ON THE COST-BENEFIT TRADEOFFS OF CLOUD STORAGE SERVICES FOR END USERS AND SERVICE PROVIDERS

GLAUBER DIAS GONÇALVES

ON THE COST-BENEFIT TRADEOFFS OF CLOUD STORAGE SERVICES FOR END USERS AND SERVICE PROVIDERS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Jussara Marques de Almeida
Coorientador: Alex Borges Vieira

Belo Horizonte

Dezembro de 2017

GLAUBER DIAS GONÇALVES

ON THE COST-BENEFIT TRADEOFFS OF CLOUD STORAGE SERVICES FOR END USERS AND SERVICE PROVIDERS

Thesis presented to the Graduate Program in Ciência da Computação of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Ciência da Computação.

Advisor: Jussara Marques de Almeida
Co-Advisor: Alex Borges Vieira

Belo Horizonte

December 2017

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Gonçalves, Glauber Dias.

G635o On the cost-benefit tradeoffs of cloud storage services for end users and service providers. / Glauber Dias Gonçalves. – Belo Horizonte, 2017.
xiv, 139 f.: il.; 29 cm.

Tese (doutorado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientadora: Jussara Marques de Almeida Gonçalves
Coorientador: Alex Borges Vieira

1. Computação - Teses. 2. Computação em nuvem. 3. Comportamento de usuário. 4. Custos e benefícios. 5. Cache. 6. Dropbox. I. Orientadora. II. Coorientador. III. Título.

CDU 519.6*32(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

On the Cost-Benefit Tradeoffs of Cloud Storage Services for End Users and Service Providers

GLAUBER DIAS GONÇALVES

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

Profa. Jussara Marques de Almeida Gonçalves - Orientadora
Departamento de Ciência da Computação - UFMG

Prof. Alex Borges Vieira - Coorientador
Departamento de Ciência da Computação - UFJF

Prof. Ana Paula Couto da Silva
Departamento de Ciência da Computação - UFMG

Prof. Djamel Fawzi Hadj Sadok
Centro de Informática - UFPE

Prof. Dorgival Olavo Guedes Neto
Departamento de Ciência da Computação - UFMG

Prof. Edmundo Roberto Mauro Madeira
Instituto de Computação - UNICAMP

Prof. Pedro Olmo Stancioli Vaz de Melo
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 18 de Dezembro de 2017.

Abstract

Cloud storage is a data-intensive Internet service that offers the means for end users to easily backup data and perform collaborative work, with files in user devices being automatically synchronized with the cloud. This service is already one of the most popular ones on the Internet, and well-established players, such as Dropbox, Google and Microsoft, face a fierce competition for customers. Despite the popularity of cloud storage, little is known about the implications of user behavior patterns and policies adopted by popular services to the costs and benefits for both end users and service providers. Our research aims at investigating such cost-benefit tradeoffs for both parties jointly. In particular, we develop workload analysis tools and analytic models so to help providