parameters of the cache. We look into how the obtained network designs differ on changing values of the cache hit rate α and of the cache power usage β. Due to lack of space, results are given here for the *germany50* network. The demand ratio is set to 0.3, which represents high traffic. Similar results on different networks can be found in [AGL$^+$12].

First, we look at the effects of changing the parameter α, shown in Figure 3.4a. Recall, that it limits what part of any single demand can be served from a cache. Increasing the significance of caches results in more being used and energy being saved. However, note that once about 15% of traffic can be cached, further gains are highly diminished. This, according to Section 3.6, is equivalent to about 800GB or just 100GB depending on the cache algorithm used.

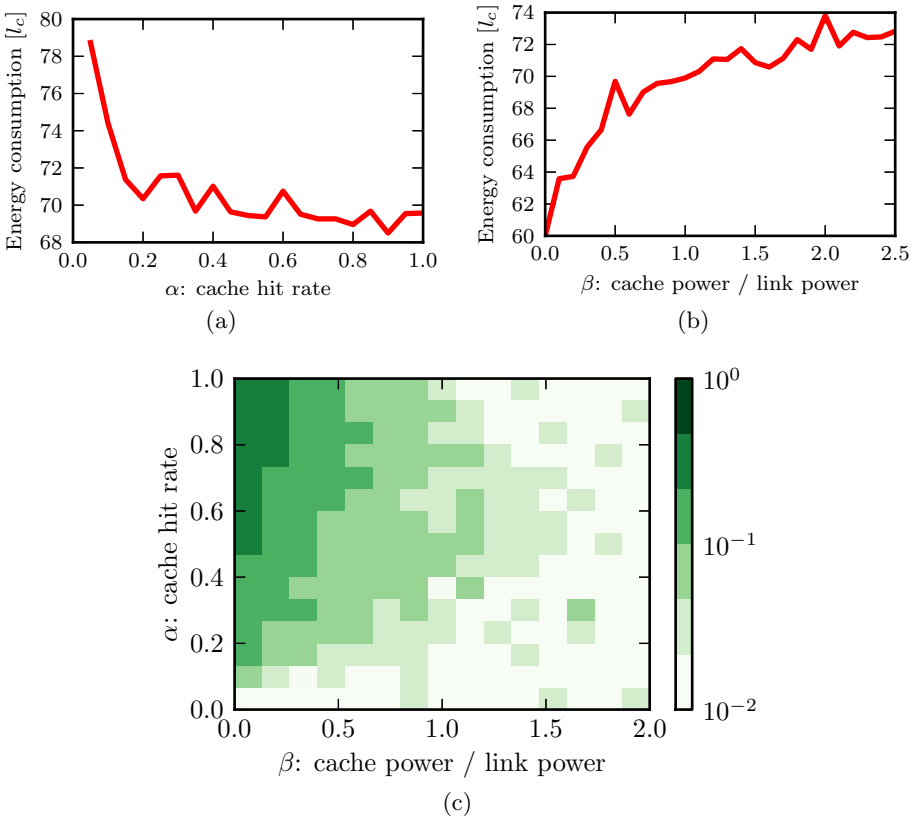
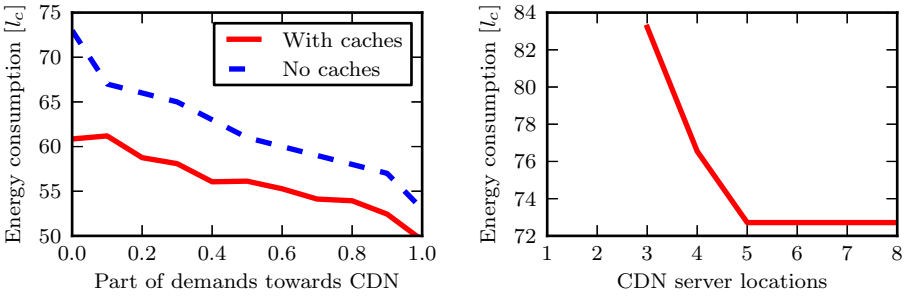


Figure 3.4: Energy consumption in network designs obtained by model with different parameters.

Figure 3.4b shows the effects of changing maximum cache power usage, β. As we can see, when the caches use no energy, the network uses 60 units of power. Then it raises, through 63.4 for $\beta = 0.1$, to 69.7 for $\beta = 0.5$. After this point, further increases to β have little effect, not increasing past 74. This is because at this point caches simply get turned off as they consume too much energy.

Figure 3.4c shows combined effects of both parameters. The demand ratio in it was increased to 0.33, to make routing without caches feasible. Then, a baseline power consumption has been established with caches disabled to be 71. For each pair of parameters, energy savings relative to that baseline are mapped to a color and displayed in appropriate region



(a) In function of CDN popularity.

(b) In function of server location cardinality.

Figure 3.5: Total energy consumption varied by CDN properties.

of the figure. The darker the color the higher the savings.

Impact of CDN parameters

Then, we investigate the impact of the cooperation with CDN, shown on Figure 3.5. Plot 3.5a shows the evolution of energy consumption in function of what part of all demands are directed towards CDN networks. The demand ratio for this plot is set to 0.33, to make routing without caches nor cooperating content providers feasible. Results both with and without caches are compared. As we can see, introducing cooperating content providers to a network without caches is highly beneficial. In the extreme case when all traffic would be served by CDNs, energy consumption would decrease by 27.4%. At today's claimed values this number is still 16.4%. Then, introducing caches to a network without CDN gives 16.7% savings. There remain 8.0% savings at today's CDN popularity. What may be a bit surprising, there are still 6.6% savings by introducing caches when 100% of traffic is served by the Content Delivery Networks. Finally, comparing network without CDN nor caches, to network with 50% of traffic served by CDN and with enabled caches, we save 23.12% of energy.

Plot 3.5b investigates how many location choices are needed to achieve good savings. In this scenario, for the sake of clarity, there is only one CDN. Its servers are potentially located in: Berlin, Frankfurt, Muenchen, Hamburg, Dortmund, Stuttgart, Leipzig and Aachen. In each data point, only the first n servers from this list are enabled. Each server is able to

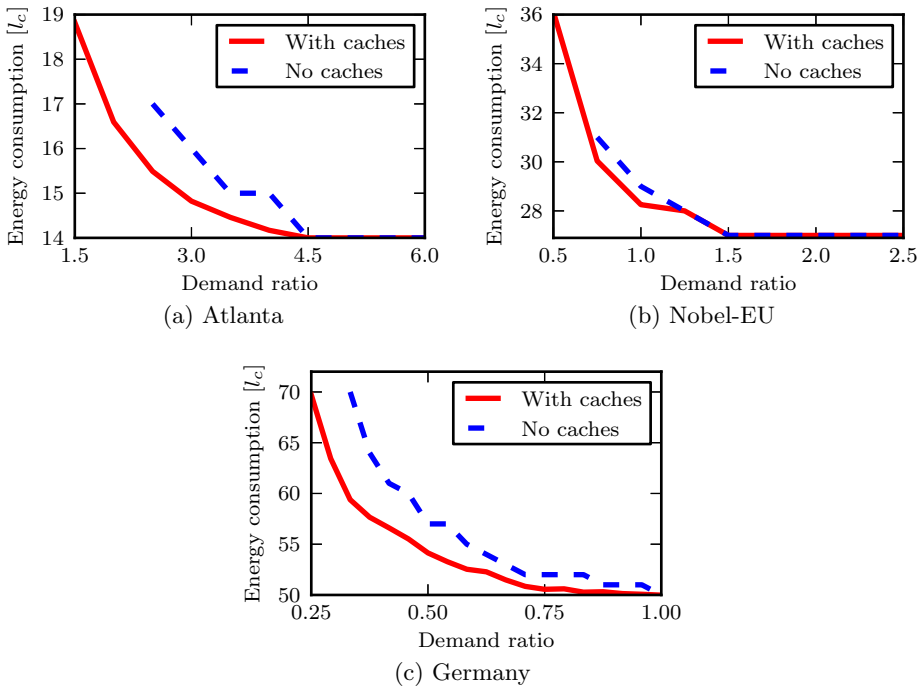


Figure 3.6: Comparison of energy consumption with and without caches in the model.

provide all the demands alone, 50% of all traffic is served by the CDN. It is infeasible to route with less than 3 locations. As we can see, increasing the number of possible choices from 3 to 5 yields around 13% of energy savings. Further increases have little effect. Thus, in this network of 50 cities, it is optimal to have 5 server locations.

Impact of traffic level

In this section, we look into the potential reduction of energy consumption of the networks in our model, both with and without usage of the caches, exploiting the variance in network traffic over time. The parameters used throughout this section are: $\alpha = 0.35$, $\beta = 0.1$ and cache bandwidth is half of a link.

Figure 3.6 shows energy consumption in function of demand ratio, that is the inverse of traffic level. As we can see, in all the networks,

Network	Nodes count	Maximum energy saved due to caches	Total energy savings (load=50%)
Atlanta	15	8.9%	21.3%
Nobel-EU	28	3.2%	21.7%
Germany	50	16.7%	22.3%

Table 3.3: Potential energy savings

enabling caches makes routing feasible under much higher loads than before. For example in the case of Germany, we can accommodate an increase in demands by one third. Then, as traffic decreases, we can save energy by turning off some devices. The right column of Table 3.3 states relative difference between energy consumption of a network under highest possible load and half of that load, with caches enabled.

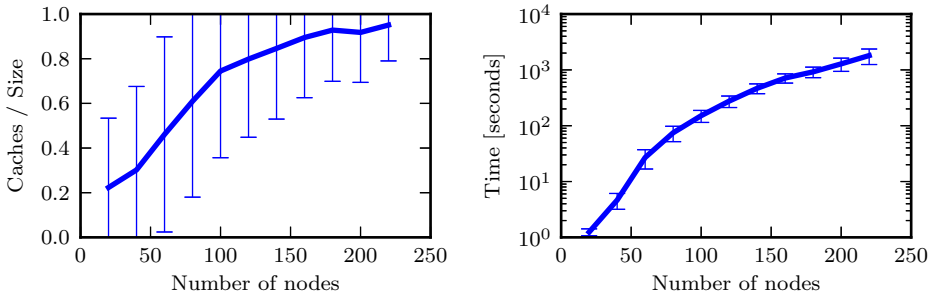
For a range of demand values, it is feasible to route without caches, but at a higher total energy cost. Note that half of maximum sustainable load is in all cases within this range. The left column of Table 3.3 shows the highest difference of power consumption accommodating the same traffic with and without caches.

As can be seen, there is a point after which there are no additional savings with falling traffic. This is when the routing is feasible on a spanning tree, using no caches. Turning off any additional device would disconnect the network.

What is interesting is the fact that caches have a much higher effect in the *germany50* than the smaller instances. We attribute that to longer routes, which mean higher energy cost to transfer the data through the network. This effect is investigated in Section 3.8.

Impact of network size

We have seen varying usage of caches in the studied networks. An explanation for that is the difference of route lengths in the diverse networks. Energy is saved by serving from a cache close to the user. Savings depend on how long would be the route traversed by the data, if it was served from content provider. A longer route yields higher reductions. However, in the biggest network we used, the *germany50*, the average route length is only 4. Furthermore, when looking at a distance traveled by an average bit of data, this length is only 2.6. We claim that in bigger



(a) Cache usage divided by network size (b) Execution time (note the log scale)

Figure 3.7: Spanning Tree Heuristic on Erdős-Rényi graphs.

networks we could see higher utility of caches.

To estimate the impact of route length, we look into results on Erdős-Rényi graphs. Recall that in these graphs, the route lengths grow logarithmically in respect to the graph size. As we need many big networks to demonstrate the effect, obtaining integer solutions directly from a solver would be impractical. Therefore, the results presented are computed using the Spanning Tree Heuristic.

Figure 3.7a shows the number of caches used divided by the number of cities in two-connected Erdős-Rényi graphs of increasing sizes. The average degree of each graph is 4, each city emits 7 demands to random other cities, cache parameters are $\alpha = 0.35$, $\beta = 0.1$ and $\gamma = 0.5$. Each data point is an average over at least two thousand instances, error bars represent standard deviation.

As we can see, with no other parameters changing, usage of caches clearly grows with increasing network sizes. In a network of size 20, having average route length around 2.3, average number of caches on is only 4.47 (22.3%), while in networks of size 220, of average route length around 4.2, there are on average 209.2 (95.1%) caches turned on. Caches see an average usage over 50% for networks of size at least 80, where the average route length is only around 3.4. This size can correspond to small networks comprised of both core and metropolitan parts, or just big core networks.

Figure 3.7b displays the computation times. The value of $\mathcal{S}$ is 0.2. The execution time grows quickly. This is not due to the number of heuristic iterations, between 20 and 220 nodes the number of relaxations

solved only doubles. However, at $n = 220$ a single relaxation takes 6 minutes on average. Thus, the time needed to find the fractional routing is the critical part of the computational cost.

3.9 Conclusions and further research

In this work, we addressed to the problem of energy saving in backbone networks. To the best of our knowledge, this is the first work to consider that impact of in-router caches, along with assigning servers of Content Delivery Networks to demands, in an energy-efficient routing.

We have proposed a new Integer Linear Programming model for saving energy in backbone networks by disabling links and caches of this network and a polynomial-time heuristic for this problem. We compared the performance of the solutions proposed by our heuristic against those found by CPLEX. In small to medium-sized instances the solutions given by the heuristic are close to that of the integer program. Being faster by orders of magnitude, it allows to find good solutions for bigger networks, where CPLEX was not able to produce any feasible solution in hours.

We studied instances based on real network topologies taken from SNDLib. The total energy savings found oscillate around 20% for realistic parameters. Part of energy saved solely due to introduction of caches is up to 16% in our instances.

As a future work, the model could be extended to enable the usage of a single cache to satisfy the demands of multiple cities, i.e. to let a cache satisfy demands to different routers and not only to its own router. The energy savings will probably grow in this model, however it would be interesting to study how this solution could be deployed.

One could also look at different network architectures. This work considered only the backbone. A next step could be introducing access networks, leading to larger instances. As the savings due to caches grow with network size, they should be substantially higher in this case. This could also motivate study of new mechanisms, e.g. layered caching.

3.10 Bibliography

[AGL+12] J. Araujo, F Giroire, Y. Liu, R. Modrzejewski, and J. Moulierac. Energy efficient content distribution. Research Report RR-8091, INRIA, 10 2012.

[Aka] Akamai. URL: http://www.akamai.com/html/riverbed/akamai_internet.html.

[BCMR12] Aruna Prem Bianzino, Luca Chiaraviglio, Marco Mellia, and Jean-Louis Rougier. Grida: Green distributed algorithm for energy-efficient ip backbone networks. *Computer Networks*, 56(14):3219–3232, August 2012.

[CGKA12] Nakjung Choi, Kyle Guan, Daniel C. Kilper, and Gary Atkinson. In-network caching effect on optimal energy consumption in content-centric networking. In *IEEE International Conference on Communications Workshops (ICC)*, pages 2889–2894, June 2012.

[CM10] Luca Chiaraviglio and Ibrahim Matta. Greencoop: Cooperative green routing with energy-efficient servers. In *First International Conference on Energy-Efficient Computing and Networking (e-Energy)*, pages 191–194, April 2010.

[CM11] Luca Chiaraviglio and Ibrahim Matta. An energy-aware distributed approach for content and network management. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 337 –342, April 2011.

[CMN11] Luca Chiaraviglio, Marco Mellia, and Fabio Neri. Minimizing isp network energy cost: Formulation and solutions. *IEEE/ACM Transactions on Networking*, 20:463–476, April 2011.

[Coh11] Nathann Cohen. *Three years of graphs and music: some results in graph theory and its applications*. PhD thesis, Université de Nice Sophia-Antipolis, 2011.

[CPL] CPLEX. URL: http://www.ibm.com/software/integration/optimization/cplex-optimization-studio/.

[Dan09] Christian Dannewitz. Netinf: An information-centric design for the future internet. In *Proceedings of 3rd GI/ITG KuVS Workshop on The Future Internet*, May 2009.

[ER60] Pál Erdős and Alfréd Rényi. On the evolution of random graphs. In *Publication of The Mathematical Institute of The Hungarian Academy of Sciences*, pages 17–61, 1960.

[GAKG11] Kyle Guan, Gary Atkinson, Daniel C. Kilper, and Ece Gulsen. On the energy efficiency of content delivery architectures. In *IEEE International Conference on Communications Workshops (ICC)*, pages 1–6, june 2011.

[GMM12] F. Giroire, D. Mazauric, and J. Moulierac. Energy efficient routing by switching-off network interfaces. *Energy-Aware Systems and Networking for Sustainable Initiatives*, pages 207–237, 2012.

[GWS12] Chang Ge, Ning Wang, and Zhili Sun. Optimizing server power consumption in cross-domain content distribution infrastructures. In *IEEE International Conference on Communications Workshops (ICC)*, pages 2628–2633, June 2012.

[HH10] G. Haßlinger and O. Hohlfeld. Efficiency of caches for content distribution on the internet. In *Teletraffic Congress (ITC), 2010 22nd International*, pages 1–8. IEEE, 2010.

[JNWC11] Chamil Jayasundara, Ampalavanapillai Nirmalathas, Elaine Wong, and Chien Aun Chan. Energy efficient content distribution for vod services. In *Optical Fiber Communication Conference*. Optical Society of America, 2011.

[JST+09] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of ACM CoNEXT*, pages 1–12, December 2009.

[Kar72] RM Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[LL10] Jeffrey T Linderoth and Andrea Lodi. Milp software. *Wiley encyclopedia of operations research and management science*, 2010.

[LSG12] Zhe Li, Gwendal Simon, and Annie Gravey. Caching policies for in-network caching. In *21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1 –7, August 2012.

[Min06] Michel Minoux. Multicommodity network flow models and algorithms in telecommunications. In *Handbook of Optimization in Telecommunications*, pages 163–184. Springer, 2006.

[MLG+11] U Mandal, C Lange, A Gladisch, P Chowdhury, and B Mukherjee. Energy-efficient content distribution over telecom network infrastructure. In *Transparent Optical Networks (ICTON), 2011 13th International Conference on*, pages 1–4. IEEE, 2011.

[MSS12] Vimal Mathew, Ramesh K. Sitaraman, and Prashant Shenoy. Energy-aware load balancing in content delivery networks. In *Proceedings IEEE INFOCOM*, pages 954 –962, March 2012.

[OCZ] OCZ Technology Group. URL: `http://www.ocztechnology.com/ocz-revodrive-3-x2-pci-express-ssd.html`.

[PCP12] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking (ICN)*, pages 55–60. ACM, 2012.

[PYRK08] Sanjoy Paul, Roy Yates, Dipankar Raychaudhuri, and Jim Kurose. The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. In *Innovations in NGN: Future Network and Services*, pages 367 –374, May 2008.

[RK09] Elisha J. Rosensweig and Jim Kurose. Breadcrumbs: Efficient, best-effort content location in cache networks. In *IEEE INFOCOM*, pages 2631 –2635, April 2009.

[RT87] Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. Wiley, 1998.

[SLW11] Yunlong Song, Min Liu, and Yuwei Wang. Power-aware traffic engineering with named data networking. In *Seventh International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 289–296, Dec. 2011.

[SND] SNDLib. URL: `http://sndlib.zib.de`.

[VHI12] W. Van Heddeghem and F. Idzikowski. Equipment power consumption in optical multilayer networks-source data. Technical report, Technical Report IBCN-12-001-01 (January 2012), available at http://powerlib. intec. ugent. be, 2012.

[Web08] M. Webb. Smart 2020: Enabling the low carbon economy in the information age. *The Climate Group London*, 2008.

[ZYLZ10] Mingui Zhang, Cheng Yi, Bin Liu, and Beichuan Zhang. Greente: Power-aware traffic engineering. In *ICNP'10: IEEE International Conference on Network Protocols*, pages 21–30, 2010.

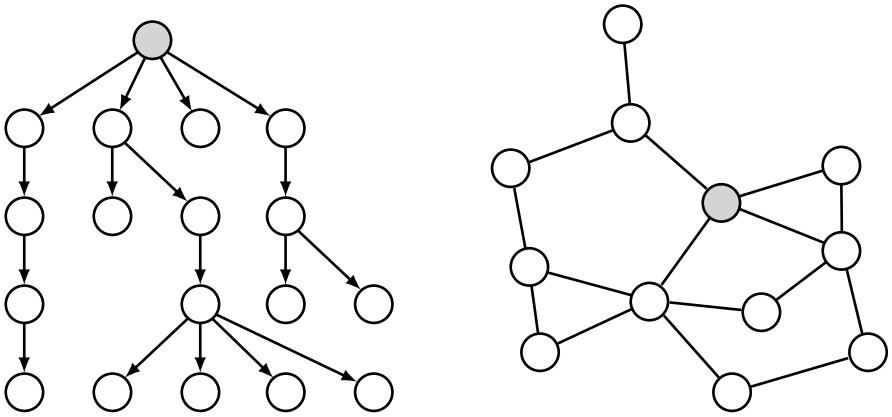
CHAPTER 4

Maintaining Balanced Trees For Structured Distributed Streaming Systems

In this chapter, we move to content distribution in the application layer. Peer-to-peer networks reduce the broadcasting redundancy by allowing clients to share the content among themselves. There are two major classes of peer-to-peer streaming networks: structured and unstructured. While structured networks allow for lower overheads and higher bandwidth utilization, concerns are raised about their robustness, up to the point that virtually all deployed solutions are unstructured. In this chapter, we attempt to answer these concerns. We show that repairing the structure of a generic broadcasting tree, after any failure, can happen in a short time.

4.1 Preliminary: live streaming overlay networks

There are two major classes of peer-to-peer live video streaming approaches. Their names vary among publications, the first one being named either *unstructured*, *mesh-based*, *gossiping* or *torrent-like*; the second named either *structured* or *tree-based*. This classification was already used in [ZLLY05].



(a) Structured (b) Unstructured

Figure 4.1: A visualization of both major overlay classes

Early systems, like [hCRZ00], influenced by IP multicast, attempted at constructing a multicast tree to stream the media. Battling all the possible shortcomings, this simple idea has evolved into elaborate algorithms like SplitStream, proposed in [CDK+03]. Throughout all the possible variations, the signature of this group of systems is active maintenance of an overlay structure that clearly defines the data flow, thus the name *structured overlays*.

On the other hand we have systems inspired by the BitTorrent, one of the best-known peer-to-peer protocols, described by Bram Cohen, its original author, in [Coh03]. The core idea of this class of overlays is organizing the peers into a random, highly-connected graph and disseminating the data using a simple, probabilistic algorithm. The first instance of an unstructured system was proposed in [BLBS03] as a way of enhancing a single-tree overlay. In [ZLLY05] it was the base for the first peer-to-peer network that streamed video to a big number of simultaneous clients. The distinguishing characteristic of this group of networks is that they do not have an overlay structure that would define the data flow, thus the name *unstructured overlays*.

Figure 4.1 visualizes both concepts. It is important to note, that the divide is mostly ideological. Watching forwarding history of any single packet will give us a tree. In [MR09] the protocol is plainly unstructured, but an elaborate structure emerges in the analysis. A system can start

by random forwarding, but retain good paths effectively turning into a structured one, like [LQK+08]. Finally there are systems, which fit neither of the above descriptions. An example of that is AQCS, proposed in [GLL09]. Despite that the classification is well entrenched, up to the point of studies comparing both classes, like [MRG07].

Unstructured systems are widely regarded the better choice, to the extent that, up to our best knowledge, no structured live streaming systems have been deployed in practice. That is often explained by the complexity of making a structured system reliable. However, in this chapter we show that reliability can be ensured, for a simple system, efficiently by a simple algorithm.

4.2 Publication

The remainder of this chapter corresponds to *Maintaining Balanced Trees For Structured Distributed Streaming Systems* by F. Giroire, R. Modrzejewski, N. Nisse and S. Pérennes, which is accepted for publication in the proceedings of 20th International Colloquium on Structural Information and Communication Complexity.

4.3 Introduction

Trees are inherent structures for data dissemination in general and particularly in peer-to-peer live streaming networks. Fundamentally, from the perspective of a peer, each atomic piece of content has to be received from some source and forwarded towards some receivers. Moreover, most of the actual streaming mechanisms ensure that a piece of information is not transmitted again to a peer that already possesses it. Therefore, this implies that dissemination of a single fragment defines a tree structure. Even in *unstructured* networks, whose main characteristic is lack of defined structure, many systems look into perpetuating such underlying trees, e.g. the second incarnation of Coolstreaming [LQK+08] or PRIME [MR09].

Unsurprisingly, early efforts into designing peer-to-peer video streaming concentrated on defining tree-based structures for data dissemination. These have been quickly deemed inadequate, due to fragility and unused bandwidth at the leaves of the tree. One possible fix to these weaknesses

was introduced in SplitStream [CDK+03]. The proposed system maintains multiple concurrent trees to tolerate failures, and internal nodes in a tree are leaf nodes in all other trees to optimize bandwidth. The construction of intertwined trees can be simplified by a randomized process, as proposed in Chunkyspread [VYF06], leading to a streaming algorithm performing better over a range of scenarios.

As found in [LQK+08], node churn is the main difficulty for live streaming networks, especially those trying to preserve structure. On the other hand, in [ZSC10] authors embrace change. Their stochastic optimization approach relies on constant random creating and breaking of relationships. To ensure network connectivity, nodes are said to keep open connections with hundreds of potential neighbours. Another approach, displayed in [LXHL11], is churn-resiliency by maintaining redundancy within the network structure. Although concentrating on a different field, authors of [PTT09] face a similar to our own problem of maintaining balanced trees, needed for connecting wireless sensors. However, their solution is periodical rebuilding the whole tree from scratch. Our solution aims at minimizing the disturbance of nodes, whose ancestors were not affected by recent failures, as well as minimizing the redundancy in the network.

The analysis of these systems focus on the feasibility, construction time and properties of the established overlay network, see for example [CDK+03, VYF06] and [DFC07] for a theoretical analysis. But these works usually abstract over the issue of tree maintenance. Generally, in these works, when some elements (nodes or links) of the networks fail, the nodes disconnected from the root execute the same procedure as for initial connection. To the best of our knowledge, there are no theoretical analysis on the efficiency of tree maintenance in streaming systems, reliability is estimated by simulations or experiments as in [CDK+03].

In this paper, we tackle this issue by designing an efficient maintenance scheme for trees. Our distributed algorithm ensures that the tree recovers fast to a "good shape" after one or multiple failures occur. We give analytic upper bounds of the convergence time. To the best of our knowledge, this is the first theoretical analysis of a repair process for live streaming systems. While the $O(n^2)$ worst case bound seems high, simulations shown in Section 4.7 suggest that the average case is closer to $O(\log n)$, which is lower than the conceivable time of rebuilding a tree

from scratch.

The problem setting is as follows. A single source provides live media to some nodes in the network. This source is the single reliable node of the network, all other peers may be subject to failure. Each node may relay the content to further nodes. Due to limited bandwidth, both source and any other node can provide media to a limited number $k \geq 2$ of nodes. The network is organized into a logical tree, rooted at the source of media. If node x forwards the stream towards node y, then x is the parent of y in the logical tree. Note that the delay between broadcasting a piece of media by the source and receiving by a peer is given by its distance from the root in the logical tree. Hence our goal is to minimize the tree depth, while following degree constraints.

As shown in [LQK+08], networks of this kind experience high rate of node joins and leaves. Leaves can be both graceful, where a node informs about imminent departure and network rearranges itself before it stops providing to the children, or abrupt (e.g. due to connection or hardware failure). In this work, we assume a *reconnection process*: when a node leaves, its children reattach to its parent. This can be done locally if each node stores the address of its grandfather in the tree. Note that this process is performed independently of the bandwidth constraint, hence after multiple failures, a node may become the parent of many nodes. The case of concurrent failures of father and grandfather can be handled by reattaching to the root of the tree. Other more sophisticated reconnection processes have been proposed, see for example [HLP+07].

This process can leave the tree in a state where either the bandwidth constraints are violated (the degree of a node is larger than k) or the tree depth is not optimal. Thus, we propose a distributed *balancing process*, where based on information about its degree and the subtree sizes of its children, a node may perform a local operation at each turn. We show that this balancing process, starting from any tree, converges to a balanced tree and we evaluate the convergence time.

Related Work. Construction of spanning trees has been studied in the context of self-stabilizing algorithms. Herault et al. propose in [HLP+07] a new analytic model for large scale systems. They assume that any pair of processes can communicate directly, under condition of knowing receiver's identifier, what is the case in Internet Protocol. They additionally assume a discovery service and a failure detection service. Under this

model they propose and prove correctness of an algorithm constructing a spanning tree over a set of processes. Similar assumptions have been used by Caron et al. in [CDPT08] to construct a distributed prefix tree and by Bosilca et al. in [BCH+09] to construct a binomial graph (Chord-like) overlay.

In this paper we assume the results of these earlier works: nodes can reliably communicate, form connections and detect failures. We do not analyze these operations at message level. Furthermore, we analyze the overlay assuming it is already a spanning tree. However, it may have an arbitrary shape, e.g. be a path or a star (all nodes connected directly to the root). This can be regarded as maintaining the tree after connection or failure of an arbitrary number of nodes.

Our results. In Section 4.4, we provide a formal definition of the problem and propose a distributed algorithm for the balancing process. The process works in a synchronous setting. At each turn, all nodes are sequentially scheduled by an adversary and must execute the process. In Section 4.5, we show that the balancing process always succeeds in $O(n^2)$ turns. Then, in Section 4.6, we study a restricted version of the algorithm in which a node performs an operation only when the subtrees of its children are balanced. In this case, we succeeded in obtaining a tight bound of $\Theta(n \log n)$ on the number of turns for the worst tree. Finally, we show that the convergence is in fact a lot faster in average for a random tree and takes a logarithmic number of turns.

4.4 Problem and Balancing Process

In this section, we present the main definitions and settings used throughout the paper, then we present our algorithm and prove some simple properties of it.

Notations

This section is devoted to some basic notations.

Let $n \in \mathbb{N}^*$. Let $T = (V, E)$ be a n-node tree rooted in $r \in V$. Let $v \in V$ be any node. The *subtree T_v rooted at v* is the subtree consisting of v and all its descendants. In other words, if $v = r$, then $T_v = T$ and,

otherwise, let e be the edge between v and its parent, T_v is the subtree of $T \setminus e = (V, E \setminus \{e\})$ containing v. Let $n_v = |V(T_v)|$.

Let $k \geq 2$ be an integer. A node $v \in V(T)$ is *underloaded* if it has at most $k-1$ children and at least one of these children is not a leaf. v is said *overloaded* if it has at least $k+1$ children. Finally, a node v with k children is *imbalanced* if there are two children x and y of v such that $|n_x - n_y| > 1$. A node is *balanced* if it is neither underloaded, nor overloaded nor imbalanced. Note that a leaf is always balanced.

A tree is a *k-ary tree* if it has no nodes that are underloaded or overloaded, i.e., all nodes have at most k children and a node with $< k$ children has only leaf-children. A rooted k-ary tree T is *k-balanced* if, for each node $v \in V(T)$, the sizes of the subtrees rooted in the children of v differ by at most one. In other words, a rooted tree is k-balanced if and only if all its nodes are balanced.

As formalized by the next claim, k-balanced trees are good for our live streaming purpose since such overlay networks (k being small compared with n) ensure a low dissemination delay while preserving bandwidth constraints.

Claim 1. *Let T be a n-node rooted tree. If T is k-balanced, then each node of T is at distance at most $\lfloor \log_k n \rfloor$ from r.*

Distributed Model and Problem

Nodes are autonomous entities running the same algorithm. Each node v has a local memory where it stores the size n_v of its subtree, the size of the subtrees of its children and the size of the subtrees of its grand-children, i.e., for any child x of v and for any child y of x, v knows n_x and n_y.

Computations performed by the nodes are based only on the local knowledge, i.e., the information present in the local memory and that concerns only nodes at distance at most 2. We consider a synchronous setting. That is, the time is slotted in turns. At each turn, any node may run the algorithm based on its knowledge and, depending on the computation, may do one of the following *operations*. In the algorithm we present, each operation done by a node v consists of rewiring at most two edges at distance at most 2 from v. More precisely, let v_1, v_k and

v_{k+1} be children of v, a be a child of v_1 and b be a child of v_k (if any). The node v may perform:

Pull operation replace the edge $\{v_1, a\}$ by the edge $\{v, a\}$. A grandchild a of v then becomes a child of v. This operation is denoted by PULL(a) and illustrated in Figure 4.2a;

Push operation replace the edge $\{v, v_{k+1}\}$ by the edge $\{v_k, v_{k+1}\}$. A child v_{k+1} of v then becomes a child of another child v_k of v. This operation is denoted by PUSH (v_{k+1,v_k}), see Figure 4.2b;

Swap operation replace the edges $\{v_1, a\}$ and $\{v_k, b\}$ by the edges $\{v_1, b\}$ and $\{v_k, a\}$. The children v_1 and v_k of v exchange two of their own children a and b. This operation is denoted by SWAP(a,b) and an example is given in Figure 4.3c. Here, a or b may not exist, in which case, one of v_1 and v_k "wins" a new child while the other one "looses" a child. This case is illustrated in Figure 4.3d.

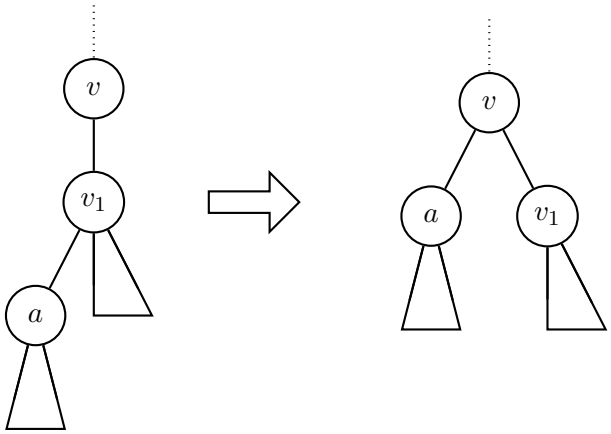
In all cases, the local memory of the at most $k^2 + 1$, including the parent of v, nodes that are concerned are updated. Note that each of these operations may be done using a constant number of messages of size $O(\log n)$.

In this setting, at every turn, all nodes sequentially run the algorithm. In order to consider the worst case scenario, the order in which all nodes are scheduled during one turn is given by an adversary. The algorithm must ensure that after a finite number of turns, the resulting tree is k-balanced. We are interested in time complexity of the worst case scenario of the repair. That is, the performance of the algorithm is measured by the maximum number of turns after which the tree becomes k-balanced, starting from any n-node tree.

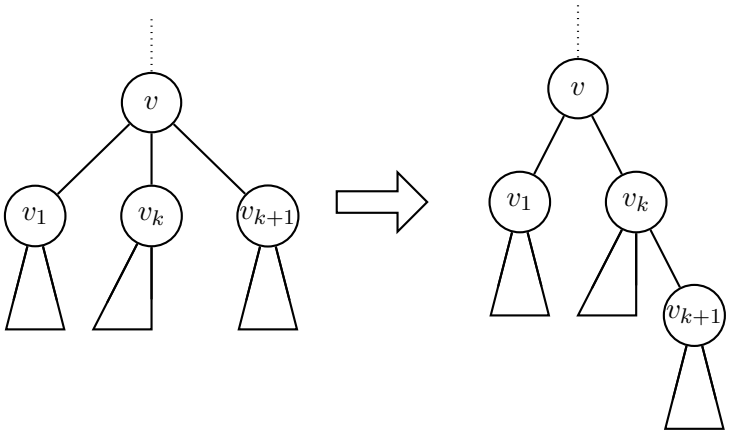
The Balancing Process

In this section, we present our algorithm, called *balancing process*. We prove some basic properties of it. In particular, while the tree is not k-balanced, the balancing process ensures that at least one node performs an operation. In the next sections, we prove that the balancing process actually allows to reach a k-balanced tree after a finite number of steps.

At each turn, a node v executes the algorithm described on Figure 4.3. To summarize, an underloaded node does a PULL, an over-

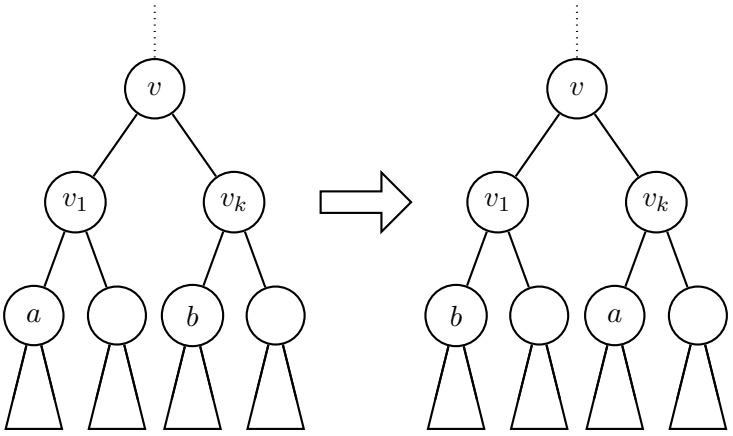


(a) PULL(a)

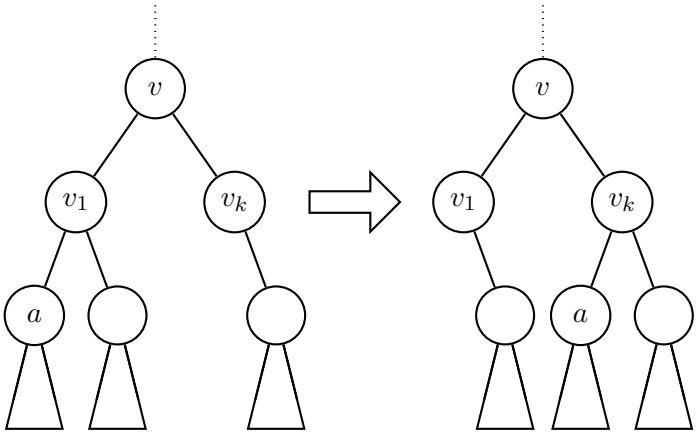


(b) PUSH(v_{k+1}, v_k)

Figure 4.2: Operations performed by node v in the balancing process



(c) SWAP(a, b)



(d) SWAP$(a, \emptyset)$

Figure 4.2: Operations performed by node v in the balancing process

Algorithm executed by a node v in a tree T. If v is not a leaf, let $(v_1, v_2, \cdots, v_d)$ be the $d \geq 1$ children of v ordered by subtree-size, i.e., $n_{v_1} \geq n_{v_2} \geq \cdots \geq n_{v_d}$.

1. **If** v is underloaded (then $d < k$), let a be a child of v_1 with biggest subtree size. **Then** node v executes PULL(a). *// That is, a becomes a child of v.*

2. **Else if** v is overloaded (then $d > k \geq 2$), **then** node v executes PUSH(v_{k+1}, v_k).

   // That is, v_{k+1} becomes a child of v_k.

3. **Else if** v is imbalanced (then $d = k$) **and if** v_1 and v_k are not overloaded, let a and b be two children of v_1 and v_k respectively such that $|n_{v_1} - n_a + n_b - (n_{v_k} - n_b + n_a)|$ is minimum (a (resp. b) may be not defined, i.e., $n_a = 0$ (resp., $n_b = 0$), if v_1 (resp v_k) is underloaded).

   Then node v execute SWAP(a, b). *// That is, a and b exchange their parent.*

Figure 4.3: Balancing Process

loaded node does a PUSH and an imbalanced node (whose children are not overloaded) does a SWAP operation. Note that a SWAP operation may exchange a subtree with an empty subtree, but cannot create an overloaded node. Intuitively, the children affected by PUSH and PULL are chosen to get probably the least imbalance (reduce the biggest or merge the two small). It is important to emphasise that the balancing process requires no memory of the past operations.

Note that if the tree if k-balanced, no operation are performed, and that, if the tree is not, at least one operation is performed.

Claim 2. *If T is not k-balanced, and all nodes execute the balancing process, then at least one node will do an operation.*

In the next section, we prove that, starting from any tree, the number of operations done by the nodes executing the balancing process is bounded. Together with the previous claim, it allows to prove

Theorem 1. *Starting from any tree T where each node executes the balancing process, after a finite number of steps, T eventually becomes k-balanced.*

Before proving the above result in next Section, we give a simple lower bound on the number of turns required by the Balancing Process. A *star* is a rooted tree where any non root-node is a leaf.

Lemma 1. *If the initial tree is a n-node star, then at least $\Omega(n)$ turns are needed before the resulting tree is k-balanced.*

4.5 Worst case analysis

In this Section we obtain an upper bound of $O(n^2)$ turns needed to balance the tree. We prove it using a potential function, whose initial value is bounded, integral and positive, may rise in a bounded number of turns and, otherwise, strictly decreases. For clarity of presentation we assume we want to obtain a 2-balanced tree. The proofs can be extended to larger k. Due to lack of space, most of them are only sketched here and can be found in [GRNP13].

Lemma 2. *Starting from any n-node rooted tree T, after having executed the Balancing Process during $O(n)$ turns, no node will do a* PUSH *operation anymore.*

Let $\mathcal{Q}$ be the sum over all nodes $u \in T$ of the distance between u and the root.

Lemma 3. *Starting from any n-node rooted tree T, there are at most $O(n^2)$ distinct (not necessarily consecutive) turns with a* PULL *operation. More precisely, the sum of the sizes of the subtrees that are pulled during the whole process does not exceed n^2.*

Proof. First, by Lemma 2, there are no PUSH operations after $O(n)$ turns. Note that a SWAP operation does not change $\mathcal{Q}$. Moreover, a PULL operation of a subtree T_v makes $\mathcal{Q}$ decrease by n_v. Since $\mathcal{Q} = \sum_{u \in V(T)} d(u, r) \leq n^2$, the sum of the sizes of the subtrees that are pulled during the whole process does not exceed n^2. □

Potential function. To prove the main result of this section, we define a potential function and show that: (1) the initial value of the potential function is bounded; (2) its value may raise due to PULL operations, but in a limited number of turns and by a bounded amount; (3) a SWAP operation may not increase its value; (4) if no PUSH nor PULL operation are done, there exists at least one node doing a SWAP operation, strictly decreasing the potential function.

We tried simple potential functions first. However, they led either to an unbounded number of turns with non-decreasing value, or to a larger upper bound. For example, it would be natural to define the potential of a node as the difference between its subtree sizes. For this potential function, (1) (2) and (3) are true, but, unfortunately, for some trees the potential function does not decrease during a turn. This function can be patched so that each operation makes the potential decrease: multiplying the potential of a node by its distance to the root. However, the potential in this case can reach $O(n^3)$.

The potential function giving the $O(n^2)$ bound is defined as follows. Recall that we consider a n-node tree T rooted in r such that all nodes have at most two children. Let $E_0 = n$ and, for any $0 \leq i \leq \lceil \log(n+1) \rceil$, let $E_i = 2E_{i+1} + 1$. Note that $(E_i)_{i \leq \lceil \log(n+1) \rceil}$ is strictly decreasing, and $0 < E_{\lceil \log(n+1) \rceil} \leq 1$. Intuitively, E_i is the mean-size of a subtree rooted in a node at distance i from the root in a balanced tree with n nodes.

Let K_i be the set of nodes of T at distance exactly $i \geq 0$ from the root and $|K_i| = k_i$, and, for any $0 \leq i \leq \lceil \log(n+1) \rceil$, let $m_i = 2^i - k_i$. Intuitively, m_i represents the number of nodes, at distance i from the root, missing compared to a complete binary tree.

For any $v \in V(T)$ at distance $0 \leq i \leq \lceil \log(n+1) \rceil$ from the root, the *default* of v, denoted by $\mu(v)$, equals $n_v - \lceil E_i \rceil$ if $n_v > E_i$ and $\lfloor E_i \rfloor - n_v$ otherwise. Note that $\mu(v) \geq 0$ since n_v is an integer.

Let the *potential at distance i from r*, $0 \leq i \leq \lceil \log(n+1) \rceil$, be $P_i = m_i \cdot \lfloor E_i \rfloor + \sum_{u \in K_i} \mu(u)$. Finally, let us define the *potential* $\mathcal{P} = \sum_{0 \leq i \leq \lceil \log(n+1) \rceil} P_i$. Since $\mu(u) \leq n$ for any $u \in V(T)$, and $\sum_{0 \leq i \leq \lceil \log(n+1) \rceil} m_i + k_i \leq 2n$, then $\mathcal{P}(T) = O(n^2)$.

Lemma 4. *For any n-node rooted tree T, a* PULL *operation of a subtree T_v may increase the potential $\mathcal{P}$ by at most $2n_v$.*

Let v be a node at distance $\lceil \log(n+1) \rceil > i \geq 0$ from the root r

of T. v is called *i-median* if it has one or two children a and b and $n_a > E_{i+1} > n_b$ (possibly v has exactly one child and $n_b = 0$).

Lemma 5. *For any n-node rooted tree T, a* SWAP *operation executed by any node v does not increase the potential $\mathcal{P}$. Moreover, if v is $(i-1)$-median then $\mathcal{P}$ strictly decreases by at least one.*

This lemma is proved by calculating the new potential, in all the possible cases of relative sizes of the children and E_i before and after the operation.

Let v be a node at distance $0 \leq i < \lceil \log(n+1) \rceil - 1$ from the root r of T. v is called *i-switchable* if it has one or two children a and b and $n_a > E_{i+1} > n_b$ (possibly v has only exactly child, and $n_b = 0$), $n_a - n_b \geq 2$ and none of its ancestors can execute a SWAP operation. Note that, if a node is *i-switchable*, then it is i-median.

Lemma 6. *Let T be a tree where no* PUSH *nor* PULL *operation is possible. If a node v is i-switchable, then either v can do a* SWAP *operation, or $0 \leq i < \lceil \log(n+1) \rceil - 2$ and it has a $(i+1)$-switchable child.*

Lemma 7. *At each turn when no* PULL *nor* PUSH *operations are done, if the tree is not balanced, then there is a i-switchable node, $0 \leq i < \lceil \log(n+1) \rceil - 1$.*

Theorem 2. *Starting from any n-node rooted tree, the balancing process reaches a 2-balanced tree in $O(n^2)$ turns.*

Proof. By Lemma 2, after $O(n)$ turns, no PUSH operations are executed anymore and all nodes have at most two children. From then, there may have only pull or SWAP operations. Moreover, by Claim 2, there is at least one operation per turn while T is not balanced. From Lemma 3, there are at most $O(n^2)$ turns with a PULL operation. Once no PUSH operations are executed anymore, from Lemmata 3, 4 and 5, potential $\mathcal{P}$ can increase by at most $O(n^2)$ in total (over all turns). Moreover, by Lemma 5, if a i-median node executes a SWAP operation, the potential $\mathcal{P}$ strictly decreases by at least one.

By Lemma 7, at each turn when no pull nor PUSH operations are done, there is an i-switchable node, $0 \leq i < \lceil \log(n+1) \rceil - 1$. Thus, by Lemma 6, at each such turn, there is an i-switchable that can execute a

SWAP operation. Since a i-switchable node is i-median ($0 \leq i < \lceil \log(n+1) \rceil - 1$), by Lemma 5, the potential $\mathcal{P}$ strictly decreases by at least one.

The result then follows from the fact that $\mathcal{P} \leq n^2$. □

4.6 Adding an extra global knowledge to the nodes

In this section, we assume an extra global knowledge: each node knows whether it has a descendant that is not balanced. This extra information is updated after each operation. Then, our algorithm is modified by adding the condition that any node v executing the balancing process can do a PULL or SWAP operation only if all its descendants are balanced. Adding this property allows to prove better upper bounds on the number of steps, by avoiding conflict between an operation performed by a node and an operation performed by one of its not balanced descendant. We moreover prove that this upper bound for our algorithm is asymptotically tight, reached when input tree is a path. The approach presented in this section is specific for $k = 2$. I.e., the objective of the Balancing Process is to reach a 2-balanced tree.

First, we define a function f used to bound the number of turns needed to balance a tree consisting of two balanced subtrees and a common ancestor. Let $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be the function defined recursively as follows.

$$
\begin{array}{ll}
\forall a \geq 0, & f(a,a) = 0 \\
\forall a \geq 1, & f(a,a-1) = 0 \\
\forall a \geq 2, & f(a,0) = 1 + f(\left\lfloor \frac{a-1}{2} \right\rfloor, 0) \\
\forall a > 2, \forall 1 \leq b < a-1, & f(a,b) = 1 + \max\left(f(\left\lceil \frac{a-1}{2} \right\rceil, \left\lfloor \frac{b-1}{2} \right\rfloor), f(\left\lfloor \frac{a-1}{2} \right\rfloor, \left\lceil \frac{b-1}{2} \right\rceil)\right)
\end{array}
$$

Lemma 8. *For any $a \geq 0$, $a \geq b \geq 0$, $f(a,b) \leq \max\{0, \log_2 a\}$.*

Now, we give a function bounding the number of turns needed to balance any tree of a given size. Let $g : \mathbb{N} \to \mathbb{N}$ be the function defined recursively as follows.

$$
\begin{array}{ll}
\forall n \in \{0,1\}, & g(n) = 0 \\
\forall n > 1, & g(n) = \max_{a \geq b \geq 0, a+b=n-1}(\max\{g(a), g(b)\} + f(a,b))
\end{array}
$$

Lemma 9. *For any $n \geq 0$, $g(n) \leq \max\{0, n \log_2 n\}$.*

We now state our main results:

Theorem 3. *Starting from any n-node rooted tree, the balancing process with global knowledge reaches a 2-balanced tree in $O(n \log n)$ turns.*

Next theorem shows that there are trees starting from which the balancing process actually uses a number of turns of the order of the above upper bound.

Theorem 4. *Starting from an n-node path rooted in one of its ends, the balancing process with global knowledge reaches a 2-balanced tree in $\Omega(n \log n)$ turns.*

4.7 Simulations

In the previous sections we obtained upper and lower bounds for the maximum number of turns needed to balance a tree of a given size. A significant gap between those bounds raises the question: which bound is closer to what happens for random instances? We investigate the performance of the algorithm running an implementation under a discrete event simulation. Scheduling of nodes within a turn is given by a simple adversary algorithm. First, it detects which nodes can perform no operation. It schedules them to move first, to ensure that they do not perform operations enabled by operations of other nodes. Then, it schedules the remaining nodes in a random order.

The process starts in a random tree. It is obtained by assigning random weights to a complete graph and building a minimum weight spanning tree over it. Figure 4.4 displays the number of turns it took to balance trees of progressing sizes. For each size the numbers are aggregated over 10000 different starting trees. The solid line marks the average, dotted lines the minimum and maximum numbers of turns and error bars show the standard deviation.

What can be seen from this figure, is that the number of turns spent to balance a random tree progresses logarithmically in regard to the tree size. This holds true both for average and the worst cases encountered. This is significantly less even than the lower bound on maximum time. This is because that comes from the particular case of star as the starting tree, which is randomly obtained with probability $\frac{1}{n!}$ and did not occur in our experiments for bigger values of n.

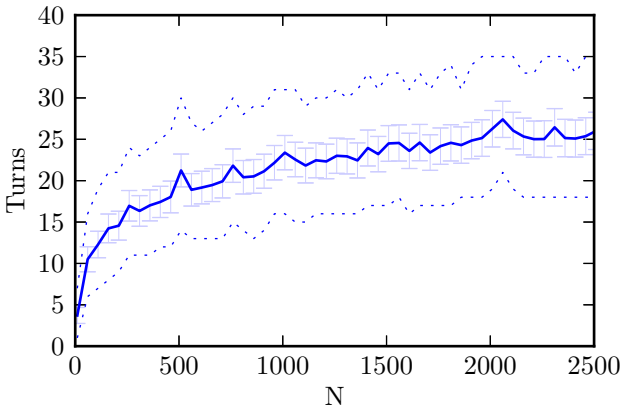


Figure 4.4: Balancing a random tree

4.8 Conclusions and future research

We have proposed a distributed tree balancing algorithm and shown following properties. The algorithm does stop only when the tree is balanced. After at most $\Omega(n)$ turns there are no overloaded nodes in the tree, what corresponds to a broadcast tree where every node receives content. This bound is reached when the starting tree is a star. Balancing process after there are no overloaded nodes lasts at most $O(n^2)$ turns. With the additional restriction that a node acts only if all of its descendants are balanced, the number of turns to balance any tree is $O(n \log n)$. This bound is reached when the starting tree is a path.

An obvious, but probably hard, open problem is closing the gap between the $O(n^2)$ upper bound and the $\Omega(n)$ lower bound on balancing time. Another possibility is examination of the algorithm's average behaviour, which as hinted by simulations should yield $O(\log n)$ bound on balancing time.

The algorithm itself can be extended to handle well the case of trees that are not regular. Furthermore, in order to approach a practical system, moving to multiple trees would be highly beneficial. Allowing the algorithm to stop with more imbalance, where children are allowed to differ by a given threshold instead of one, could lead to a faster convergence.

4.9 Bibliography

[BCH+09] G. Bosilca, C. Coti, T. Herault, P. Lemarinier, and J. Dongarra. Constructing resiliant communication infrastructure for runtime environments. In *International Conference in Parallel Computing*, pages 441–451, 2009.

[BLBS03] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, pages 102–113. ACM, 2003.

[CDK+03] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, page 313, 2003.

[CDPT08] E. Caron, A.K. Datta, F. Petit, and C. Tedeschi. Self-stabilization in tree-structured peer-to-peer service discovery systems. In *IEEE Symposium on Reliable Distributed Systems*, pages 207–216, 2008.

[Coh03] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72. Citeseer, 2003.

[DFC07] G. Dan, V. Fodor, and I. Chatzidrossos. On the performance of multiple-tree-based peer-to-peer live streaming. In *26th IEEE International Conference on Computer Communications*, pages 2556–2560, 2007.

[GLL09] Y. Guo, C. Liang, and Y. Liu. Aqcs: Adaptive queue-based chunk scheduling for P2P live streaming. *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, pages 433–444, 2009.

[GRNP13] Frederic Giroire, Modrzejewski Remigiusz, Nicolas Nisse, and Stéphane Pérennes. Maintaining Balanced Trees For Structured Distributed Streaming Systems. Research Report

RR-8309, INRIA, May 2013. URL: `http://hal.inria.fr/hal-00824269`.

[hCRZ00] Y. hua Chu, S.G. Rao, and H. Zhang. A case for end system multicast. In *Proc. of ACM Sigmetrics*, pages 1–12, 2000.

[HLP+07] T. Herault, P. Lemarinier, O. Peres, L. Pilard, and J. Beauquier. A model for large scale self-stabilization. In *IEEE Parallel and Distributed Processing Symposium*, pages 1–10, 2007.

[LQK+08] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang. Inside the new coolstreaming: Principles, measurements and performance implications. In *27th IEEE International Conference on Computer Communications*, 2008.

[LXHL11] Zhenyu Li, Gaogang Xie, Kai Hwang, and Zhongcheng Li. Churn-resilient protocol for massive data dissemination in p2p networks. *IEEE Parallel and Distributed Systems*, 22(8):1342–1349, 2011.

[MR09] N. Magharei and R. Rejaie. Prime: Peer-to-peer receiver-driven mesh-based streaming. *IEEE/ACM Transactions on Networking*, 17(4):1052–1065, 2009.

[MRG07] N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1424–1432. Ieee, 2007.

[PTT09] Meng-Shiuan Pan, Chia-Hung Tsai, and Yu-Chee Tseng. The orphan problem in zigbee wireless networks. *IEEE Transactions on Mobile Computing*, 8(11):1573–1584, 2009.

[VYF06] Vidhyashankar Venkataraman, Kaouru Yoshida, and Paul Francis. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *14th IEEE International Conference on Network Protocols*, pages 2–11, 2006.

[ZLLY05] X. Zhang, J. Liu, B. Li, and T.S.P. Yum. CoolStreaming/DONet: A data-driven overlay network for efficient live

media streaming. In *proceedings of IEEE Infocom*, volume 3, pages 13–17, 2005.

[ZSC10] Shaoquan Zhang, Ziyu Shao, and Minghua Chen. Optimal distributed p2p streaming under node degree bounds. In *18th IEEE International Conference on Network Protocols*, pages 253–262, 2010.

CHAPTER 5

Analysis of the Repair Time in Distributed Storage Systems

In this final contribution towards reducing network inefficiencies, we move from content distribution to distributed applications. One such application, with big bandwidth requirements, are online backups. A conservative approach to this task employs data centers. However, these usually are far away from the users. Instead, it is possible to use storage located at the perimeters of other nearby users of a distributed system. This, again, raises questions about reliability. In this chapter, we look into expected data lifetime in a distributed storage system, where nodes are subject to faults and departures and are connected with a limited bandwidth. This work makes use of queuing theory and more generally Markov chains, which are introduced in the preliminary section.

5.1 Preliminary: Queues and Markov chains

When looking into the distributed storage system in this chapter, we analyze the distribution of data and perform a Markov chain analysis to deduce the data life time. First, we find out how the interactions of various elements of the system can be hidden behind simple failure

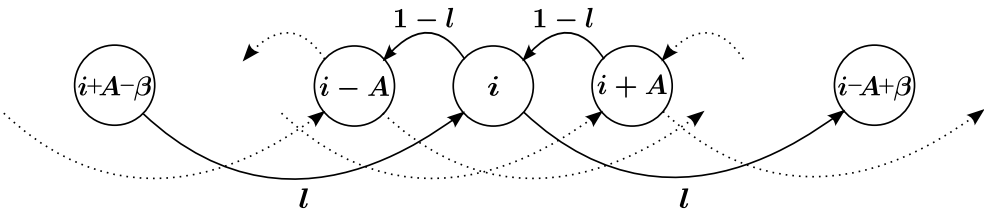


Figure 5.1: An example Markov chain of a $M^\beta/D/1$ queue, with service rate A, arrival rate l and batch size β. Only states which can transition to or from state i are shown.

and repair rates. Then, we model the repair process as a single queue of all blocks in the network that are in need of repair. Queuing theory provides us with tools to deal with such models. A good example of further reading on the subject can be the book [Coo81].

The exact type of queue we have is $M^\beta/D/1$. M^β means that arrivals are batch Poissonian. In fact, there are batches of two possible sizes, each type coming at its own rate of the Poisson distribution. D states that the service time is deterministic, as we know nearly exactly how fast the peers are able to repair blocks. Finally, 1 stands for a single server, which is the whole network. The queue is in fact a simple Markov chain, similar to the one exemplified on figure 5.1. We proceed to find the steady state of it. From there, we can infer interesting qualities, like probability of losing data or bandwidth requirements.

5.2 Publication

The remainder of this chapter corresponds to *Repair Time in Distributed Storage Systems* by F. Giroire, S. K. Gupta, R. Modrzejewski, J. Monteiro and S. Pérennes, which is accepted for publication in the proceedings of 6th International Conference on Data Management in Cloud, Grid and P2P Systems.

5.3 Introduction

Nano datacenters (NaDa) are highly distributed systems owned and controlled by the service provider. This alleviates the need of incentives and mitigates the risk of malicious users, but otherwise they face the same

challenges as peer-to-peer systems. The set-top boxes realizing them are connected using consumer links, which can be relatively slow, unreliable and congested. The devices themselves, compared to servers in a traditional datacenter, are prone to failures and temporary disconnections, e.g. if the user cuts the power supply when not in home. When originally proposed in [VLM+09], they were assumed to be available no more than 85% of the time, with values as low as 7% possible.

In this paper we concentrate on application of NaDa, or any similar *peer-to-peer* system, for backup storage. In this application, users want to store massive amounts of data indefinitely, accessing them very rarely, i.e. only when original copies are lost. Due to risk of peer failures or departures, redundancy data is introduced to ensure long term data survival. To this end, most of the proposed storage systems use either the simple replication or the space efficient erasure codes [WK02], such as the Reed-Solomon or Regenerating Codes [DGWR07].

The redundancy needs to be maintained by a self-repair process. Its speed is crucial to determine the system reliability, as long repairs exponentially increase the probability of losing data. The limiting factor, in this setting, is the upload link capacity.

Imagine a scenario where the system is realized using home connections, out of which an average 128kbps are allocated to the backup application. Furthermore, each device is limited to 300GB, while average data stored is 100GB, redundancy is double, 100 devices take part in each repair and the algorithms are as described in the following sections. A naive back-of-envelope computation gives that the time needed to repair contents of a failed device is 17 hours ($= 100 \cdot 8 \cdot 10^9\text{kb}/(100 \cdot 128\text{kbps})$). This translates, by our model, to a probability of data loss per year (PDLPY) of 10^{-8}. But, taking into account all findings presented in this work, the actual time can reach 9 days. This gives a PDLPY of 0.2, many orders of magnitude more than the naive computation. Hence, it is important to have models that estimate accurately the repair time for limited bandwidth.

Our contribution

We propose a new analytical model that precisely estimates the repair time and the probability of losing data in distributed storage systems. This model takes into account the bandwidth constraints and inherent

workload imbalance (young peers inherently store less data than the old ones, thus they contribute asymmetrically to the reconstruction process) effect on the efficiency. It allows system designers to obtain an accurate choice of system parameters to obtain a desired data durability.

We discuss how far the distribution of the reconstruction time given by the model is from the exponential, classically used in the literature. We exhibit the different possible shapes of this distribution in function of the system parameters. This distribution impacts the durability of the system. We also show a somewhat counter-intuitive result that we can reduce the reconstruction time by using a less bandwidth efficient Regenerating Code. This is due to a degree of freedom given by erasure codes to choose which peers participate in the repair process.

To the best of our knowledge, this is the first detailed model proposed to estimate the distribution of the reconstruction time under limited bandwidth constraints. We validate our model by an extensive set of simulations and by test-bed experimentation using the Grid'5000 platform, see [Gri] for its description.

Related Work

Several works related to highly distributed storage systems have been done, and a number of systems have been proposed [CDH+06, BDET00, BTcC+04, KBC+00], but few theoretical studies exist. In [RP06, ADN07, DA06] the authors use a Markov chain model to derive the lifetime of the system. In these works, the reconstruction times are independent for each fragment. They follow an exponential or geometric distribution, which is a tunable parameter of the models. However, in practice, a large number of repairs start at the same time when a disk is lost, corresponding to tens or hundreds of GBs of data. Hence, the reconstructions are not independent of each other. Furthermore, in these models, only the average analysis are studied and the impact of congestion is not taken into account.

Dandoush et al. in [DAN09] perform a simulation study of the download and the repairing process. They use the NS2 simulator to measure the distribution of the repair time. They state that a hypo-exponential distribution is a good fit for the block reconstruction time. However, again, concurrent reconstructions are not considered. Picconi et al. in [PBS07] study the durability of storage systems. Using simulations

they characterize a function to express the repair rate of systems based on replication. However, they do not study the distribution of the reconstruction time and the case of erasure coding. Venkatesan et al. in [VIH12] study placement strategies for replicated data, deriving a simple approximation for mean time to data loss by studying the expected behaviour of most damaged data block. The closest to our work is [FLP+10] by Ford et al., where authors study reliability of distributed storage in Google, what constitutes a datacenter setting. However, they do not look into load imbalance, their model tracks only one *representative* data fragment and is not concerned by competition for bandwidth.

Organization

The remainder of this paper is organized as follows: in the next section we give some details about the studied system, then in Section 5.5 we discuss the impact of load imbalance. The queueing model is presented in the Section 5.6, followed by its mathematical analysis. The estimations are then validated via an extensive set of simulations in Section A.5. Lastly, in Section 5.8, we compare the results of the simulations to the ones obtained by experimentation.

5.4 System Description

This section outlines the mechanisms of the studied system and our modelling assumptions.

Storage. In this work we assume usage of the Regenerating Codes, as described in [DGWR07], due to their high storage and bandwidth efficiency. More discussion of them follows later in this section. All data stored in the system is divided into *blocks* of uniform size. Each block is further subdivided into s *fragments* of size L_f, with r additional fragments of redundancy. All these $n = s + r$ fragments are distributed among random devices. We assume that in practice this distribution is performed with a Distributed Hash Table overlay like Pastry [RD01]. This, due to practical reasons, divides devices into subsets called *neighbourhoods* or *leaf sets*.

Our model does not assume ownership of data. The device originally introducing a block into the system is not responsible for its storage or

maintenance. We simply deal with a total number of B blocks of data, which results in $F = n \cdot B$ fragments stored in N cooperating devices. As a measure of fairness, or *load balancing*, each device can store up to the same amount of data equal to C fragments. Note that C can not be less than average number of fragments per device $\bar{D} = {}^{\mathrm{F}}\!/_{\mathrm{N}}$.

In the following we treat a device or peer and its disk as synonyms.

Bandwidth. Devices of NaDa are connected using consumer connections. These, in practice, tend to be asymmetric with relatively low upload rates. Furthermore, as the backup application occasionally uploads at maximum throughput for prolonged times, while the consumer expects the application to not interfere with his network usage, we assume it is allocated only a fraction of the actual link capacity. Each device has a maximum upload and download bandwidth, respectively BW_{up} and BW_{down}. We set $BW_{down} = 10BW_{up}$ (in real offerings, this value is often between 4 and 20). The bottleneck of the system is considered to be the access links (e.g. between a DSLAM and an ADSL modem) and not the network internal links.

Availability and failures. Mirroring requirements of practical systems, we assume devices to stay connected at least a few hours per day. Following the work by Dimakis [DGWR07] on network coding, we use values of availability and failure rate from the PlanetLab [Pla] and Microsoft PCs traces [BDET00]. To distinguish transient unavailability, which for some consumers is expected on a daily basis, from permanent failures, a timeout is introduced. Hence, a device is considered as failed if it leaves the network for more than 24 hours. In that case, all data stored by it is assumed to be lost.

The Mean Time To Failure (MTTF) in the Microsoft PCs and the PlanetLab scenarios are respectively 30 and 60 days. The device failures are then considered as independent, like in [RP06], and Poissonian with mean value given by the traces explained above. We consider a discrete time in the following and the probability to fail at any given time step is denoted as $\alpha = {}^{1}\!/_{\mathrm{MTTF}}$.

Repair process. When a failure is detected, neighbours of the failed device start a reconstruction process, to maintain desired redundancy

level. For each fragment stored at the failed disk, a random device from the neighbourhood is chosen to be the *reconstructor*. It is responsible for downloading necessary data from remaining fragments of the block, reconstructing and storing the fragment.

Redundancy schemes. Minimum Bandwidth Regenerating Codes, assumed in this paper, are very efficient due to not reconstructing the exact same lost fragment, but creating a new one instead, in the spirit of Network Coding. The reconstructor downloads, combines and stores small *subfragments* from d devices having other fragments of the repaired block. We call d the repair degree, $s \leq d \leq n$. Construction of the code requires some additional redundancy for each fragment. In other words L_r, the total amount of data transferred for a repair of a fragment, is greater than L_f by some overhead factor. This factor, the efficiency of the code, has been given for MBR in [DGWR07] as:

$$\delta_{MBR}(d) = \frac{2d}{2d - s + 1}.$$

The most bandwidth efficient case is clearly when $d = n - 1$. However, as we will show in following sections, it may be beneficial to set it to a lower value to give the reconstruction an additional degree of freedom.

The model presented in this work was also successfully applied to other redundancy schemes. Minimum Storage Regenerating Codes, also defined in [DGWR07], are more space efficient at the cost of additional transfer overhead. Reed-Solomon codes, more popular in practice, are reconstructed by recreating the input data and then coding again the lost fragment. In both cases the only difference for the model are different values of L_r. In practical systems, it may be interesting for RS-based systems to reconstruct at one device, but store the new fragment on some other one. This is especially true for *saddle*-based systems, where we wait until a few fragments of a block are lost, to repair them all at once. The model gives good results also for these more complicated cases. We omit them due to lack of space, and because this only brings slightly longer analysis with little new insight.

5.5 Preliminary: Impact of Disk Asymmetry

In this section we show that the efficiency of the system is affected by the imbalanced distribution of data among devices. Then, we estimate analytically this imbalance and its impact. After this preliminary study, the definition of the queuing model is given in Section 5.6.

Factor of efficiency. When a device fails, it is replaced by a new device with an empty disk. Since disks fill up during the system life, a recently replaced disk is empty, while an old disk contains many fragments. Hence, at any given time, disks with very heterogeneous number of fragments are present in the system. This heterogeneity has a strong impact on the reconstruction process: (1) when a disk dies, the number of block reconstructions that start depends on the number of fragments present in this disk. A lot of fragments are lost if the disk was full, but much less for a young disk. (2) during the repair, the devices have to send fragments to the reconstructors that rebuild the missing fragments. A device storing more fragments has to send a lot more fragments during this phase than a device with fewer fragments. Hence, such devices become a bottleneck of the system. On the other hand, the less loaded devices stay idle during some part of the time.

To estimate the impact of this imbalance on the system, we introduce a *factor of efficiency* ρ when the system is under load, defined as

$$\rho(load) = \frac{work}{\min(load, throughput)}$$

where *load* is the sum, over all devices, of the number of fragments in queues at the beginning of the time step; *throughput* is the maximum number of fragments that can be reconstructed by the whole system in one time step ($BW_{up} \cdot N \cdot \tau$, accounted in time steps of size τ); and *work* is the number of fragments that were effectively uploaded by the devices during the time step. When $\rho = 1$, the system works at its maximum speed, meaning that no device was idle while another one could not finish its work. Note that ρ greatly depends of the load. If the load is very large, compared to the bandwidth of the system, every device works at almost full capacity and the efficiency is close to one. Similarly, when the load is small, everybody has few fragments to upload and all the work is

Table 5.1: Summary of the main notations.

N	Total number of devices
s	Number of initial fragments of a block
r	Number of redundancy fragments of a block
n	Number of fragments of a block, $n = s + r$
d	Repair degree of the Regenerating Code, by default $d = n - 1$
δ_{MBR}	Efficiency of the Regenerating Codes
L_f	Size of a fragment, in bytes
L_r	Amount of data to repair a fragment
B	Total number of blocks in the system
F	Total number of fragments in the system
α	Peer failure rate ($\alpha = 1/MTTF$)
N_F	Number of devices with full disks
φ	Ratio of full disks, N_F/N
C	Capacity of a disk (number of fragments)
$\bar{D}$	Average number of fragments per disk
x	Disk size factor, $x = C/\overline{D}$
BW_{up}	Peer upload bandwidth (kbit/s)
v	Rate at which a disk fills up (fragments per cycle)
$T_{\max}$	Number of time steps to fill up a disk, $T_{\max} = C/v$

done. But, between these two cases, the imbalance between the devices causes a range of inefficiencies.

Estimation of the Imbalance The disk size has in fact a very strong effect on the general imbalance of the system. Figure 5.2 shows a histogram with the number of fragments in failed disks. These results are obtained by simulation of $N = 200$ devices with $MTTF = 60$ days (1440 hours). The amount of data per device is $14GB$. We set $s = r = 7$, and the fragment size $l_r = 2$ MB. Hence we have a total of $F = 7{\cdot}10^5$ fragments in the system. Then, the average number of fragments per device is $\bar{D} = 7000$.

We denote the disk capacity of devices as C (number of fragments). Hence, $x = C/\overline{D}$ is the *disk size factor*, i.e., how big is the disk when compared to the average amount of fragments per disk in the system.

When the factor $x = 3$ (that is, disk capacity $C = 21,000$ fragments), the imbalance is very large. At the opposite, when $x = 1.1$, the disk size is close to the average number of pieces per disk in the system. Hence, most of the disk fillings become full, 83% in our example. The disks that are not full (17%) have an almost uniform distribution. In the following, we give a method to calculate that imbalance analytically.

Disk age and disk size distributions can be precisely approximated for systems with a large number of blocks. The block fragments are reconstructed by devices that have free space in their disks (i.e., there are $N - N_F$ such devices, where N_F is the number of devices with full disks). Since these devices are chosen at random to reconstruct the blocks, at each time step the distribution of the rebuilt fragments among devices follows a multinomial distribution with parameters: the number of rebuilt fragments and $1/(N - N_F)$. As the multinomial distribution is very concentrated around its mean, the *filling up process can be approximated by an affine process of its age*, in which, at each time step, each disk gets the number of reconstructed fragments divided by the number of non-full devices, roughly

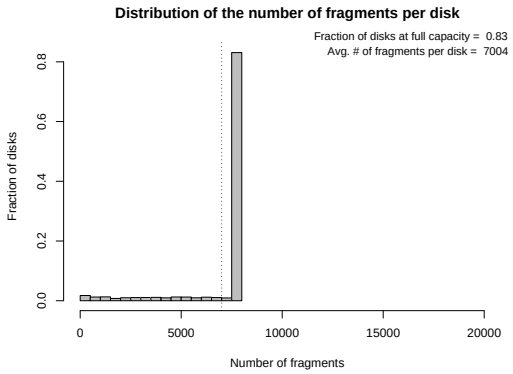
$$v = \frac{\alpha F}{N - N_F}$$

where α is the device failure rate. This filling process stops when the disk is full. That is after a number of time steps $T_{\max}$ such that $C = \alpha T_{\max} F/(N - N_F)$, where C is the device disk capacity (maximum number of fragments per disk). The number of fragments of a disk thus depends on the age of the disk.

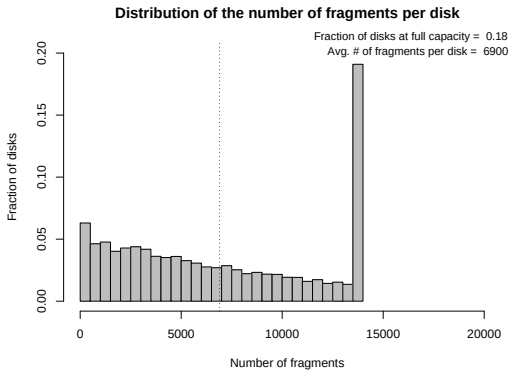
At each time step a disk has a probability α to experience a failure. Hence, the dead age of a disk follows a geometric law of parameter α. That is, $\Pr[\text{dead age} = T] = (1 - \alpha)^{T-1}\alpha$. Hence the distribution of the number of fragments in a disk follows a truncated geometric distribution, that is, for $1 \leq T < T_{\max}$

$$\begin{aligned} &Pr[D = vT] = (1 - \alpha)^{T-1}\alpha, \text{ and} \\ &Pr[D = C] = 1 - (1 - \alpha)^{T_{\max}}. \end{aligned} \tag{5.1}$$

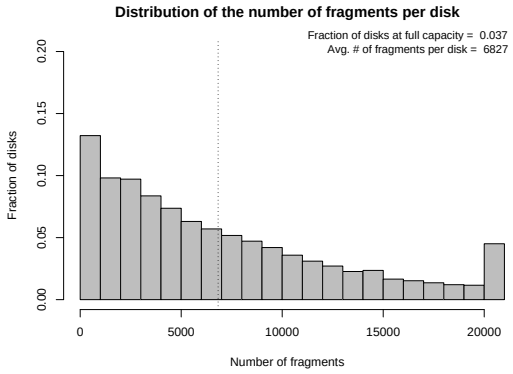
Note that here v, N_F, and $T_{\max}$ are unknown for the moment. The value of v depends on the number of full disks N_F, and of $T_{\max}$ depends directly of the filling rate v. To find the value of these variables, we use the fact that we know the expectation of the geometric distribution



(a) $x = 1.1$



(b) $x = 2.0$



(c) $x = 3.0$

Figure 5.2: Distribution of fragments per failed disk for different disk size factor x of 1.1, 2, and 3. The number of full disks in each scenario is respectively 83%, 18%, and 4%. (y-scales are different)

which is just the average number of fragments inside the system. This number is F/N (we neglect here the fragments that are in reconstruction, first order approximation for small α). Hence, we get $\mathbb{E}[D] = \overline{D} := F/N$. By definition, the expectation is also given by

$$\mathbb{E}[D] = \sum_{i=1}^{T_{\max}-1} vi(1-\alpha)^{i-1}\alpha + C(1-(1-\alpha)^{T_{\max}}).$$

To obtain $T_{\max}$, we now have to solve the equation:

$$\frac{1}{x} = \frac{1-\alpha-(1-\alpha)^{T_{\max}+1}}{\alpha T_{\max}},$$

obtained by identifying the two expressions for the expectation, by dividing by v, and because $C = x\overline{D}$. By solving that equation using the Maple software, we obtain that

$$T_{\max} = \frac{\alpha W(\frac{1}{\alpha}\ln(1-\alpha)x(1-\alpha)^{\frac{x+\alpha-x\alpha}{\alpha}}) - \ln(1-\alpha)x(1-\alpha)}{\ln(1-\alpha)\alpha},$$

where W is the Lambert W function. For example, when $MTTF = 1440$ hours ($\alpha = 1/1440$), the number of full disks and the number of time steps to fill up a disk are displayed in Table 5.2a. We verify that these values are very close to the ones obtained by simulation (Figure 5.2).

Effects of the Imbalance on the Bandwidth Efficiency Since some devices store less fragments, their load during the reconstruction process is also smaller. Thus, the overall bandwidth of the system is not fully utilized.

In a system using Regenerating Codes encoding, to repair a fragment, $d = n - 1$ small sub-fragments have to be sent to the device in charge of the reconstruction. Simulations show that the speed of the reconstruction is given by the time that the *most loaded device* takes to send the fragment. This time is in turn given by the number of fragments stored by this device. We get this number from the distribution of the number of fragments per device previously derived. For a majority of data blocks, *the most loaded device storing one of its fragment is in fact a full disk*. This claim is valid for most systems in practice, that is, for the parameters usually found in the literature.

x	N_F (in %)	$T_{\max}(hours)$
1.1	83	278
1.5	42	1257
2	20	2293
3	6	4060

(a)

x	1.1	1.5	2	3
φx	0.91	0.63	0.4	0.18
P_{full}	$1-10^{-14}$	$1-10^{-5}$	$1-10^{-3}$	0.92

(b)

Table 5.2: (a) The number of full disks and the number of time steps to fill up a disk, for $MTTF = 1440$ hours. (b) Fraction of full disks and the probability of a block to have at least one fragment on a full disk.

Indeed, recall that N_F denotes the number of full disks (and $\varphi = N_F/N$ the fraction of full disks). We compute the probability for a block that one of its fragment is on a full device (with $n-1$ available fragments when it is being repaired). Recall also that a full disk stores x times the average number of fragments per disk in the system. Then, the fraction of fragments stored on full disks is φx. The probability of a block to have at least one fragment on a full disk is then

$$P_{full} = 1 - (1 - x\varphi)^{n-1}.$$

For a system with $n = 14$ (the value of N_F for different values of x is given above), the probability for different disk capacities is displayed in Table 5.2b. We see that for most practical systems, each block has a fragment on a full disk. Hence, it is enough to consider the work done by the most loaded devices to obtain the reconstrution times. These devices have a load greater than the average load by a factor of $\frac{1}{x}$.

Factor of efficiency. An other way to phrase it: the factor of efficiency ρ of the system is approximately

$$\rho \approx \frac{1}{x}$$

where x is the fraction between disk capacity and the average number of fragments per disk.

More complex models for large disk capacities. We consider that in practice, as a measure of load balancing, the storage system sets a limit of disk capacity not too far from the average amount of data stored. A factor x between 1.1 and 3 seems reasonable. For systems with a very large disk capacity (for example $x = 10$), ρ has to be estimated in a different way. In that case a large number of blocks store no fragments on full disks. It is thus not enough to only consider the load of the full disks. This difficulty can be addressed by using a *multi-queue model*. The devices are partitioned into a number C of classes, depending on the number of data they store. The model has one queue per class. When a disk fails, we estimate the number of fragments that each class has to upload, that is how much work they do, and in this way derive the factor of efficiency ρ. The analysis of this model is beyond the scope of our study.

5.6 The Queueing Model

We introduce here a *Markovian Model* that allows us to estimate the reconstruction time under bandwidth constraints. The model makes an important assumption:

1. *The limiting resource is always the upload bandwidth.*

Assumption 1 is reasonable as download and upload bandwidths are strongly asymmetric in common installations. Using this assumption, we model the storage system with a *queue storing the upload load of the global system*.

Model Definition

We model the storage system with a Markovian queuing model storing the upload needs of the global system. The model has one server, Poissonian batch arrivals and deterministic time service ($M^\beta/D/1$, where β is the batch size function). We use a discrete time model. The peers in charge of repairs process blocks in a FIFO order.

Chain States. The state of the chain at a time t is the current number of fragments in reconstruction, denoted by $Q(t)$.

Transitions. At each time step, the system reconstructs blocks as fast as its bandwidth allows it. The upload bandwidth of the system, $BW_{up}N$, is the limiting resource. Then, the *service* provided by the server is

$$\mu = \rho \frac{BW_{up} N \tau}{L_r},$$

which corresponds to the number of fragments that can be reconstructed at each time step τ. The factor ρ is the bandwidth efficiency as calculated in the previous section, and L_r is the number of bytes transferred to repair one fragment. Hence, the number of fragments repaired during a time step t is $\mu(t) = \min(\mu, Q(t))$.

The *arrival process* of the model is caused by peer failures. When a failure occurs, all the fragments stored in that device are lost. Hence, a large number of block repairs start at the same time. We model this with batch inputs (sometimes also called *bulk arrival* in the literature). The size of an arrival is given by the number of fragments that were stored on the disk. As explained in Section 5.5, it follows a truncated geometric distribution.

We define β as a random variable taking values $\beta \in \{0, v, 2v, \ldots, T_{max}v\}$, which represents the number of fragments inside a failed disk (see Equation (5.1) for the probability distribution function of β). Recall that v is the speed at which empty disks get filled, and that $T_{max} = C/v$ is the elapsed time to fill a disk. Further on, β/v is the elapsed time to have a disk with β fragments.

The arrival process of the model is Poissonian. A batch arrives during a time step with probability f, with $f \approx \alpha N$. For the simplicity of the exposition, we consider here that only one failure can happen during a time step (note that to ensure this, it is sufficient to choose a small enough time step). Formally, the transitions of the chain are, for $\forall i \geq \mu$,

$$\begin{array}{lcll} Q_i & \rightarrow & Q_{i-\mu} & \text{with prob. } 1-f \\ Q_i & \rightarrow & Q_{i-\mu+\beta}, \forall \beta & \text{with prob. } f(1-\alpha)^{\frac{\beta}{v}-1}\alpha \\ Q_i & \rightarrow & Q_{i-\mu+C} & \text{with prob. } f(1-(1-\alpha)^{T_{\max}}) \end{array}$$

When $0 \leq i < \mu$, the i blocks in the queue at the beginning of the time step are reconstructed at the end. Hence, we have transitions without

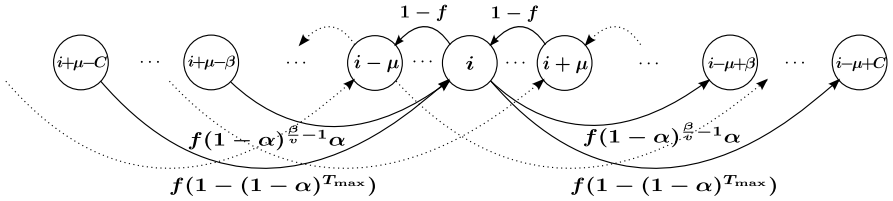


Figure 5.3: Transition around state i of the Markovian queuing model.

the term $i - \mu$:

$$\begin{array}{lcll} Q_i & \rightarrow & Q_0 & \text{with prob. } 1-f \\ Q_i & \rightarrow & Q_\beta, \forall\beta & \text{with prob. } f(1-\alpha)^{\frac{\beta}{v}-1}\alpha \\ Q_i & \rightarrow & Q_C & \text{with prob. } f(1-(1-\alpha)^{T_{\max}}) \end{array}$$

Figure 5.3 presents the transitions for a state i. The following table summarizes the notation introduced in this section.

$Q(t)$	Number of fragments to be repaired
f	Batch arrival rate, $f = \alpha N$
β	Number of fragments on a failed disk (i.e., batch size)
ρ	Factor of efficiency, $\rho \approx \frac{1}{x}$
μ	Service rate, $\mu = \rho BW_{up} N\tau / L_r$ (fragments per time step)

Analysis

Here, we give the expressions to estimate the values of two important system metrics: the distribution of the block reconstruction time and the probability of data loss. These expressions are derived from the stationary distribution of the Markovian model, as presented in the following.

A Normalized Model. The queuing model has a service of μ and an input process of average $f\beta$. To simplify the presentation of the analysis, we introduce then a *normalized model* with service of 1, hence an input of mean $\beta' = \beta/\mu$.

Stationary Distribution

We analyze here the stationary state of this normalized queuing model. As the chain is irreducible and aperiodic, it exists when the service rate is larger than the load. Let P be the probability generating function of the Markovian model, that is P is defined as:

$$P(z) = \sum_i P_i z^i,$$

where P_i is the probability that the system is in state i, that is, i fragments have to be repaired.

The system reconstructs one block per time step (unless of course, no block is in the queue). It is translated in the generating function language into a division by z. The effect of a peer failure is translated by a multiplication by the probability generating function of the input I, defined as

$$I(z) = \sum_{j=0}^{\infty} I_j z^j,$$

with I_j the probability that the batch is of size j. Hence, we obtain the functional equation

$$\left(\frac{P(z) - P_0}{z} + P_0\right) I(z) = P(z).$$

It gives

$$P(z) = \frac{(z-1)P_0}{\frac{z}{I(z)} - 1}.$$

As $P(1) = 1$, $I(z) - z$ admits 1 as a root and thus can be written as $I(z) - z = (z-1)Q(z)$. We have

$$P(z) = \frac{P_0 I(z)}{Q(z)}. \tag{5.2}$$

As we have seen in Section 5.5, the size of the input follows a truncated geometric distribution of parameter α. A batch is of size vj with probability $(1-\alpha)^{j-1}\alpha$, for $j \in [0, 1, ..., T_{\max}]$. It gives

$$I(z) = (1-f) + f \sum_{j=1}^{T_{\max}-1} (1-\alpha)^{j-1}\alpha z^{vj} + f(1-\alpha)^{T_{\max}-1} z^{vT_{\max}}.$$

It can be rewritten as

$$I(z) = 1 + \frac{f(z^v - 1)(z^{T_{\max}}(1-\alpha)^{T_{\max}} - 1)}{(1-\alpha)z^v - 1}.$$

We factorize $I(z) - z$ by $(z-1)$. We get

$$\begin{aligned} Q(z) &= I(z) - z \\ &= (z-1)(-1 + \tfrac{f(\sum_{j=1}^{v} z^i)(z^{vT_{\max}}(1-\alpha)^{T_{\max}} - 1)}{(1-\alpha)z^v - 1}). \end{aligned}$$

The value of P_0 is obtained by the normalization $\sum_{i=0}^{\infty} P_i = 1$ which implies $P(1) = 1$.

$$P_0 = \frac{Q(1)}{I(1)} = 1 - \frac{1}{\alpha}(fv((1-\alpha)^{T_{\max}} - 1)).$$

We now have an expression of the three terms of Equation 5.2 and we get a close form of the probability generating function $P(z)$.

Distribution of the Waiting Time

The distribution of the block reconstruction time is given by the stationary distribution P of the model calculated above. As we have Markovian (batch) arrivals, the probability for a batch to arrive when there are n blocks in the queue is exactly P_n (for the difference of distribution for an arriving customer and an outside observer, see for example [Coo81]). If there are Q fragments in the queue when a batch of size $\beta' = jv$ arrives, the arriving fragments have waiting times of $Q+1$, $Q+2$, $Q+\beta'$. We define the probability generating function J as

$$J(z) = \sum_{j=1}^{T_{\max}} \left((1-\alpha)^{j-1} \alpha \sum_{i=1}^{jv} z^i \right).$$

The probability generating function W of the waiting times then is just

$$W(z) = P(z)J(z).$$

The distribution of the waiting times can then be directly obtained from the generating function by extracting its coefficients

$$\Pr(W = k) = [z^k]W(z) = \left. \frac{d^k W(z)}{k!(dz)^k} \right|_{z=0}. \tag{5.3}$$

The first coefficients can be computed numerically and then a singularity analysis gives the asymptotic behavior, see for example [FS08]. Hence, the value of $\Pr(W = k)$ can be computed analytically. However, in the following, we also use another method and calculate them numerically by iterating the queuing model.

Number of Dead Blocks

The expected number of dead blocks is indirectly given by the model by computing the waiting time in the queue of a block that has to be reconstructed.

As a matter of fact, a block dies if it loses, before the end of the reconstruction, the $r - 1$ fragments of redundancy that it has left when the repair starts, plus an additional fragment. The probability for a device to still be alive after a period of time of θ time step is $(1 - \alpha)^\theta$, where α is the probability for a disk to die during a time step, that is

$$\alpha = \frac{\tau}{MTBF}.$$

Hence a good approximation of the probability $\Pr[die]$ to die during a reconstruction lasting a time θ is given by

$$\Pr[die|W = \theta] = \sum_{i=r}^{s+r} \binom{s+r}{i} (1 - (1 - \alpha)^\theta)^i ((1 - \alpha)^\theta)^{s+r-i}.$$

For practical systems, the ratio $\theta/MTTF$ is small as the probability to of data loss should be very low. Hence $\Pr[die]$ is well approximated by

$$\Pr[die|W = \theta] \approx \binom{s+r}{r} (1 - (1 - \alpha)^\theta)^r ((1 - \alpha)^\theta)^{s-1}.$$

From this and from the distribution of the waiting time, we get the probability to die during a reconstruction, P_D, with

$$P_D = \sum_{i=0}^{\infty} \Pr[\text{die}|W = i] \Pr[W = i].$$

The number of dead blocks during a time T, D_T, is then obtained by the number of reconstructions during T, R_T:

$$D_T = P_D R_T. \tag{5.4}$$

Bandwidth Usage

The bandwidth usage is directly given by the distribution of the number of reconstructions being processed by the system, which comes from the stationary distribution of the queuing model.

5.7 Results

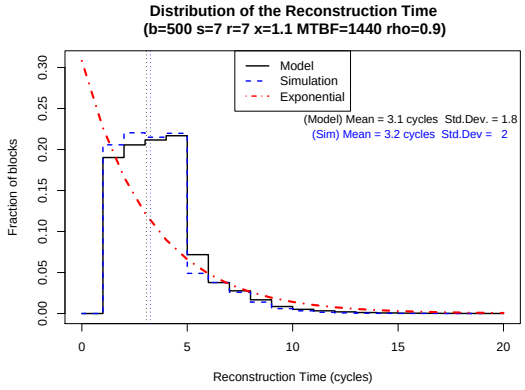
To validate our model, we compare its results with the ones produced by simulations, and test-bed experimentation. We use a custom cycle-based simulator. The simulator models the evolution of the states of blocks during time (number of available fragments and where they are stored) and the reconstructions being processed. When a disk failure occurs, the simulator updates the state of all blocks that have lost a fragment, and starts the reconstruction if necessary. The bandwidth is implemented as a queue for each device. The reconstructions are processed in FIFO order.

We study the distribution of the reconstruction time and compare it with the exponential distribution which is often used in the literature. We then discuss the cause of the data losses. Finally, we present two important practical implementation points: (1) when choosing the parameters of the Regenerating Code, it is important to give to the device in charge of the repair a choice between several peers to retrieve the data; (2) we show the strong impact of different scheduling options on the data loss rate.

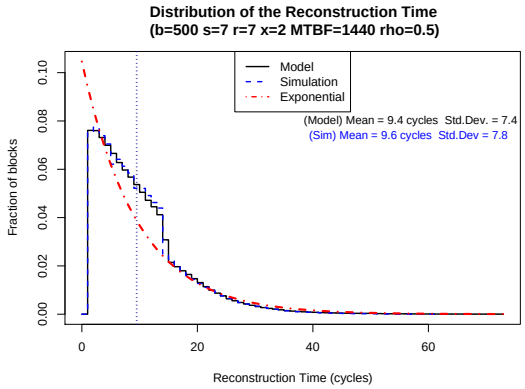
Distribution of Reconstruction Time

Figure 5.4 shows the distribution of the reconstruction time and the impact of the peer asymmetry on the reconstruction time for the following scenario: N = 100, s = 7, r = 7, L_r=2 MB, B = 50000, MTTF = 60 days, BW_{up} = 128 kpbs. All parameters are kept constant, except the disk size factor x (recall that x is the ratio of the maximum capacity over the average amount of data per device).

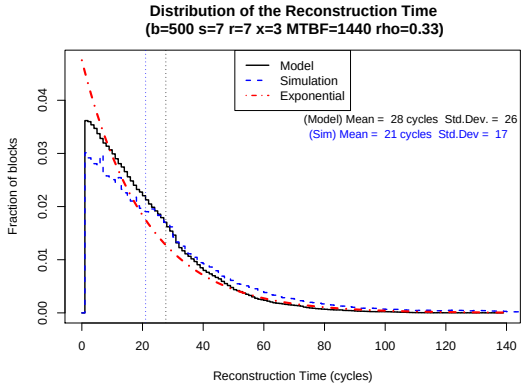
First, we see that the model (dark solid line) closely matches the simulations (blue dashed line). For example, when $x = 1.1$ (top plot), the curves are almost merged. The average reconstruction times are 3.1 cycles vs 3.2 for the model. We see that there is a small gap when $x = 3$.



(a) $x = 1.1$



(b) $x = 2.0$



(c) $x = 3.0$

Figure 5.4: Distribution of reconstruction time for different disk capacities x of 1.1, 2, and 3 times the average amount. The average reconstruction times of simulations are respectively 3.2, 9.6, and 21 hours (Note that some axis scales are different).

As a matter of fact, we saw in Section 5.5 that simulating the queue of the full disks is an approximation in this case, as only 92% of the blocks have a fragment on a full disk.

Second, we confirm the strong impact of the disk capacity. We see that for the three values of x considered, the shape of the reconstruction times are very different. When the disk capacity is close to the average number of fragments stored per disk (values of x close to 1), almost all disks store the same number of fragments (83% of full disks). Hence, each time there is a disk failure in the system, the reconstruction times span between 1 and C/μ, explaining the rectangle shape. The tail is explained by multiple failures happening when the queue is not empty. When x is larger, disks also are larger, explaining that it takes a longer time to reconstruct when there is a disk failure (the average reconstruction time raises from 3.2 to 9.6 and 21. when x goes from 1.1 to 2. and 3.). As the number of fragments per disk follows a truncated geometric distribution, we see the rectangle shape is replace by a trapezoidal shape explained by the large range of disk fillings.

Third, we compare the distributions obtained with the exponential distribution that is classically used in the literature. We see that the distributions are far from the exponential when $x = 1.1$ and $x = 2$, but get closer for $x = 3$. Hence, as we will confirm, the exponential distribution is only a good choice for some given sets of parameters. To finish, note that the tails of the distribution are close to exponential.

Figure 5.5 presents the distribution of a distributed storage system experiencing three different rates of failures: MTTF of 90, 180 and 360 days. We clearly see the evolution of the shape of the distribution due to the larger probability to experience failures when the peer queues are still loaded. The average reconstruction time increases from 5 hours when the MTTF is 360 days to 12 hours when the MTTF is 90 days.

We ran simulations for different sets of parameters. We present in Table 5.3 a small subset of these experiments.

From Where the Deads Come From?

In this section, we discuss in which circumstances the system has more chances to lose some data. First a preliminary remark: backup systems are conceived to experience basically no data loss. Thus, for realistic sets of parameters, it would be necessary to simulate the systems for a

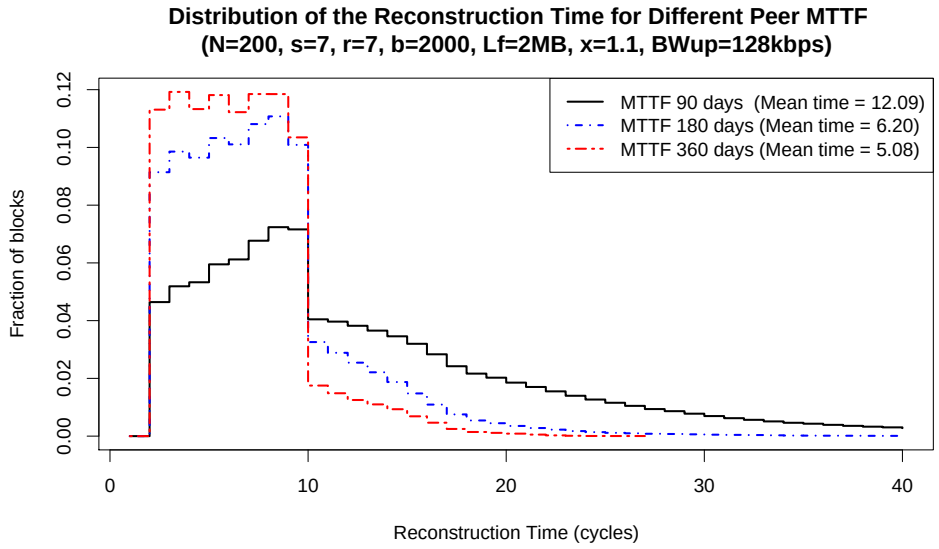


Figure 5.5: Distribution of reconstruction time for different MTBF. Different shapes for different values.

prohibitive time to see data losses in our simulations. We hence present here results for scenarios where the redundancy of the data is lowered ($r = 3$ and $r = 5$).

We plot in Figure 5.6 the cumulative number of dead blocks that the system experiences for different reconstruction times. We give this fraction in function of the time the block spent in the system before dying. For the queuing model, we derive the expected number of blocks that died at time T from the distribution of the reconstruction time. A block dies at time T if its reconstruction process lasts a time $\theta \geq T$ and that it loses r fragments during time T with at least one exactly at time T. This can be expressed as

$$N[\text{die at time } T] = \Pr[\text{die at time } T] \sum_{\theta \geq T} NP[W = \theta]$$

with

$$\Pr[\text{die at time } T] = \binom{s+r-1}{r-1}(1-(1-\alpha)^T)^r((1-\alpha)^T)^{s-1} - \binom{s+r-1}{r-1}(1-(1-\alpha)^{T-1})^r((1-\alpha)^T)^{s-1}.$$

Table 5.3: Reconstruction time T (in hours) for different system parameters

(a) Disk capacity c.

c	1.1	1.5	2.0	3.0
T_{sim}	3.26	5.50	9.63	21.12
T_{model}	3.06	5.34	9.41	21

(b) Peer Lifetime (MTBF).

$MTBF$	60	120	180	365
T_{sim}	3.26	2.90	2.75	2.65
T_{model}	2.68	2.60	2.49	2.46

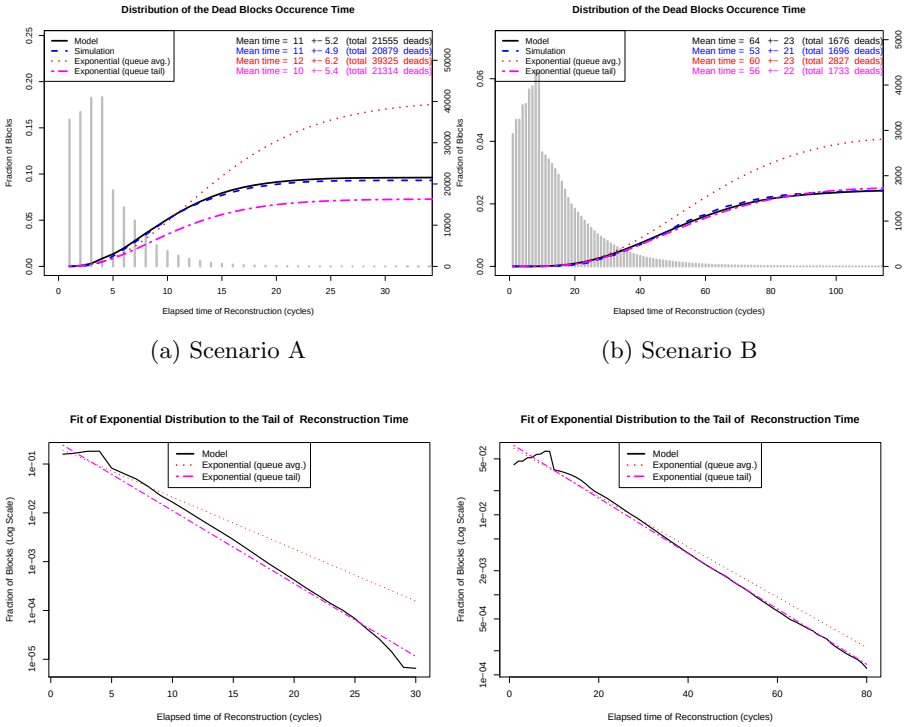
(c) Peer Upload Bandwidth (kbps).

$upBW$	64	128	256	512
T_{sim}	8.9	3.30	1.70	1.07
T_{model}	8.3	3.10	1.61	1.03

We give the distribution of the reconstruction times as a reference (vertical lines). The model (black solid line) and the simulation results (blue dashed line) are compared for two scenarios with different number of blocks: there is twice more data in Scenario B.

The first observation is that the queueuing models predict well the number of dead experienced in the simulation, for example, in the scenario A the values are 21,555 versus 20,879. The results for an exponential reconstruction time with the same mean value are also plotted (queue avg.). We see that this model is not close to the simulation for both scenarios (almost the double for Scenario A). We also test a second exponential model (queue tail): we choose it so that its tail is as close as possible to the tail than the queuing model (see Figures 5.6b and 5.6d). We see that it gives a perfect estimation of the dead for Scenario B, but not for Scenario A.

In fact, two different phenomena appear in these two scenarios. In Scenario B (higher redundancy), the *lost blocks are mainly coming from long reconstructions*, from 41 to 87 cycles (tail of the gray histogram). Hence, a good exponential model can be found by fitting the parameters to the tail of the queuing model. On the contrary, in Scenario A



(a) Scenario A

(b) Scenario B

(c) Scenario A (fitting)

(d) Scenario B (fitting)

Figure 5.6: (Top): Distribution of dead blocks reconstruction time for two different scenarios. Scenario A: $N = 200, s = 8, r = 3, b = 1000, MTTF = 60$ days. Scenario B: $N = 200, s = 8, r = 5, b = 2000, MTTF = 90$ days. (Bottom): Fitting of exponential distribution with the tail of queueing model (axis scales are different).

(lower redundancy), the *data loss comes from the majority of short reconstructions*, from 5.8 to 16.2 cycles (the right side of the rectangular shape). Hence, in Scenario A, having a good estimate of the tail of the distribution is not at all sufficient to be able to predict the failure rate of the system. It is necessary to have a good model of the complete distribution!

Discussing the Implementation of Regenerating Codes

As presented in Section 5.4, when the redundancy is added using regenerating codes, $n = s + r$ devices store a fragment of the block when s are

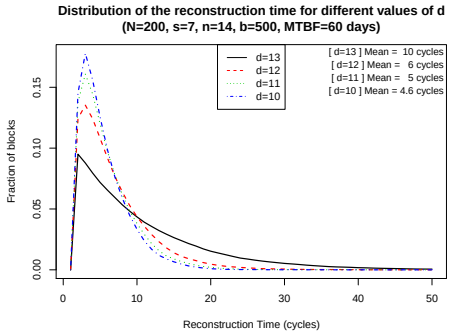


Figure 5.7: Distribution of reconstruction time for different values of degree d.

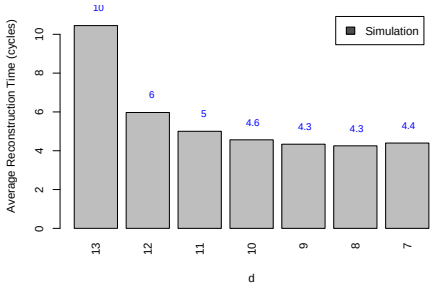


Figure 5.8: Average Reconstruction Time for different values of degree d. Smaller d implies more data transfers, but may mean smaller reconstruction times!

enough to retrieve the block. When a fragment is lost, $s \leq d \leq n-1$ peers are in charge of repairing the fragments. The larger d is, the smaller is the bandwidth needed for the repair. Figures 5.7 and 5.8 show the reconstruction time for different values of the degree d. We observe an interesting phenomena: at the opposite of the common intuition, the average reconstruction time decreases when the degree decreases: 10 cycles for $d = 13$, and only 6 cycles for $d = 12$. The bandwidth usage increases though (because the δ_{MBR} is higher when d is smaller). The explanation is that the decrease of the degree *introduces a degree of freedom* in the choice of the devices that send a sub-fragment to the device that will store the repaired fragment. Hence, the system is able to lower the load of the more loaded disks and to *balance more evenly the load between peers*.

In fact, we can estimate for which degree of freedom, the reconstruction time is minimum. It happens when the load of the full disks is the same as the load of the other disks. We define $\delta = n - 1 - d$ the allowed degree of freedom for the choice of which peers uploads the sub-fragments. The full disks store a proportion φx of the fragments of the system, with φ the fraction of full disks. We simply look at the how much work we *must* do on the full disks. The probability to have i fragments (among the $n-1$ fragments) on full disks is $\binom{n-1}{i}(\varphi x)^i(1-\varphi x)^{n-1-i}$. Those blocks sends $i - \delta$ units of work the full disks (whenever $i \geq \delta$).

So the load of the full disks is

$$\sum_{i=\delta}^{n-1}(i-\delta)\binom{n-1}{i}(\varphi x)^i(1-\varphi x)^{n-1-i}.$$

We presented here a cut argument for only two classes of peers (full disks and non full disks). This argument can be generalized to any number of peer classes.

When the load of the full disks becomes equal to the load of the other disks ($\sum_{i=\delta}^{n-1}(d-i+\delta)\binom{n-1}{i}(\varphi x)^i(1-\varphi x)^{n-1-i}$), it is no more useful to decrease d. We see that the average reconstruction time increases when d is too small, as the increased usage of bandwidth is no more compensated by a better balance of the load.

Note that this phenomena exists for other codes like Reed Solomon where the device in charge of the reconstruction has to retrieve s fragments among the $s+r-1$ remaining fragments.

Scheduling

As peers have a large number of repairs to carry out but very limited bandwidth, the question of which repairs to do first is crucial. In this section, we study three different scheduling choices: FIFO, RANDOM, and MOST-DAMAGED data block first.

The FIFO is the default scheduling in the simulator, as discussed in Section 5.4, the blocks are processed in the order of arrival. In the RANDOM scheduling, the simulator processes blocks in a random order (at each time step the list of blocks to be reconstructed is shuffled). In the MOST-DAMAGED scheduling the blocks are ordered by the level of redundancy (i.e., blocks with less fragments available come first). In case of tied values, then the FIFO order is assumed.

Figure 5.9 presents the reconstruction time of these three schedulings. All strategies give almost the same average reconstruction time, 4.40, 4.43, 4.43 respectively for FIFO, RANDOM and MOST-DAMAGED. We see that their distribution changes slightly. In the RANDOM order the shape has the form of a geometric distribution, with many blocks finishing the reconstruction "early". However, as depicted in Figure 5.9, the differences in the number of dead blocks are enormous. When using the RANDOM scheduling, the dead increases considerably, as expected.

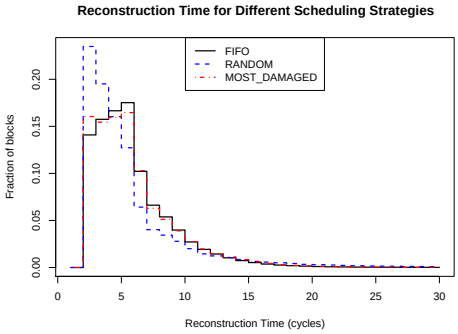


Figure 5.9: Reconstruction time for different scheduling strategies. The average reconstruction time is almost the same (4.4 cycles), but the distribution changes.

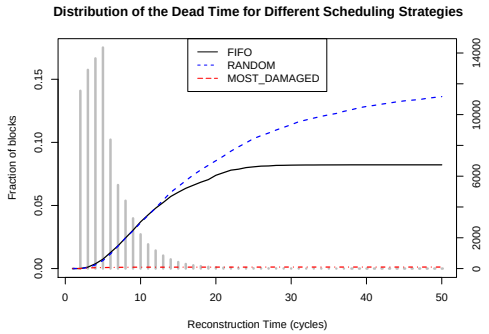


Figure 5.10: Cumulative number of dead blocks for different scheduling strategies. Processing the most damaged first is the best strategy.

MOST-DAMAGED has a reconstruction time very close to the others but the number of losses is much lower. Hence, this is the strategy of choice when implementing such systems.

5.8 Experimentation

Aiming at validating the simulation and the model results, we performed a batch of real experimentation using the GRID'5000 platform [Gri]. We used a prototype of storage system implemented by a private company (Ubistorage [ubi]).

Our goal is to validate the main behavior of the reconstruction time in a real environment with shared and constrained bandwidth, and measure how close they are to our results.

Storage System Description

In few words, the system is made of a storage layer (upper layer) built on top of the DHT layer (lower layer) running Pastry [RD01]. The lower layer is in charge of managing the logical topology: finding peers, routing, alerting of peer arrivals or departures. The upper layer is in charge of storing and monitoring the data.

Storing the data. The system uses Reed-Solomon erasure codes [LMS+97] to introduce redundancy. Each data block has a device responsible of monitoring it. This peer keeps a list of the devices storing a fragment of the block. The fragments of the blocks are stored locally on the PASTRY leafset of the peer in charge [LMSM09].

Monitoring the system. The storage system uses the information given by the lower level to discover device failures. In PASTRY, a peer checks periodically if the members of its leafset are still up and running. When the upper layer receives a message that a peer left, the peer in charge updates its block status.

Monitored metrics. The application monitors and keep statistics on the amount of data stored on its disks, the number of performed reconstructions along with their duration, the number of dead blocks that cannot be reconstructed. The upload and download bandwidth of devices can be adjusted.

The Grid'5000 Infrastructure

GRID'5000 is an infrastructure dedicated to the study of large scale parallel and distributed systems. It provides a highly reconfigurable, controllable and monitorable experimental platform to scientists. The platform contains 1582 machines accounting for 3184 processors and 5860 cores. The machines are geographically distributed on 9 different hosting sites in France (two additional sites in Luxemburg and Porto Alegre, Brazil are being added). These site are connected to RENATER Education and Research Network with a 10Gb/s link.

Results

There exist a lot of different storage systems with different parameters and different reconstruction processes. The goal of the paper is not to precisely tune a model to a specific one, but to provide a general analytical framework to be able to predict any storage system behavior. Hence, we are more interested here by the global behavior of the metrics than by their absolute values.

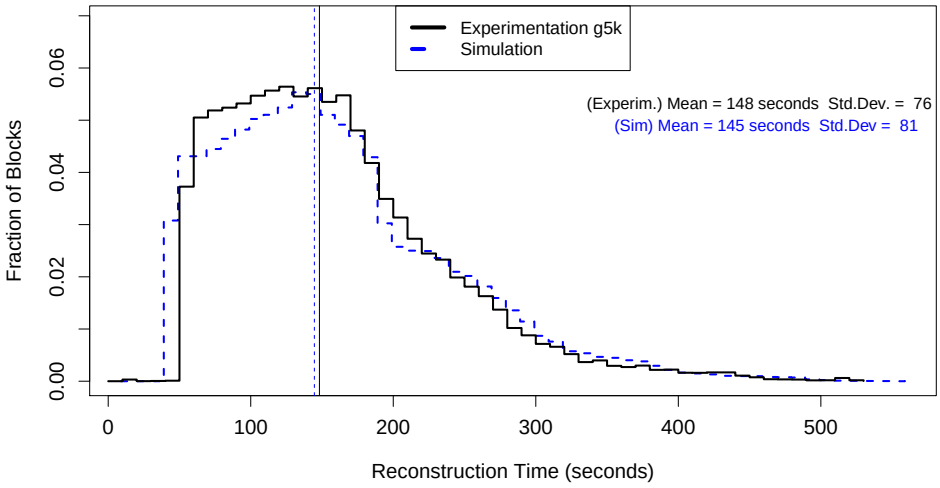


Figure 5.11: Distribution of reconstruction time on a experimentation with 64 nodes during 4 hours compared to simulations.

Studied Scenario. By using simulations we can easily evaluate several years of a system, however when doing experimentation this is not the case. We need to plan our experiments to last a few hours. Hence, we define an *acceleration factor*, as the ratio between experiment duration and the time of real system we want to imitate. Our goal is to check the bandwidth congestion in a real environment. Thus, we decided to shrink the disk size (e.g., from 10 gigabytes to 100 megabytes, a reduction of 100×), inducing a much smaller time to repair a failed disk. Then, the device failure rate is increased (from months to a few hours) to keep the ratio between disk failures and repair time proportional. The bandwidth limit value, however, is kept close to the one of a "real" system. The idea is to avoid inducing strange behaviors due to very small packets being transmitted in the network.

Figure 5.11 presents the distribution of the reconstruction times for two different experimentation involving 64 nodes on 2 different sites of Grid'5000. The amount of data per node is 100 MB (disk capacity 120MB), the upload bandwidth 128 KBps, $s = 4$, $r = 4$, $L_F = 128$ KB. We confirm that the simulator gives results very close to the one

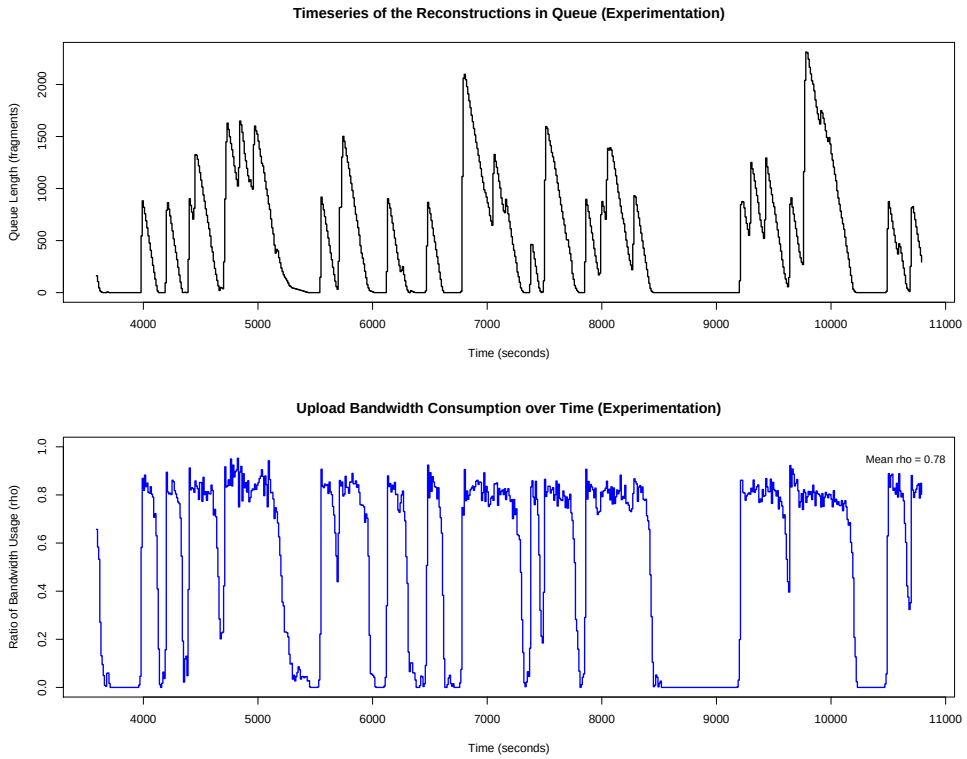


Figure 5.12: Timeseries of the queue size during time (top) and the upload bandwidth ratio (bottom).

obtained by experimentation. The average value of reconstruction time differs from some seconds.

Moreover, to have an intuition of the system dynamics over time, in Figure 5.12 we present a timeseries of the number of blocks in the queues (top plot) and the total upload bandwidth consumption (bottom plot). We note that the rate of reconstructions (the descending lines on the top plot) follows an almost linear shape. Comforting our claim that a deterministic processing time of blocks could be assumed. In these experiments the disk size factor is $x = 1.2$, which gives a theoretical efficiency of 0.83. We can observe that in practice, the factor of bandwidth utilization, ρ, is very close to this value (value of $\rho = 0.78$ in the bottom plot).

5.9 Conclusion

In this paper, we propose and analyze a new Markovian analytical model to model the repair process of distributed storage systems. This model takes into account the correlation between data repairs that compete for the same bandwidth. We bring to light the impact of peer heterogeneity on the system efficiency. The model is validated by simulation and by real experiments on the Grid'5000 platform.

We show that the exponential distribution classically taken to model the reconstruction time is valid for certain sets of parameters, but that different shapes of distribution appear for other parameters. We show that it is not enough to be able to estimate the tail of the repair time distribution to obtain a good estimate of the system loss rate.

The results provided are for systems using Regenerating Codes that are the best codes known for bandwidth efficiency, but the model is general and can be adapted to other codes. We exhibit an interesting phenomena to keep in mind when choosing the code parameter: it is useful to keep a degree of freedom on the choice of the users participating in the repair process so that loaded or deficient users do not slow down the repair process, even if it means less efficient codes.

In addition, we confirm the strong impact of scheduling on the system loss rate.

5.10 Bibliography

[ADN07] Sara Alouf, Abdulhalim Dandoush, and Philippe Nain. Performance analysis of peer-to-peer storage systems. *Proceedings of the 20th International Teletraffic Congress (ITC)*, LNCS 4516:642–653, 2007.

[BDET00] William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. *ACM SIGMETRICS Perf. Eval. Review*, 28:34–43, 2000.

[BTcC+04] Ranjita Bhagwan, Kiran Tati, Yu chung Cheng, Stefan Savage, and Geoffrey M. Voelker. Total recall: System sup-

port for automated availability management. In *Proc. of the USENIX NSDI*, pages 337–350, 2004.

[CDH+06] Byung-Gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon, M. Frans Kaashoek, John Kubiatowicz, and Robert Morris. Efficient replica maintenance for distributed storage systems. In *Proc. of USENIX NSDI*, pages 45–58, 2006.

[Coo81] Robert B. Cooper. *Introduction to Queuing Theory*. North Holland New York, 1981.

[DA06] Anwitaman Datta and Karl Aberer. Internet-scale storage systems under churn – a study of the steady-state using markov models. In *Procedings of the IEEE Intl. Conf. on Peer-to-Peer Computing (P2P)*, pages 133–144, 2006.

[DAN09] A. Dandoush, S. Alouf, and P. Nain. Simulation analysis of download and recovery processes in P2P storage systems. In *Proc. of the Intl. Teletraffic Congress (ITC)*, pages 1–8, France, 2009.

[DGWR07] A.G. Dimakis, P.B. Godfrey, M.J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Proc. of IEEE INFOCOM*, pages 2000–2008, May 2007.

[FLP+10] Daniel Ford, François Labelle, Florentina I Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. Availability in globally distributed storage systems. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pages 1–7, 2010.

[FS08] P. Flajolet and R. Sedgewick. *Analytic combinatorics*. Cambridge University Press, 2008.

[Gri] Grid5000Platform. https://www.grid5000.fr/.

[KBC+00] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells,

et al. OceanStore: an architecture for global-scale persistent storage. *ACM SIGARCH Computer Architecture News*, 28(5):190–201, 2000.

[LMS+97] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proceedings of the 29th annual ACM symposium on Theory of computing*, pages 150–159, 1997.

[LMSM09] S. Legtchenko, S. Monnet, P. Sens, and G. Muller. Churn-resilient replication strategy for peer-to-peer distributed hash-tables. In *Proceedings of SSS*, volume LNCS 5873, pages 485–499, 2009.

[PBS07] Fabio Picconi, Bruno Baynat, and Pierre Sens. Predicting durability in dhts using markov chains. In *Proceedings of the 2nd Intl. Conference on Digital Information Management (ICDIM)*, volume 2, pages 532–538, Oct. 2007.

[Pla] Planetlab. URL: `http://www.planet-lab.org/`.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. of IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware)*, volume LNCS 2218, pages 329–350, 2001.

[RP06] Sriram Ramabhadran and Joseph Pasquale. Analysis of long-running replicated systems. In *Proc. of IEEE INFOCOM*, pages 1–9, Spain, 2006.

[ubi] Ubistorage. http://www.ubistorage.com/.

[VIH12] Vinodh Venkatesan, Ilias Iliadis, and Robert Haas. Reliability of data storage systems under network rebuild bandwidth constraints. In *2012 IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 189–197, 2012.

[VLM+09] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulié, Christophe Diot, and Pablo Rodriguez. Greening the internet

with nano data centers. In *ACM CoNEXT '09*, pages 37–48, Rome, Italy, 2009.

[WK02] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Revised papers from the 1st Intl. Workshop on Peer-to-Peer Systems (IPTPS)*, volume LNCS 2429, pages 328–337, 2002.

CHAPTER 6

Conclusions and perspectives

The contributions of this thesis provide tools to assess a number of potential solutions for making the future Internet more efficient. There are many approaches to reducing the redundancy in the traffic, some of them well established. We looked into the ones which are not yet the standard, but are considered as possible future directions. From a high level perspective, these are all simple ideas: store a copy close to your users for further reuse, make the users share among themselves and store their own data as close as possible. Complexity arises when one tries to implement them. However, we do not look deep into details of particular implementations. We try to abstract over the complexity, to look into the potential of the ideas themselves, keeping to a realistic setting. Instead of proposing systems and tweaking their efficiency, we evaluate impact of systems with a given efficiency. Ultimately the questions we are trying to answer take the form of *what would really be the potential benefit of putting any system of some known properties into realistic conditions?*

Two models are devoted to estimating the potential energy savings thanks to introductions of caches. First, we studied energy optimization in network provisioning with in-network caches. We found that basing on realistic network and power models, but with some optimistic simplifying assumptions, up to 11% of energy can be saved by introducing the caches. The most propitious enhancements to the model could be studying the actual dynamics encountered by caching algorithms, as well as relaxing the regularity of the network model.

Then, we looked into energy-aware management of an already deployed core network. This has shown that, over a number of realistic network instances, we can save over 20% of energy exploiting daily traffic variations. A palpable way to enhance this result would be merging this model with the aforementioned one.

Both studies treat cache as a black box, which performs its work according to some static properties. Real-life caches are more complicated. Their performance is determined by the interplay of replacement algorithms and the stream of requests served, which are intrinsically random. Exploring how temporal and geographical distributions of requests affect the optimal cache deployment and operation is an interesting research direction. Furthermore, we had to speculate about devices that are not available yet, as well as about network structure and traffic, which are trade secrets of the operators. The best way to fully benefit of this work is to apply it from inside an operator, using specific and accurate data.

Both studies hint that substantial savings are possible. Even if the money saved by reducing energy consumption do not outweigh deployment cost in a short term, bandwidth savings themselves are already a good incentive for augmenting networks with caches. However, the advantages of in-network caching may be overshadowed by alternatives and the whole picture may change with next generations of hardware not yet revealed to the public. Thus, until a decisive trend arises in the industry, they remain an active research topic.

The next two studies are concentrating on highly distributed systems, which can be either user ran peer-to-peer or operator controlled (e.g. nano data centers). First, we analyzed a live streaming network with a tree structure. We proved an upper time limit for repairing the tree, after an arbitrary failure, by a simple algorithm. We also found, by means of simulation, that on average such a repair takes very short time. The obvious continuations of this work are formal study of the average behaviour and modelling multiple concurrent trees structure.

Second, we looked into data survivability in distributed backups system. We found that back-of-envelope calculations may overestimate it by orders of magnitude, comparing to a model carefully following data and workload distributions. The general framework presented in this study can be improved by adjusting the model to closely match a target system.

Currently, this kind of systems do not play a major role in the industry. Data backup relies heavily on trust. Delegating it to unknown stranger in a peer-to-peer network is deemed too high risk by many users. Centralized solutions, especially some recent backup-oriented offerings[1] are already cost-effective and ensure the trust by contracts. On the other hand, peer-to-peer video streaming will probably gain importance. Raise of video streaming traffic leads to network congestions. This in turn leads to tensions between content providers and network operators. These have already escalated up to involving law enforcement[2]. This raises incentives for p2p video streaming and coincides with new means. WebRTC[3] is being implemented in the major web browsers. It allows real-time browser-to-browser communication. Browser support removes a major issue, which always was the need to install additional software, making participation in a peer-to-peer streaming network as easy as clicking a YouTube link today. All things considered, design and implementation of peer-to-peer streaming networks may be a very interesting perspective in the coming years.

To answer questions posed in this thesis, I have learnt a number of useful techniques. Some of them are theoretical tools, which allow me to approach algorithmic challenges in a structured way. Other are more empirical, like simulations and experiments. One trick I am particularly satisfied with is using simple but revealing implementations of abstract systems for a quick peek into their properties. This has allowed us to weed out a number false hypotheses early, in our more theoretical forays. Another lesson is the importance of changing directions as more promising ones are appearing. This happened when we explored unstructured streaming networks, to finally concentrate on a structured one.

[1] For example `http://aws.amazon.com/glacier/`

[2] `http://www.reuters.com/article/2013/07/11/eu-telecoms-idUSL6N0FH1OL20130711`

[3] `http://www.webrtc.org/reference/architecture`

APPENDIX A

Weighted Improper Colouring

This appendix presents a study that is not concerned by reducing redundancy in network traffic. Instead, it is motivated by frequency assignment in satellite networks, what places in it the link layer. While energy saving in network was not an original motivation of this work, notice that reducing radio interference does significantly reduce power consumption (and therefore increase battery life) of mobile devices.

A.1 Publication

The remainder of this chapter corresponds to *Weighted Improper Colouring* by J. Araujo, J-C. Bermond, F. Giroire, F. Havet, D. Mazauric and R. Modrzejewski which was published in the Journal of Discrete Algorithms volume 16, which is an extended version of the work of same title and authors published in the proceedings of 22nd International Workshop on Combinatorial Algorithms.

A.2 Introduction

Let $G = (V, E)$ be a graph. A *k-colouring* of G is a function $c : V \to \{1, \ldots, k\}$. The colouring c is *proper* if $uv \in E$ implies $c(u) \neq c(v)$. The *chromatic number* of G, denoted by $\chi(G)$, is the minimum integer k such that G admits a proper k-colouring. The goal of the VERTEX

Colouring problem is to determine $\chi(G)$ for a given graph G. It is a well-known NP-hard problem [Kar72].

A k-colouring c is *l-improper* if $|\{v \in N(u) \mid c(v) = c(u)\}| \leq l$, for all $u \in V$ (as usual in the literature, $N(u)$ stands for the set $\{v \mid uv \in E(G)\}$). Given a non-negative integer l, the *l-improper chromatic number* of a graph G, denoted by $\chi_l(G)$, is the minimum integer k such that G admits an l-improper k-colouring. Given a graph G and an integer l, the Improper Colouring problem consists in determining $\chi_l(G)$ and is also NP-hard [Woo90, CHS09]. Indeed, if $l = 0$, observe that $\chi_0(G) = \chi(G)$. Consequently, Vertex Colouring is a particular case of Improper Colouring.

In this work we define and study a new variation of the Improper Colouring problem for edge-weighted graphs. An edge-weighted graph is a pair (G, w) where $G = (V, E)$ is a graph and $w : E \to \mathbb{R}_+^*$. Given an edge-weighted graph (G, w) and a colouring c of G, the *interference* of a vertex u in this colouring is defined by

$$I_u(G, w, c) = \sum_{\{v \in N(u) | c(v) = c(u)\}} w(u, v).$$

For any non-negative real number t, called *threshold*, we say that c is a *weighted t-improper k-colouring* of (G, w) if c is a k-colouring of G such that $I_u(G, w, c) \leq t$, for all $u \in V$.

Given a threshold $t \in \mathbb{R}_+^*$, the minimum integer k such that the graph G admits a weighted t-improper k-colouring is the *weighted t-improper chromatic number* of (G, w), denoted by $\chi_t(G, w)$. Given an edge-weighted graph (G, w) and a threshold $t \in \mathbb{R}_+^*$, determining $\chi_t(G, w)$ is the goal of the Weighted Improper Colouring problem. Note that if $t = 0$ then $\chi_0(G, w) = \chi(G)$, and if $w(e) = 1$ for all $e \in E$, then $\chi_l(G, w) = \chi_l(G)$ for any positive integer l. Therefore, the Weighted Improper Colouring problem is clearly NP-hard since it generalises Vertex Colouring and Improper Colouring.

On the other hand, given a positive integer k, we define the *minimum k-threshold* of (G, w), denoted by $T_k(G, w)$ as the minimum real t such that (G, w) admits a weighted t-improper k-colouring. Then, for a given edge-weighted graph (G, w) and a positive integer k, the Threshold Improper Colouring problem consists in determining $T_k(G, w)$. The Threshold Improper colouring problem is also NP-hard. This

fact follows from the observation that determining whether $\chi_l(G) \leq k$ is NP-complete, for every $l \geq 2$ and $k \geq 2$ [CCW86, CGJ95, CHS09]. Consequently, in particular, it is a NP-complete problem to decide whether a graph G admits a weighted t-improper 2-colouring when all the weights of the edges of G are equal to one, for every $t \geq 2$.

Motivation

Our initial motivation to these problems was the design of satellite antennas for multi-spot MFTDMA satellites [AAG+05]. In this technology, satellites transmit signals to areas on the ground called *spots*. These spots form a grid-like structure which is modelled by an hexagonal cell graph. To each spot is assigned a radio channel or colour. Spots are interfering with other spots having the same channel and a spot can use a colour only if the interference level does not exceed a given threshold t. The level of interference between two spots depends on their distance. The authors of [AAG+05] introduced a factor of mitigation γ and the interference of remote spots are reduced by a factor $1 - \gamma$. When the interference level is too low, the nodes are considered to not interfere anymore. Considering such types of interference, where nodes at distance at most i interfere, leads to the study of the i-th power of the graph modelling the network and a case of special interest is the power of grid graphs (see Section A.4).

Related Work

Our problems are particular cases of the Frequency Assignment problem (FAP). FAP has several variations that were already studied in the literature (see [AvHK+07] for a survey). In most of these variations, the main constraint to be satisfied is that if two vertices (mobile phones, antennas, spots, etc.) are close, then the difference between the frequencies that are assigned to them must be greater than some function which usually depends on their distance.

There is a strong relationship between most of these variations and the $L(p_1, \ldots, p_d)$-labelling problem [Yeh06]. In this problem, the goal is to find a colouring of the vertices of a given graph G, in such a way that the difference between the colours assigned to vertices at distance i is at least p_i, for every $i = 1, \ldots, d$.

In some other variants, for each non-satisfied interference constraint a penalty must be paid. In particular, the goal of the Minimum Interference Frequency Assignment problem (MI-FAP) is to minimise the total penalties that must be paid, when the number of frequencies to be assigned is given. This problem can also be studied for only *co-channel interference*, in which the penalties are applied only if the two vertices have the same frequency. However, MI-FAP under these constraints does not correspond to Weighted Improper Colouring, because we consider the co-channel interference, i.e. penalties, just between each vertex and its neighbourhood.

The two closest related works we found in the literature are [MS03] and [FLM$^+$00]. However, they both apply penalties over co-channel interference, but also to the *adjacent channel interference*, i.e. when the colours of adjacent vertices differ by one unit. Moreover, their results are not similar to ours. In [MS03], they propose an enumerative algorithm for the problem, while in [FLM$^+$00] a Branch-and-Cut method is proposed and applied over some instances.

Results

In this article, we study both parameters $\chi_t(G, w)$ and $T_k(G, w)$. We first present general bounds; in particular we show a generalisation of Lovász's Theorem for $\chi_t(G, w)$. We after show how to transform an instance of Threshold Improper colouring into an equivalent one where the weights are either one or M, for a sufficiently large M.

Motivated by the original application, we then study a special interference model on various grids (square, triangular, hexagonal) where a node produces a noise of intensity 1 for its neighbours and a noise of intensity 1/2 for the nodes that are at distance two. We derive the weighted t-improper chromatic number for all possible values of t.

Finally, we propose a heuristic and a Branch-and-Bound algorithm to solve Threshold Improper colouring for general graphs. We compare them to an integer linear programming formulation on random cell-like graphs, namely Voronoi diagrams of random points of the plan. These graphs are classically used in the literature to model telecommunication networks [BKLZ97, GK00, HAB$^+$09].

A.3 General Results

In this section, we present some results for Weighted Improper colouring and Threshold Improper colouring for general graphs and general interference models.

Upper bounds

Let (G, w) be an edge-weighted graph with positive real weights given by $w : E(G) \to \mathbb{Q}_+^*$. For any vertex $v \in V(G)$, its *weighted degree* is $d_w(v) = \sum_{u \in N(v)} w(u, v)$. The *maximum weighted degree* of G is $\Delta(G, w) = \max_{v \in V} d_w(v)$.

Given a k-colouring $c : V \to \{1, \ldots, k\}$ of G, we define, for every vertex $v \in V(G)$ and colour $i = 1, \ldots, k$, $d^i_{w,c}(v) = \sum_{\{u \in N(v) | c(u) = i\}}(u, v)$. Note that $d^{c(v)}_{w,c}(v) = I_v(G, w, c)$. We say that a k-colouring c of G is *w-balanced* if c satisfies the following property:

$$\text{For any vertex } v \in V(G),\ \ I_v(G, w, c) \leq d^j_{w,c}(v),\ \ \text{for every } j = 1, \ldots, k.$$

We denote by $\gcd(w)$ the greatest common divisor of the weights of w (observe that $\gcd(w) > 0$ because we just consider positive weights). We use here the generalisation of the gcd to non-integer numbers (e.g. in $\mathbb{Q}$) where a number x is said to divide a number y if the fraction y/x is an integer. The important property of $\gcd(w)$ is that the difference between two interferences is a multiple of $\gcd(w)$; in particular, if for two vertices v and u, $d^i_{w,c}(v) > d^j_{w,c}(u)$, then $d^i_{w,c}(v) \geq d^j_{w,c}(u) + \gcd(w)$.

If t is not a multiple of the $\gcd(w)$, that is, there exists an integer $a \in \mathbb{Z}$ such that $a \gcd(w) < t < (a + 1)\gcd(w)$, then $\chi^w_t(G) = \chi^w_{a \gcd(w)}(G)$.

Proposition 1. *Let (G, w) be an edge-weighted graph. For any $k \geq 2$, there exists a w-balanced k-colouring of G.*

Proof. Let us colour $G = (V, E)$ arbitrarily with k colours and then repeat the following procedure: if there exists a vertex v coloured i and a colour j such that $d^i_{w,c}(v) > d^j_{w,c}(v)$, then recolour v with colour j. Observe that this procedure neither increases (we just move a vertex from one colour to another) nor decreases (a vertex without neighbour on its colour is never moved) the number of colours within this process. Let W be the sum of the weights of the edges having the same colour

in their end-vertices. In this transformation, W has increased by $d^j_{w,c}(v)$ (edges incident to v that previously had colour j in its endpoint opposite to v), but decreased by $d^i_{w,c}(v)$ (edges that previously had colour i in both of their end-vertices). So, W has decreased by $d^i_{w,c}(v) - d^j_{w,c}(v) \geq \gcd(w)$. As $W \leq |E| \max_{e \in E} w(e)$ is finite, this procedure finishes and produces a w-balanced k-colouring of G. □

The existence of a w-balanced colouring gives easily some upper bounds on the weighted t-improper chromatic number and the minimum k-threshold of an edge-weighted graph (G, w). It is a folklore result that $\chi(G) \leq \Delta(G) + 1$, for any graph G. Lovász [Lov66] extended this result for Improper Colouring problem using w-balanced colouring. He proved that $\chi_l(G) \leq \lceil \frac{\Delta(G)+1}{l+1} \rceil$. In what follows, we extend this result to weighted improper colouring.

Theorem 5. *Let (G, w) be an edge-weighted graph with $w : E(G) \to \mathbb{Q}^*_+$, and t a multiple of $\gcd(w)$. Then*

$$\chi_t(G, w) \leq \left\lceil \frac{\Delta(G, w) + \gcd(w)}{t + \gcd(w)} \right\rceil.$$

Proof. If t, ω, and G are such that $\chi_t(G, \omega) = 1$, then the inequality is trivially satisfied. Thus, consider that $\chi_t(G, \omega) > 1$.

Observe that, in any w-balanced k-colouring c of a graph G, the following holds:

$$d_w(v) = \sum_{u \in N(v)} w(u, v) \geq k d^{c(v)}_{w,c}(v). \tag{A.1}$$

Let $k^* = \left\lceil \frac{\Delta(G,w)+\gcd(w)}{t+\gcd(w)} \right\rceil \geq 2$ and c^* be a w-balanced k^*-colouring of G. We claim that c^* is a weighted t-improper k^*-colouring of (G, w).

By contradiction, suppose that there is a vertex v in G such that $c^*(v) = i$ and that $d^i_{w,c}(v) > t$. Since c^* is w-balanced, $d^j_{w,c}(v) > t$, for all $j = 1, \ldots, k^*$. By the definition of $\gcd(w)$ and as t is a multiple of $\gcd(w)$, it leads to $d^j_{w,c}(v) \geq t + \gcd(w)$ for all $j = 1, \ldots, k^*$. Combining this inequality with Inequality (A.1), we obtain:

$$\Delta(G, w) \geq d_w(v) \geq k^*(t + \gcd(w)),$$

giving

$$\Delta(G, w) \geq \Delta(G, w) + \gcd(w),$$

a contradiction. The result follows.

□

Note that when all weights are unit, we obtain the bound for the improper colouring derived in [Lov66]. Brooks [Bro41] proved that for a connected graph G, $\chi(G) = \Delta(G) + 1$ if, and only if, G is complete or an odd cycle. One could wonder for which edge-weighted graphs the bound we provided in Theorem 5 is tight. However, Correa *et al.* [CHS09] already showed that it is NP-complete to determine if the improper chromatic number of a graph G attains the upper bound of Lovász, which is a particular case of Weighted Improper colouring, i.e. of the bound of Theorem 5.

We now show that w-balanced colourings also yield upper bounds for the minimum k-threshold of an edge-weighted graph (G, w). When $k = 1$, then all the vertices must have the same colour, and $T_1(G, w) = \Delta(G, w)$. This may be generalised as follows, using w-balanced colourings.

Theorem 6. *Let (G, w) be an edge-weighted graph with $w : E(G) \to \mathbb{R}_+^*$, and let k be a positive integer. Then*

$$T_k(G, w) \leq \frac{\Delta(G, w)}{k}.$$

Proof. Let c be a w-balanced k-colouring of G. Then, for every vertex $v \in V(G)$:

$$kT_k(G, w) \leq kd_{w,c}^{c(v)}(v) \leq d_w(v) = \sum_{u \in N(v)} w(u, v) \leq \Delta(G, w)$$

□

Because $T_1(G, w) = \Delta(G, w)$, Theorem 6 may be restated as $kT_k(G, w) \leq \ldots \leq T_1(G, w)$. This inequality may be generalised as follows.

Theorem 7. *Let (G, w) be an edge-weighted graph with $w : E(G) \to \mathbb{R}_+$, and let k and p be two positive integers. Then*

$$T_{kp}(G, w) \leq \frac{T_p(G, w)}{k}.$$

Proof. Set $t = T_p(G, w)$. Let c be a t-improper p-colouring of (G, w). For $i = 1, \ldots, p$, let G_i be the subgraph of G induced by the vertices coloured i by c. By definition of improper colouring $\Delta(G_i, w) \leq t$ for all $1 \leq i \leq p$. By Theorem 6, each (G_i, w) admits a t/k-improper k-colouring c_i with colours $\{(i-1)k+1, \ldots, ik\}$. The union of the c_i's is then a t/k-improper kp-colouring of (G, w). □

Theorem 7 and its proof suggest that to find a kp-colouring with small impropriety, it may be convenient to first find a p-colouring with small impropriety and then to refine it. In addition, such a strategy allows to adapt dynamically the refinement. In the above proof, the vertex set of each part G_i is again partitioned into k parts. However, sometimes, we shall get a better kp-colouring by partitioning each G_i into a number of k_i parts, with $\sum_{i=1}^{p} k_i = kp$. Doing so, we obtain a T-improper kp-colouring of (G, w), where $T = \max\{\frac{\Delta(G_i,w)}{k_i}, 1 \leq i \leq p\}$.

One can also find an upper bound on the minimum k-threshold by considering first the $k-1$ edges of largest weight around each vertex. Let (G, w) be an edge-weighted graph, and let $v_1, \ldots, v_n$ be an ordering of the vertices of G. The edges of G may be ordered in increasing order of their weight. Furthermore, to make sure that the edges incident to any particular vertex are totally ordered, we break ties according to the label of the second vertex. Formally, we say that $v_iv_j \leq_w v_iv_{j'}$ if either $w(v_iv_j) < w(v_iv_{j'})$ or $w(v_iv_j) = w(v_iv_{j'})$ and $j < j'$. With such a partial order on the edge set, the set $E_w^k(v)$ of $\min\{|N(v)|, k-1\}$ greatest edges (according to this ordering) around a vertex is uniquely defined. Observe that every edge incident to v and not in $E_w^k(v)$ is smaller than an edge of $E_k(v)$ for $\leq_w$.

Let G_w^k be the graph with vertex set $V(G)$ and edge set $\bigcup_{v \in V(G)} E_w^k(v)$. Observe that every vertex of $E_w^k(v)$ has degree at least $\min\{|N(v)|, k-1\}$, but a vertex may have an arbitrarily large degree. For if any edge incident to v has a greater weight than any edge not incident to v, the degree of v in G_w^k is equal to its degree in G. However we now prove that at least one vertex has degree $k-1$.

Proposition 2. *If (G, w) is an edge-weighted graph, then G_w^k has a vertex of degree at most $k-1$.*

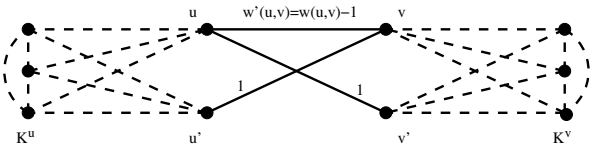


Figure A.1: Construction of G' from G using edge $uv \in E(G)$ and $k = 4$ colours. Dashed edges represent edges of weight M.

Proof. Suppose for a contradiction, that every vertex has degree at least k, then for every vertex x there is an edge xy in $E(G_w^k) \setminus E_w^k(x)$, and so in $E_w^k(y) \setminus E_w^k(x)$. Therefore, there must be a cycle $(x_1, \ldots, x_r)$ such that, for all $1 \leq i \leq r$, $x_i x_{i+1} \in E_w^k(x_{i+1}) \setminus E_w^k(x_i)$ (with $x_{r+1} = x_1$). It follows that $x_1x_2 \leq_w x_2x_3 \leq_w \cdots \leq_w x_rx_1 \leq_w x_1x_2$. Hence, by definition, $w(x_1x_2) = w(x_2x_3) = \cdots = w(x_rx_1) = w(x_1x_2)$. Let m be the integer such that x_m has maximum index in the ordering $v_1, \ldots, v_n$. Then there exists j and j' such that $x_m = v_j$ and $x_{m+2} = v_{j'}$. By definition of m, we have $j > j'$. But this contradicts the fact that $x_m x_{m+1} \leq_w x_{m+1}x_{m+2}$. □

Corollary 1. *If (G, w) is an edge-weighted graph, then G_w^k has a proper k-colouring.*

Proof. By induction on the number of vertices. By Proposition 2, G_w^k has a vertex x of degree at most $k - 1$. Trivially, $G_w^k - x$ is a subgraph of $(G - x)_w^k$. By the induction hypothesis, $(G - x)_w^k$ has a proper k-colouring, which is also a proper k-colouring of $G_w^k - x$. This colouring can be extended in a proper k-colouring of G_w^k, by assigning to x a colour not assigned to any of its $k - 1$ neighbours. □

Corollary 2. *If (G, w) is an edge-weighted graph, then $T_k(G, w) \leq \Delta(G \setminus E(G_w^k), w)$.*

Transformation

In this section, we prove that the Threshold Improper Colouring problem can be transformed into a problem mixing proper and improper colouring. More precisely, we prove the following:

Theorem 8. *Let (G, w) be an edge-weighted graph where w is an integer-valued function, and let k be a positive integer. We can construct an edge-weighted graph (G^*, w^*) such that $w^*(e) \in \{1, M\}$ for any $e \in E(G^*)$, satisfying $T_k(G, w) = T_k(G^*, w^*)$, where $M = 1 + \sum_{e \in E(G)} w(e)$.*

Proof. Consider the function $f(G, w) = \sum_{\{e \in E(G) | w(e) \neq M\}} (w(e) - 1)$.

If $f(G, w) = 0$, all edges have weight either one or M and G has the desired property. In this case, $G^* = G$. Otherwise, we construct a graph G' and a function w' such that $T_k(G', w') = T_k(G, w)$, but $f(G', w') = f(G, w) - 1$. By repeating this operation $f(G, w)$ times we get the required edge-weighted graph (G^*, w^*).

In case $f(G, w) > 0$, there exists an edge $e = uv \in E(G)$ such that $2 \leq w(e) < M$. G' is obtained from G by adding two complete graphs on $k - 1$ vertices K^u and K^v and two new vertices u' and v'. We join u and u' to all the vertices of K^u and v and v' to all the vertices of K^v. We assign weight M to all these edges. Note that, u and u' (v and v') always have the same colour, namely the remaining colour not used in K^u (resp. K^v).

We also add two edges uv' and $u'v$ both of weight 1. The edges of G keep their weight in G', except the edge $e = uv$ whose weight is decreased by one unit, i.e. $w'(e) = w(e) - 1$. Thus, $f(G', \omega') = f(G, \omega) - 1$ as we added only edges of weights 1 and M and we decreased the weight of e by one unit.

Now consider a weighted t-improper k-colouring c of (G, w). We produce a weighted t-improper k-colouring c' of (G', w') as follows: we keep the colours of all the vertices in G, we assign to u' (v') the same colour as u (resp. v), and we assign to K^u (resp. K^v) the $k - 1$ colours different from the one used in u (resp. v).

Conversely, from any weighted improper k-colouring c' of (G', w'), we get a weighted improper k-colouring c of (G, w) by just keeping the colours of the vertices that belong to G.

For such colourings c and c' we have that $I_x(G, w, c) = I_x(G', w', c')$, for any vertex x of G different from u and v. For $x \in K^u \cup K^v$, $I_x(G', w', c') = 0$. The neighbours of u with the same colour as u in G' are the same as in G, except possibly v' which has the same colour of u if, and only if, v has the same colour of u. Let $\epsilon = 1$ if v has the same colour as u, otherwise $\epsilon = 0$. As the weight of uv decreases by one and we add the edge uv' of weight 1 in G', we

get $I_u(G', w', c') = I_u(G, w, c) - \epsilon + w'(u, v')\epsilon = I_u(G, w, c)$. Similarly, $I_v(G', w', c') = I_v(G, w, c)$. Finally, $I_{u'}(G', w', c') = I_{v'}(G', w', c') = \epsilon$. But $I_u(G', w', c') \geq (w(u, v) - 1)\epsilon$ and so $I_{u'}(G', w', c') \leq I_u(G', w', c')$ and $I_{v'}(G', w', c') \leq I_v(G', w', c')$. In summary, we have

$$\max_x I_x(G', w', c') = \max_x I_x(G, w, c)$$

and therefore $T_k(G, w) = T_k(G', w')$. □

In the worst case, the number of vertices of G^* is $n + m(w_{max} - 1)2k$ and the number of edges of G^* is $m + m(w_{max} - 1)[(k + 4)(k - 1) + 2]$ with $n = |V(G)|$, $m = |E(G)|$ and $w_{max} = \max_{e \in E(G)} w(e)$.

In conclusion, this construction allows to transform the Threshold Improper Colouring problem into a problem mixing proper and improper colouring. Therefore the problem consists in finding the minimum l such that a (non-weighted) l-improper k-colouring of G^* exists with the constraint that some subgraphs of G^* must admit a proper colouring. The equivalence of the two problems is proved here only for integers weights, but it is possible to adapt the transformation to prove it for rational weights.

A.4 Squares of Particular Graphs

As mentioned in the introduction, Weighted Improper colouring is motivated by networks of antennas similar to grids [AAG+05]. In these networks, the noise generated by an antenna undergoes an attenuation with the distance it travels. It is often modelled by a decreasing function of d, typically $1/d^\alpha$ or $1/(2^{d-1})$.

Here we consider a simplified model where the noise between two neighbouring antennas is normalised to 1, between antennas at distance two is 1/2 and 0 when the distance is strictly greater than two. Studying this model of interference corresponds to study the Weighted Improper colouring of the square of the graph G, that is the graph G^2 obtained from G by joining every pair of vertices at distance two, and to assign weights $w_2(e) = 1$, if $e \in E(G)$, and $w_2(e) = 1/2$, if $e \in E(G^2) \setminus E(G)$. Observe that in this case the interesting threshold values are the non-negative multiples of 1/2.

Figure A.2 shows some examples of colouring for the square grid. In Figure A.2b, each vertex x has neither a neighbour nor a vertex at distance two coloured with its own colour, so $I_x(G^2, w_2, c) = 0$ and G^2 admits a weighted 0-improper 5-colouring. In Figure A.2c, each vertex x has no neighbour with its colour and at most one vertex of the same colour at distance 2. So $I_x(G^2, w_2, c) = 1/2$ and G^2 admits a weighted 0.5-improper 4-colouring.

For any $t \in \mathbb{R}_+$, we determine the weighted t-improper chromatic number for the square of infinite paths, square grids, hexagonal grids and triangular grids under the interference model w_2. We also present lower and upper bounds for $\chi_t(T^2, w_2)$, for any tree T and any threshold t.

Infinite paths and trees

In this section, we characterise the weighted t-improper chromatic number of the square of an infinite path, for all positive real t. Moreover, we present lower and upper bounds for $\chi_t(T^2, w_2)$, for a given tree T.

Theorem 9. *Let $P = (V, E)$ be an infinite path. Then,*

$$\chi_t(P^2, w_2) = \begin{cases} 3, & \text{if } 0 \leq t < 1; \\ 2, & \text{if } 1 \leq t < 3; \\ 1, & \text{if } 3 \leq t. \end{cases}$$

Proof. Let $V = \{v_i \mid i \in \mathbb{Z}\}$ and $E = \{(v_{i-1}, v_i) \mid i \in \mathbb{Z}\}$. Each vertex of P has two neighbours and two vertices at distance two. Consequently, the equivalence $\chi_t(P^2, \omega_2) = 1$ if, and only if, $t \geq 3$ holds trivially.

There is a 2-colouring c of (P^2, w_2) with maximum interference 1 by just colouring v_i with colour $(i \ mod \ 2) + 1$. So $\chi_t(P^2, w_2) \leq 2$ if $t \geq 1$. We claim that there is no weighted 0.5-improper 2-colouring of (P^2, w_2). By contradiction, suppose that c is such a colouring. If $c(v_i) = 1$, for some $i \in \mathbb{Z}$, then $c(v_{i-1}) = c(v_{i+1}) = 2$ and $c(v_{i-2}) = c(v_{i+2}) = 1$. This is a contradiction because v_i would have interference 1.

Finally, the colouring $c(v_i) = (i \ mod \ 3) + 1$, for every $i \in \mathbb{Z}$, is a feasible weighted 0-improper 3-colouring. □

Theorem 10. *Let $T = (V, E)$ be a (non-empty) tree. Then,* $\left\lceil \frac{\Delta(T) - \lfloor t \rfloor}{2t+1} \right\rceil + 1 \leq \chi_t(T^2, w_2) \leq \left\lceil \frac{\Delta(T)-1}{2t+1} \right\rceil + 2$.

Proof. The lower bound is obtained by two simple observations. First, $\chi_t(H, w) \leq \chi_t(G, w)$, for any $H \subseteq G$. Let T be a tree and v be a node of maximum degree in T. Then, observe that the weighted t-improper chromatic number of the subgraph of T^2 induced by v and its neighbourhood is at least $\lceil \frac{\Delta(T)-\lfloor t \rfloor}{2t+1} \rceil + 1$. Indeed, the colour of v can be assigned to at most $\lfloor t \rfloor$ vertices on its neighbourhood. Any other colour used in the neighbourhood of v cannot appear in more than $2t + 1$ vertices because each pair of vertices in the neighbourhood of v is at distance two.

Let us look now at the upper bound. Choose any node $r \in V$ to be the root of T. Colour r with colour 1. Then, by a breadth-first traversal in the tree, for each visited node v colour all the children of v with the $\lceil \frac{\Delta(T)-1}{2t+1} \rceil$ colours different from the ones assigned to v and to its parent in such a way that at most $2t + 1$ nodes have the same colour. This is a feasible weighted t-improper k-colouring of T^2, with $k \leq \lceil \frac{\Delta(T)-1}{2t+1} \rceil + 2$, since each vertex interferes with at most $2t$ vertices at distance two which are children of its parent. □

For a tree T and the weighted function w^2, Theorem 10 provides upper and lower bounds on $\chi_t(T^2, w_2)$, but we do not know the computational complexity of determining $\chi_t(T^2, w_2)$.

Grids

In this section, we show the optimal values of $\chi_t(G^2, w_2)$, whenever G is an infinite square, hexagonal or triangular grid, for all the possible values of t.

Square Grid

The square grid is the graph $\mathfrak{S}$ in which the vertices are all integer linear combinations $ae_1 + be_2$ of the two vectors $e_1 = (1, 0)$ and $e_2 = (0, 1)$, for any $a, b \in \mathbb{Z}$. Each vertex (a, b) has four neighbours: its *down neighbour* $(a, b-1)$, its *up neighbour* $(a, b+1)$, its *right neighbour* $(a+1, b)$ and its *left neighbour* $(a-1, b)$ (see Figure A.2a).

Theorem 11.

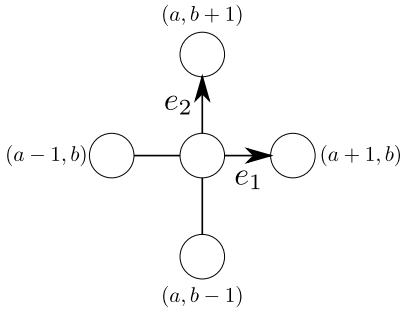
$$\chi_t(\mathfrak{S}^2, w_2) = \begin{cases} 5, & \text{if } t = 0; \\ 4, & \text{if } t = 0.5; \\ 3, & \text{if } 1 \leq t < 3; \\ 2, & \text{if } 3 \leq t < 8; \\ 1, & \text{if } 8 \leq t. \end{cases}$$

Proof. If $t = 0$, then the colour of vertex (a, b) must be different from the ones used on its four neighbours. Moreover, all the neighbours have different colours, as each pair of neighbours is at distance two. Consequently, at least five colours are needed. The following construction provides a weighted 0-improper 5-colouring of $(\mathfrak{S}^2, w_2)$: for $0 \leq j \leq 4$, let $A_j = \{(j, 0) + a(5e_1) + b(2e_1 + 1e_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 4$, assign the colour $j + 1$ to all the vertices in A_j (see Figure A.2b).

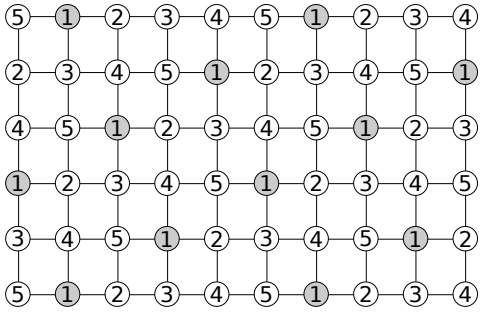
When $t = 0.5$, we claim that at least four colours are needed to colour $(\mathfrak{S}^2, w_2)$. The proof is by contradiction. Suppose that there exists a weighted 0.5-improper 3-colouring of it. Let (a, b) be a vertex coloured 1. None of its neighbours is coloured 1, otherwise (a, b) has interference 1. If three neighbours have the same colour, then each of them will have interference 1. So two of its neighbours have to be coloured 2 and the two other ones 3 (see Figure A.3a). Now consider the four nodes $(a-1, b-1)$, $(a - 1, b + 1)$, $(a + 1, b - 1)$ and $(a + 1, b + 1)$. For all configurations, at least two of these four vertices have to be coloured 1 (the ones indicated by a * in Figure A.3a). But then (a, b) will have interference at least 1, a contradiction. A weighted 0.5-improper 4-colouring of $(\mathfrak{S}^2, w_2)$ can be obtained as follows (see Figure A.2c): for $0 \leq j \leq 3$, let $B_j = \{(j, 0) + a(4e_1) + b(3e_1 + 2e_2) \mid \forall a, b \in \mathbb{Z}\}$ and $B'_j = \{(j + 1, 2) + a(4e_1) + b(3e_1 + 2e_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 3$, assign the colour $j + 1$ to all the vertices in B_j and in B'_j.

If $t = 1$, there exists a weighted 1-improper 3-colouring of $(\mathfrak{S}^2, w_2)$ given by the following construction: for $0 \leq j \leq 2$, let $C_j = \{(j, 0) + a(3e_1) + b(e_1 + e_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 2$, assign the colour $j + 1$ to all the vertices in C_j.

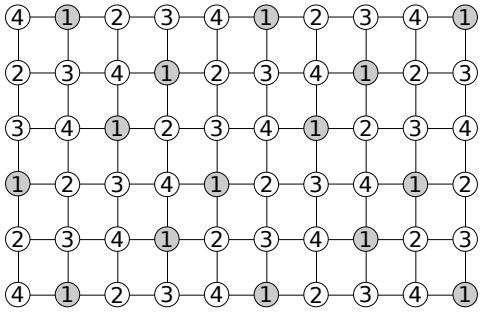
Now we prove by contradiction that for $t = 2.5$ we still need at least three colours in a weighted 2.5-improper colouring of $(\mathfrak{S}^2, w_2)$. Consider a weighted 2.5-improper 2-colouring of $(\mathfrak{S}^2, w_2)$ and let (a, b) be a ver-



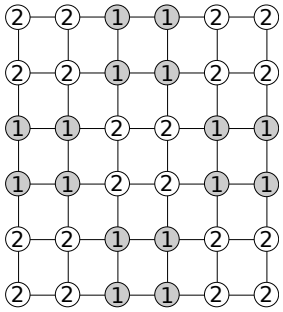
(a)



(b)



(c)



(d)

Figure A.2: Optimal colourings of $(\mathfrak{S}^2, w_2)$: (b) weighted 0-improper 5-colouring of $(\mathfrak{S}^2, w_2)$, (c) weighted 0.5-improper 4-colouring of $(\mathfrak{S}^2, w_2)$, and (d) weighted 3-improper 2-colouring of $(\mathfrak{S}^2, w_2)$.

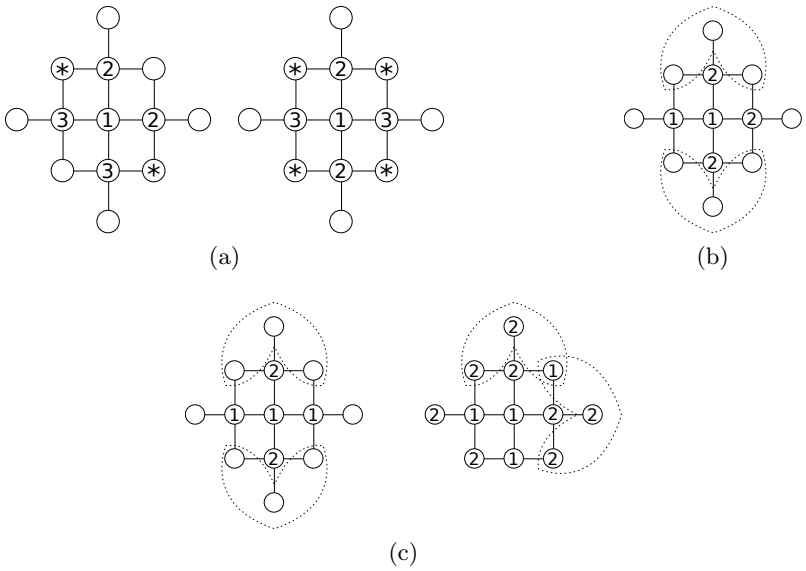


Figure A.3: Lower bounds for the square grid: (a) if $t \leq 0.5$ and $k \leq 3$, there is no weighted t-improper k-colouring of $(\mathfrak{S}^2, w_2)$; (b) the first case when $t \leq 2.5$ and $k \leq 2$, and (c) the second case.

tex coloured 1. Vertex (a, b) has at most two neighbours of colour 1, otherwise it will have interference 3. We distinguish three cases:

1. Exactly one of its neighbours is coloured 1; let $(a-1, b)$ be this vertex. Then, the three other neighbours are coloured 2 (see Figure A.3b). Consider the two sets of vertices $\{(a-1, b-1), (a+1, b-1), (a, b-2)\}$ and $\{(a-1, b+1), (a+1, b+1), (a, b+2)\}$ (these sets are surrounded by dotted lines in Figure A.3b); each of them has at least two vertices coloured 1, otherwise the vertex $(a, b-1)$ or $(a, b+1)$ will have interference 3. But then (a, b) having four vertices at distance two coloured 1 has interference 3, a contradiction.

2. Two neighbours of (a, b) are coloured 1.

   a) These two neighbours are opposite, say $(a-1, b)$ and $(a+1, b)$ (see Figure A.3c left). Consider again the two sets $\{(a-1, b-1), (a+1, b-1), (a, b-2)\}$ and $\{(a-1, b+1), (a+1, b+1), (a, b+$

2)} (these sets are surrounded by dotted lines in Figure A.3c left); they both contain at least one vertex of colour 1 and therefore (a, b) will have interference 3, a contradiction.

b) The two neighbours of colour 1 are of the form $(a, b-1)$ and $(a-1, b)$ (see Figure A.3c right). Consider the two sets of vertices $\{(a+1, b-1), (a+1, b+1), (a+2, b)\}$ and $\{(a+1, b+1), (a-1, b+1), (a, b+2)\}$ (these sets are surrounded by dotted lines in Figure A.3c right); these two sets contain at most one vertex of colour 1, otherwise (a, b) will have interference 3. Moreover, each of these sets cannot be completely coloured 2, otherwise $(a+1, b)$ or $(a, b+1)$ will have interference at least 3. So vertices $(a+1, b-1)$, $(a+2, b)$, $(a, b+2)$ and $(a-1, b+1)$ are of colour 2 and the vertex $(a+1, b+1)$ is of colour 1. But then $(a-2, b)$ and $(a-1, b-1)$ are of colour 2, otherwise (a, b) will have interference 3. Thus, vertex $(a-1, b)$ has exactly one neighbour coloured 1 and we are again in Case 1.

3. All neighbours of (a, b) are coloured 2. If one of these neighbours has itself a neighbour (distinct from (a, b)) of colour 2, we are in Case 1 or 2 for this neighbour. Therefore, all vertices at distance two from (a, b) have colour 1 and the interference in (a, b) is 4, a contradiction.

A weighted 3-improper 2-colouring of $(\mathfrak{S}^2, w_2)$ can be obtained as follows: a vertex of the grid (a, b) is coloured with colour $(\lfloor \frac{a}{2} \rfloor + \lfloor \frac{b}{2} \rfloor \mod 2) + 1$, see Figure A.2d.

Finally, since each vertex has four neighbours and eight vertices at distance two, there is no weighted 7.5-improper 1-colouring of $(\mathfrak{S}^2, w_2)$ and, whenever $t \geq 8$, one colour suffices. □

Hexagonal Grid

There are many ways to define the system of coordinates of the hexagonal grid. Here, we use grid coordinates as shown in Figure A.4. The hexagonal grid graph is then the graph $\mathfrak{H}$ whose vertex set consists of pairs of integers $(a, b) \in \mathbb{Z}^2$ and where each vertex (a, b) has three neighbours: $(a-1, b)$, $(a+1, b)$, and $(a, b+1)$ if $a+b$ is odd, or $(a, b-1)$ otherwise.

Theorem 12.

$$\chi_t(\mathfrak{H}^2, w_2) = \begin{cases} 4, & \text{if } 0 \leq t < 1; \\ 3, & \text{if } 1 \leq t < 2; \\ 2, & \text{if } 2 \leq t < 6; \\ 1, & \text{if } 6 \leq t. \end{cases}$$

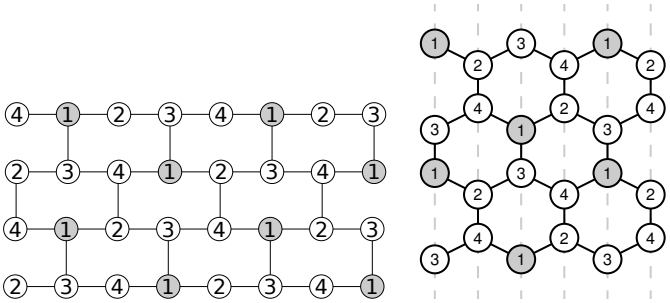
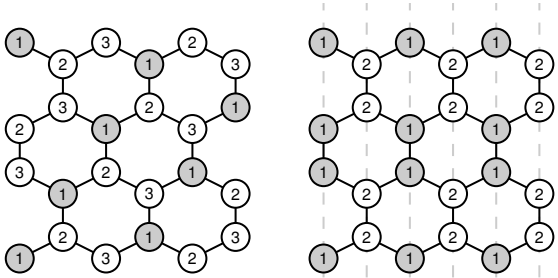


Figure A.4: Weighted 0-improper 4-colouring of $(\mathfrak{H}^2, w_2)$. Left: Graph with coordinates. Right: Corresponding hexagonal grid in the euclidean space.



(a) $t = 1$, $k = 3$ (b) $t = 2$, $k = 2$

Figure A.5: (a) weighted 1-improper 3-colouring of $(\mathfrak{H}^2, w_2)$ and (b) weighted 2-improper 2-colouring of $(\mathfrak{H}^2, w_2)$.

Proof. Note first, that when $t = 0$, at least four colours are needed to colour the grid, because a vertex and its neighbourhood in $\mathfrak{H}$ form a clique of size four in $\mathfrak{H}^2$. The same number of colours are needed if we allow a threshold $t = 0.5$. To prove this fact, let A be a vertex (a, b) of $\mathfrak{H}$ and $B = (a-1, b)$, $C = (a, b-1)$ and $D = (a+1, b)$ be its neighbours in $\mathfrak{H}$. Denote by $G = (a-2, b)$, $E = (a-1, b-1)$, $F = (a-2, b-1)$, $H = (a+1, b-1)$, $I = (a+2, b-1)$ and $J = (a+1, b-2)$ (see Figure A.6a). By

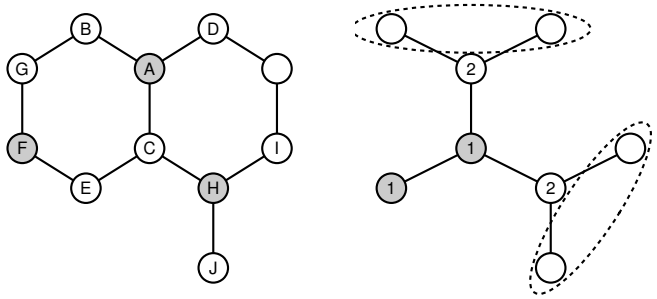
contradiction, suppose there exists a weighted 0.5-improper 3-colouring of $\mathfrak{H}^2$. Consider a node A coloured 1. Its neighbours B, C, D cannot be coloured 1 and they cannot all have the same colour. W.l.o.g., suppose that two of them B and C have colour 2 and D has colour 3. Then E, F and G cannot be coloured 2 because of the interference constraint in B and C. If F is coloured 3, then G and E are coloured 1, creating interference 1 in A. So F must be coloured 1 and G and E must be coloured 3. Then, H can be neither coloured 2 (interference in C) nor 3 (interference in E). So H is coloured 1. The vertex I is coloured 3, otherwise the interference constraint in H or in C is not satisfied. Then, J can receive neither colour 1, because of the interference in H, nor colour 2, because of the interference in C, nor colour 3, because of the interference in I.

There exists a construction attaining this bound and the number of colours, i.e. a 0-improper 4-colouring of $(\mathfrak{H}^2, w_2)$ as depicted in Figure A.4. We define for $0 \leq j \leq 3$ the sets of vertices $A_j = \{(j,0) + a(4e_1) + b(2e_1 + e_2) | \forall a, b \in \mathbb{Z}\}$. We then assign the colour $j+1$ to the vertices in A_j. This way no vertex experiences any interference as vertices of the same colours are at distance at least three.

For $t = 1.5$ it is not possible to colour the grid with less than three colours. By contradiction, suppose that there exists a weighted 1.5-improper 2-colouring. Consider a vertex A coloured 1. If all of its neighbours are coloured 2, they have already interference 1, so all the vertices at distance two from A need to be coloured 1; this gives interference 3 in A. Therefore one of A's neighbours, say D, has to be coloured 1 and consider that the other two neighbours B and C are coloured 2. B and C have at most one neighbour of colour 2. It implies that A has at least two vertices at distance two coloured 1. This is a contradiction, because the interference in A would be at least 2 (see Figure A.6b).

Figure A.5a presents a weighted 1-improper 3-colouring of $(\mathfrak{H}^2, w_2)$. To obtain this colouring, let $B_j = \{(j,0) + a(3e_1) + b(e_1 + e_2) \mid \forall a, b \in \mathbb{Z}\}$, for $0 \leq j \leq 2$. Then, we colour all the vertices in the set B_j with colour $j+1$, for every $0 \leq j \leq 2$.

For $t < 6$, it is not possible to colour the grid with one colour. As a matter of fact, each vertex has three neighbours and six vertices at distance two in $\mathfrak{H}$. Using one colour leads to an interference equal to 6. There exists a 2-improper 2-colouring of the hexagonal grid as depicted



(a) (b)

Figure A.6: Lower bounds for the hexagonal grid. (a) when $t \leq 0.5$ and $k \leq 3$, there is no weighted t-improper k-colouring of $(\mathfrak{H}^2, w_2)$; (b) vertices coloured 2 force a vertex coloured 1 in each ellipse, leading to interference 2 in central node.

in Figure A.5b. We define for $0 \leq j \leq 1$ the sets of vertices $C_j = \{(j,0) + a(2e_1) + be_2 | \forall a, b \in \mathbb{Z}\}$. We then assign the colour $j+1$ to the vertices in C_j.

□

Triangular Grid

The triangular grid is the graph $\mathfrak{T}$ whose vertices are all the integer linear combinations $af_1 + bf_2$ of the two vectors $f_1 = (1,0)$ and $f_2 = (\frac{1}{2}, \frac{\sqrt{3}}{2})$. Thus we may identify the vertices with the ordered pairs (a,b) of integers. Each vertex $v = (a,b)$ has six neighbours: its *right neighbour* $(a+1,b)$, its *right-up neighbour* $(a,b+1)$, its *left-up neighbour* $(a-1,b+1)$, its *left neighbour* $(a-1,b)$, its *left-down neighbour* $(a,b-1)$ and its *right-down neighbour* $(a+1,b-1)$ (see Figure A.8).

Theorem 13.

$$\chi_t(\mathfrak{T}^2, w_2) = \begin{cases} 7, & \text{if } t = 0; \\ 6, & \text{if } t = 0.5; \\ 5, & \text{if } t = 1; \\ 4, & \text{if } 1.5 \leq t < 3; \\ 3, & \text{if } 3 \leq t < 5; \\ 2, & \text{if } 5 \leq t < 12; \\ 1, & \text{if } 12 \leq t. \end{cases}$$

Proof. The full proof of this theorem comprises 45 pages of technical details. It can bee found in [ABG$^+$11a]. Here, we only present the optimal constructions.

There is a weighted 0-improper 7-colouring of $(\mathfrak{T}^2, w_2)$ as depicted in Figure A.7a. This colouring can be obtained by the following construction: for $0 \leq j \leq 6$, let $A_j = \{(j, 0) + a(7f_1) + b(2f_1 + f_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 6$, assign the colour $j + 1$ to all the vertices in A_j.

A weighted 0.5-improper 6-colouring of $(\mathfrak{T}^2, w_2)$ can be obtained by the following construction (see Figure A.7b): for $0 \leq j \leq 11$, let $B_j = \{(j, 0) + a(12f_1) + b(2f_1 + f_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 5$, assign the colour $j + 1$ to all the vertices in B_j, B_6 with colour 2, B_7 with colour 1, B_8 with colour 4, B_9 with colour 3, B_{10} with colour 6 and B_{11} with colour 5.

To obtain a weighted 1-improper 5-colouring of $(\mathfrak{T}^2, w_2)$, for $0 \leq j \leq 4$, let $C_j = \{(j, 0) + a(5f_1) + b(2f_1 + f_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 4$, assign the colour $j + 1$ to all the vertices in C_j. See Figure A.7c.

$(\mathfrak{T}^2, w_2)$ has a weighted 1.5-improper 4-colouring as depicted in Figure A.7d. Formally, this colouring can be obtained by the following construction: for $0 \leq j \leq 3$, let $D_j = \{(j, 0) + a(4f_1) + b(f_1 + 2f_2) \mid \forall a, b \in \mathbb{Z}\}$; then assign colour 4 to all the vertices in D_0, 1 to all the vertices in D_1, 3 to all the vertices in D_2 and 2 to all the vertices in D_3. Now, for $0 \leq j \leq 3$, let $D'_j = \{(j, 1) + a(4f_1) + b(f_1 + 2f_2) \mid \forall a, b \in \mathbb{Z}\}$. Then, for $0 \leq j \leq 3$, assign colour $j + 1$ to all the vertices in D'_j.

For a weighted 3-improper 3-colouring of $(\mathfrak{T}^2, w_2)$ set, for $0 \leq j \leq 2$, $E_j = \{(j, 0) + a(3f_1) + b(f_2) \mid \forall a, b \in \mathbb{Z}\}$. Then, for $0 \leq j \leq 2$, assign the colour $j + 1$ to all the vertices in E_j. See Figure A.7e.

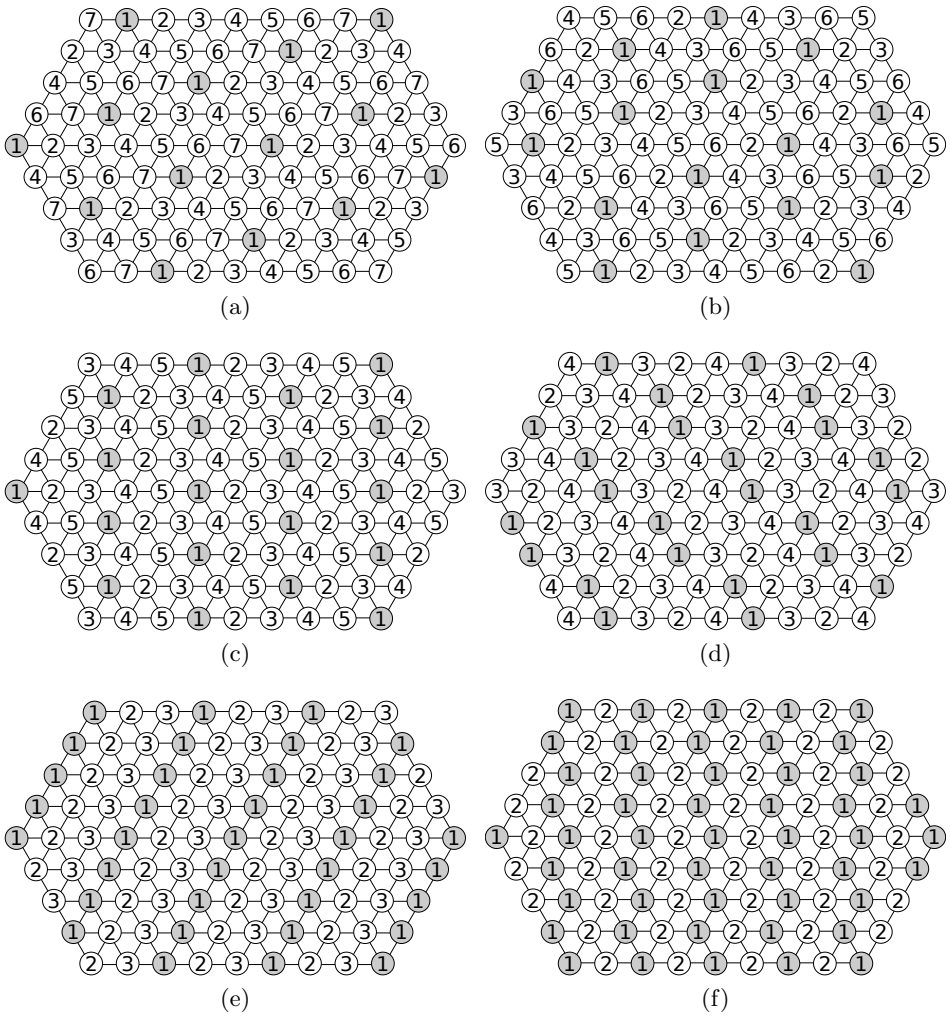


Figure A.7: Optimal colourings of $(\mathfrak{T}^2, w_2)$: (a) weighted 0-improper 7-colouring, (b) weighted 0.5-improper 6-colouring, (c) weighted 1-improper 5-colouring, (d) weighted 1.5-improper 4-colouring, (e) weighted 3-improper 3-colouring, and (f) weighted 5-improper 2-colouring.

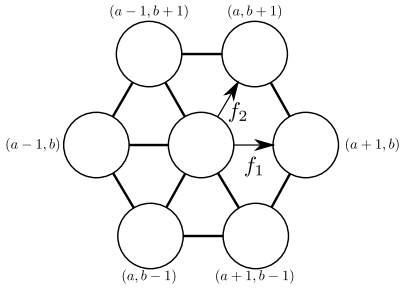


Figure A.8: Notations used in proofs of existence of weighted improper colourings of $(\mathfrak{T}^2, w_2)$.

A weighted 5-improper 2-colouring of $(\mathfrak{T}^2, w_2)$ is obtained as follows: for $0 \leq j \leq 1$, let $F_j = \{(j,0) + a(2f_1) + b(f_1 + 2f_2) \mid \forall a, b \in \mathbb{Z}\}$ and $F'_j = \{(j-1,1) + a(2f_1) + b(f_1 + 2f_2) \mid \forall a, b \in \mathbb{Z}\}$. Then, for $0 \leq j \leq 1$, assign the colour $j+1$ to all the vertices in F_j and in F'_j. See Figure A.7f. □

A.5 Integer Linear Programming Formulations, Algorithms and Results

In this section, we look at how to solve the Weighted Improper colouring and Threshold Improper colouring for general instances inspired by the practical motivation. We present integer linear programming models for both problems. These models can be solved exactly for small sized instances using solvers like CPLEX[1]. For larger instances, the solvers can take a prohibitive time to provide exact solutions. It is usually possible to obtain a sub-optimal solution stopping the solver after a limited time. If the time is too short, the quality of the solution may be unsatisfactory. Thus, we introduce two algorithmic approaches to find good solutions for Threshold Improper colouring in a short time: a simple polynomial-time greedy heuristic and an exact Branch-and-Bound algorithm. We compare the three methods on different sets of instances, among them Poisson-Voronoi tessellations as they are good models of antenna networks [BKLZ97, GK00, HAB+09].

[1] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Integer Linear Programming Models

Given an edge-weighted graph $G = (V, E, w)$, $w : E \to \mathbb{R}_+^*$, and a positive real threshold t, we model Weighted Improper Colouring by using two kinds of binary variables. Variable x_{ip} indicates if vertex i is coloured p and variable c_p indicates if colour p is used, for every $1 \leq i \leq n$ and $1 \leq p \leq l$, where l is an upper bound for the number of colours needed in an optimal weighted t-improper colouring of G. l can be trivially chosen of value n, but a better value may be given by the results of Section A.3. The model follows:

$$
\begin{array}{rcl}
\min & \sum_{p=1}^{l} c_p & \\
\text{subject to} & & \\
& \sum_{ij \in E \text{ and } j \neq i} w(i,j) x_{jp} \leq t + M(1 - x_{ip}) & \forall i \in V, 1 \leq p \leq l \\
& c_p \geq x_{ip} & \forall i \in V, 1 \leq p \leq l \\
& \sum_{p=1}^{l} x_{ip} = 1 & \forall i \in V \\
& x_{ip} \in \{0,1\} & \forall i \in V, 1 \leq p \leq l \\
& c_p \in \{0,1\} & 1 \leq p \leq l
\end{array}
$$

where M is a large integer. For instance, it is sufficient to choose $M > \sum_{uv \in E} w(u,v)$.

For Threshold Improper Colouring, given an edge-weighted graph $G = (V, E, w)$, $w : E \to \mathbb{R}_+^*$, and a number of possible colours $k \in \mathbb{N}^*$, the model we consider is:

$$
\begin{array}{rcl}
\min & t & \\
\text{subject to} & & \\
& \sum_{ij \in E \text{ and } j \neq i} w(i,j) x_{jp} \leq t + M(1 - x_{ip}) & \forall i \in V, 1 \leq p \leq l \\
& \sum_{p=1}^{k} x_{ip} = 1 & \forall i \in V \\
& x_{ip} \in \{0,1\} & \forall i \in V, 1 \leq p \leq l
\end{array}
$$

We give directly these models to the ILP solver CPLEX without using any preprocessing or any other technique to speed the search for an optimal solution.

Algorithmic approach

In this section, we show a Branch-and-Bound algorithm and a randomised greedy heuristic to tackle Threshold Improper colouring.

Both are based on common procedures to determine the order in which vertices are coloured and colours are tried for a single vertex. Although, the Branch-and-Bound needs an ordering of the vertices to be coloured as input while the heuristic colours the vertices at the same time the order is being processed.

Order of vertices and colours

The order in which the vertices are chosen to be coloured follows essentially the same idea as the DSATUR algorithm, created by Daniel Brélaz [Bré79].

Consider a graph $G = (V, E, w)$, $w : E \to \mathbb{R}_+^*$ and a partial colouring $c : U \to \{1, \ldots, k\}$, where $U \subseteq V$. We say that vertex v is *coloured* if $v \in U$, otherwise it is *uncoloured*. We define the *total potential interference* in vertex v to be:

$$I_{c,v}^{tot} = \sum_{\{u \in V | uv \in E \text{ and } v \notin U\}} w(u, v),$$

which is the sum of interferences for all colours induced in v by all its already coloured neighbours.

The idea for both algorithms is to first colour vertices with highest total potential interference. Whenever more than one vertex has the highest total potential interference, one of them is chosen at random. At the beginning, when all vertices have $I_{c,v}^{tot} = 0$, one of the highest weighted degree is chosen instead.

Consider the following steps:

1. Colour a random vertex with maximal sum of incoming weights.
2. Colour a random vertex with maximal total potential interference.
3. If all vertices all coloured, stop. Otherwise, repeat step 2.

Note that the total potential interference does not depend on the actual colours assigned to the vertices. Thus, in order to decide which is the next vertex to be coloured, both algorithms, Branch-and-Bound and heuristic, use these three steps. However, the Branch-and-Bound algorithm needs an order to colour the vertices as input. So, we decide which order to give to the Branch-and-Bound algorithm as input by running these three steps and using a single colour.

The procedure above specifies the order of vertices. For the order of colours to try, we define the *potential interference* in vertex v for colour x as:

$$I_{c,v,x} = \sum_{\{u \in V | uv \in E \text{ and } c(v)=x\}} w(u,v)$$

Anytime one of our algorithms colours a vertex, it tries the colours in order of increasing potential interference.

Branch-and-Bound Algorithm

Having an ordering procedure for both vertices and colours, we construct a simple Branch-and-Bound algorithm using them. The order of vertices to colour is fixed before running the algorithm, following the procedure in Section A.5. Then, the ordered vertices are coloured by a recursive function that tries all the possible colours for each vertex as far as no interference constraint is violated. The order in which the colours are tried is also presented in the previous section. Our algorithm outputs all the feasible colourings it finds and, as all the possible colours are tried, the one using the minimum number of colours is an optimal one.

Here you have a pseudo code for the algorithm:

Algorithm 1: Branch&Bound

input : edge-weighted graph (G, w), number of colours k, partial colouring c, upper bound t and corresponding colouring $\tilde{c}$, order in which vertices should be coloured O

output: new upper bound t' and corresponding colouring $\tilde{c}$'

if $\max_{v \in V} I_v(G, w, c) \geq t$ **then**
  return t and $\tilde{c}$
if *all vertices are coloured in* c **then**
  return($\max_{v \in V} I_v(G, w, c)$ and c)
v = next vertex uncoloured in c according to O
for $x \in$ *possible colours in order of increasing* $I_{c,v,x}$ **do**
  (t and $\tilde{c}$) = Branch&Bound(G, k, $c \cap (v \leftarrow x)$, t, $\tilde{c}$, O)
return t and $\tilde{c}$

Where by $c \cap (v \leftarrow x)$ we mean a partial colouring where colour of vertex v (which was uncoloured in c) is set to x, and colours of all other vertices

are as in c. The algorithm is first called with all vertices uncoloured and $t = \infty$.

This algorithm displays a problematic behaviour. Imagine the partial colouring of the first few vertices yields good results locally, but implies a suboptimal interference at a more distant part of the graph. As the solution search takes exponential time in number of vertices, it is easy to envision that the time required to change the colouring of first vertices can be prohibitively long.

Greedy Heuristic

Here we propose a randomised greedy heuristic that, repeated multiple, but not exponentially many times, finds similar solutions to the above Branch-and-Bound without the mentioned problem. On the other hand, there are some solutions that are impossible to find with it, no matter the number of tries. An example of such an unobtainable solution is the optimal colouring of infinite square grid with 2 colours.

Algorithm 2: Levelling Heuristic

input : edge-weighted graph (G, w), number of colours k, upper bound t
output: **failed** or a colouring c

$c(v) = \emptyset \quad \forall v \in V$
for $i \in \{1, \ldots, |V|\}$ **do**
  v = next, in order of increasing $I^{tot}_{c,v}$, vertex uncoloured in c
  for $x \in$ *possible colours in order of increasing* $I_{c,v,x}$ **do**
    if *colouring* v *with* x *does not cause* $\max_{v \in V} I_v(G, w, c) \geq t$ **then**
      $c(v) = x$
      break the inner loop
  if $c(v) = \emptyset$ **then**
    return failed
return c

Note that there is substantial randomness in this algorithm. The first vertex is the one of the ones with highest weighted degree. In the extreme case of regular graphs, this already means any vertex at random. If we use the simple interference function defined in Section A.4, then the second vertex is a random neighbour of the first vertex. Any time

there are multiple vertices with maximum total potential interference, we choose one at random. Similarly, the choice of colours is also random in case of equal potential interference.

Above algorithm is first called with $t = \infty$. Whenever it returns a colouring, we set $t = \max_{v \in V} I_v(G, w, c)$ for further iterations. It is repeated for a given number of times, or until a time limit is reached. In all instances in the following sections the program is constrained by a time limit. This means that the algorithm is called for an unknown, but probably big number of times (e.g. for a 6-regular grid of 1024 vertices the program performs on average over 500 runs of the algorithm per second).

As a *randomised greedy colouring* heuristic, it has to be ran multiple times to achieve satisfactory results. This is not a practical issue due to low computational cost of each run. The local immutable colouring decision is taken in time $O(k\Delta)$. Then, after each such decision, the interference has to be propagated, which takes linear time in the vertex degree. This gives a computational complexity bound $O(kn\Delta)$-time.

Validation

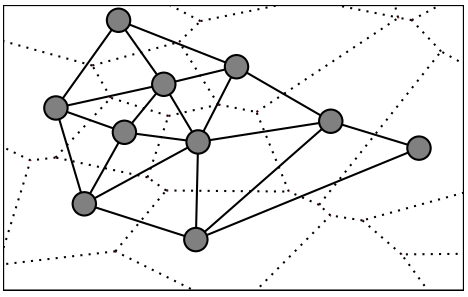
In this section we validate our algorithmic approaches at Threshold Improper colouring, by examining performance of their implementations. Tests cover a wide range of parameters, mostly on Delaunay graphs (see section A.5).

Implementation

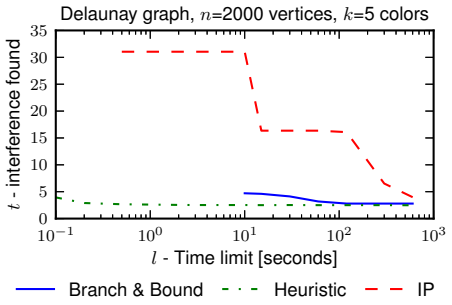
The ILP model is constructed out of the input graph and given directly to the CPLEX ILP solver. Branch-and-Bound algorithm is implemented in a straightforward way in the Python programming language. The greedy heuristic has a highly optimised implementation in the Cython programming language[2].

In results displayed below, all programs are run simultaneously on the same quad-core enterprise-grade CPU. Both the Branch-and-Bound and greedy heuristic are limited to a single core. CPLEX is allowed to both the remaining cores.

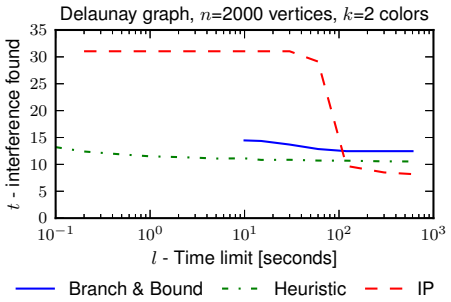
[2]This is the faster implementation envisioned in [ABG+11b].



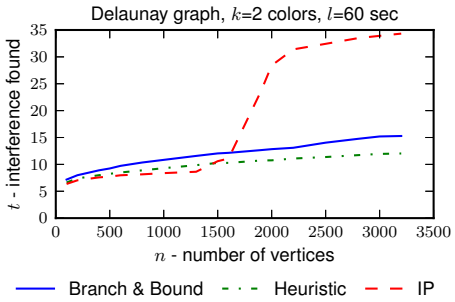
(a) Example Delaunay graph, dotted lines delimit corresponding Voronoi diagram cells



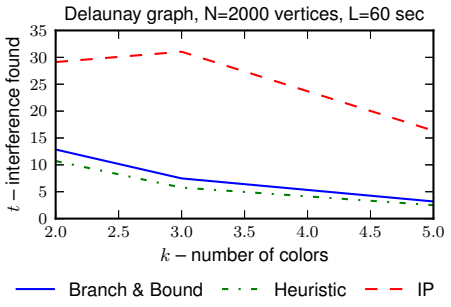
(b) Over time



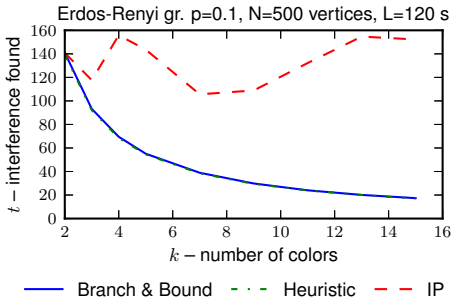
(c) Over time



(d) Over size



(e) Over colours



(f) Over colours

Figure A.9: Results comparison for Levelling heuristic, Branch-and-Bound algorithm and Integer Linear Programming Formulation.

Graphs

We consider random Delaunay graphs (dual of Voronoi diagram). This kind of graphs is an intuitive approximation of a network of irregular

cells. To obtain a graph in this class, take a set of random points uniformly distributed over a square. These represent the vertices of the graph. To obtain the edges, compute a Delaunay triangulation. This can be done e.g. with Fortune's algorithm described in [For87] in $O(n \log n)$ time.

See Figure A.9a for a depiction of a fragment of such graph. Vertices are arranged according to the positions of original random points. Dotted lines delimit corresponding Voronoi diagram cells. Only edges between vertices visible on the illustration are displayed.

Note that, to follow the model of the physical motivation, we are dealing with very sparse graphs. The average degree in Delaunay graph G converges to six (this results follows from the observation that G is planar and triangulated, thus $|E(G)| = 3|V(G)| - 6$ by Euler's formula). To get an idea about the proposed algorithms' performance in denser graphs, we also run some tests on Erdös-Rényi graphs with expected degree equal to 50.

The interference model we consider in all experiments is the one described in Section A.4: adjacent nodes interfere by 1 and nodes at distance two interfere by 1/2.

Results

Figure A.9 shows a performance comparison of the above-mentioned algorithms. For all the plots, each data point represents an average over a number (between 24 and 100) of different graphs. The experiment procedure is as follows. For each graph size considered in an experiment, a number of graphs is generated. Each of those graphs is transformed into a set of instances, one for each desired number of allowed colours. All the programs are run on each instance, once for each desired value of time limit. Finally, a data point is created with results and all the parameters, averaged over the number of graphs.

Figures A.9b and A.9c plot how results for a problem instance get enhanced with increasing time limits. Plot A.9d shows how well all the programmes scale with increasing graph sizes. Plots A.9e and A.9f show decreasing interference along increasing the number of colours allowed.

One immediate observation about both the heuristic and Branch-and-Bound algorithm is that they provide good solutions in relatively short time. On the other hand, with limited time, they fail to improve up

to optimal results, especially with a low number of allowed colours. An example near-optimal solution found in around three minutes was not improved by Branch-and-Bound in over six days.

The heuristic, is able to provide good results in sub-second times and scales better with increasing graph sizes than the Branch-and-Bound. It is also not prone to spending a lot time exploring a sub-optimal branch of a decision tree. Still, in many cases it is unable to obtain optimal results and displays a worse end result than an integer linear program, given enough time.

Solving the ILP does not scale with increasing graph sizes as well as our simple algorithms. Furthermore, Figure A.9e reveals one problem specific to ILP. When increasing the number of allowed colours, obtaining small interferences gets easier. But this introduces additional constraints in the formulation, thus increasing the complexity for a solver.

Proposed algorithms also work well for denser graphs. Figure A.9f plots interferences for different numbers of colours allowed found by the programs for an Erdös-Rényi graph with $n = 500$ and $p = 0.1$. This gives us an average degree equal to 50. Both Branch-and-Bound and heuristic programs achieve acceptable, and nearly identical, results. But the large number of constraints makes the integer linear programming formulation very inefficient.

A.6 Conclusion, Open Problems and Future Directions

In this paper, we introduced and studied a new colouring problem, Weighted Improper Colouring. This problem is motivated by the design of telecommunication antenna networks in which the interference between two vertices depends on different factors and can take various values. For each vertex, the sum of the interferences it receives should be less than a given threshold value.

We first give general bounds on the weighted-improper chromatic number. We then study the particular case of infinite paths, trees and grids: square, hexagonal and triangular. For these graphs, we provide their weighted-improper chromatic number for all possible values of t. Finally, we propose a heuristic and a Branch-and-Bound algorithm to find good solutions of the problem. We compare their results with the

one of an integer linear programming formulation on cell-like networks, Poisson-Voronoi tessellations.

Many problems remain to be solved:

- The study of the grid graphs, we considered a specific function where vertices at distance one interfere by 1 and vertices at distance two by 1/2. Other weight functions should be considered. e.g. $1/d^2$ or $1/(2^{d-1})$, where d is the distance between vertices.

- Other families of graphs could be considered, for example hypercubes.

- We showed that the Threshold Improper Colouring problem can be transformed into a problem with only two possible weights on the edges 1 and ∞, that is a mix of proper and improper colouring. This simplify the nature of the graph interferences but at the cost of an important increase of instance sizes. We want to further study this. In particular, let $G = (V, E, w)$ be an edge-weighted graph where the weights are all equal to 1 or M. Let G_M be the subgraph of G induced by the edges of weight M; is it true that if $\Delta(G_M) << \Delta(G)$, then $\chi_t(G, w) \leq \chi_t(G) \leq \left\lceil \frac{\Delta(G,w)+1}{t+1} \right\rceil$? A similar result for $L(p, 1)$-labelling [HRS08] suggests it could be true.

Note that the problem can also be solved *algorithmically* for other classes of graphs and for other functions of interference. We started looking in this direction in [ABG+11a]. The problem can be expressed as a linear program and then be solved exactly using solvers such as CPLEX or Glpk[3] for small instances of graphs. For larger instances, we propose a heuristic algorithm inspired by DSATUR [Bré79] but adapted to the specifics of our colouring problem. We used it to derive colouring with few colours for Poisson-Voronoi tessellations as they are good models of antenna networks [BKLZ97, GK00, HAB+09]. We plan to further investigate the algorithmic side of our colouring problem.

[3] http://www.gnu.org/software/glpk/

A.7 Bibliography

[AAG+05] S. Alouf, E. Altman, J. Galtier, J.F. Lalande, and C. Touati. Quasi-optimal bandwidth allocation for multi-spot MFT-DMA satellites. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 560–571. IEEE, 2005.

[ABG+11a] J. Araujo, J.-C. Bermond, F. Giroire, F. Havet, D. Mazauric, and R. Modrzejewski. Weighted Improper Colouring. Research Report RR-7590, INRIA, 2011.

[ABG+11b] Julio Araujo, Jean-Claude Bermond, Frédéric Giroire, Frédéric Havet, Dorian Mazauric, and Remigiusz Modrzejewski. Weighted improper colouring. In Costas Iliopoulos and William Smyth, editors, *Combinatorial Algorithms*, volume 7056 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg, 2011.

[AvHK+07] K.I. Aardal, S.P.M. van Hoesel, A.M.C.A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79–129, 2007.

[BKLZ97] F. Baccelli, M. Klein, M. Lebourges, and S. Zuyev. Stochastic geometry and architecture of communication networks. *Telecom. Systems*, 7(1):209–227, 1997.

[Bré79] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[Bro41] R. L. Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(02):194–197, 1941.

[CCW86] L. J. Cowen, R. H. Cowen, and D. R. Woodall. Defective colorings of graphs in surfaces: Partitions into subgraphs of bounded valency. *Journal of Graph Theory*, 10(2):187–195, 1986.

[CGJ95] L.J. Cowen, W. Goddard, and C.E. Jesurum. Defective coloring revisited. *J. Graph Theory*, 24:205–219, 1995.

[CHS09] R. Correa, F. Havet, and J-S. Sereni. About a Brooks-type theorem for improper colouring. *Australasian Journal of Combinatorics*, 43:219–230, 2009.

[FLM+00] M. Fischetti, C. Lepschy, G. Minerva, G. Romanin-Jacur, and E. Toto. Frequency assignment in mobile radio systems using branch-and-cut techniques. *European Journal of Operational Research*, 123(2):241–255, 2000.

[For87] S. Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2(1):153–174, 1987.

[GK00] P. Gupta and P.R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, 2000.

[HAB+09] M. Haenggi, J.G. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti. Stochastic geometry and random graphs for the analysis and design of wireless networks. *Selected Areas in Communications, IEEE Journal on*, 27(7):1029–1046, 2009.

[HRS08] Frédéric Havet, Bruce Reed, and Jean-Sébastien Sereni. L(2,1)-labelling of graphs. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 621–630, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Lov66] L. Lovász. On decompositions of graphs. *Studia Sci. Math. Hungar.*, 1:273–238, 1966.

[MS03] C. Mannino and A. Sassano. An enumerative algorithm for the frequency assignment problem. *Discrete Applied Mathematics*, 129(1):155–169, 2003.

[Woo90] D. R. Woodall. Improper colorings of graphs. In R. Nelson and R. J. Wilson, editors, *Pitman Res. Notes Math. Ser.*, volume 218, pages 45–63. Longman Scientific and Technical, 1990.

[Yeh06] Roger K. Yeh. A survey on labeling graphs with a condition at distance two. *Discrete Mathematics*, 306(12):1217 – 1231, 2006.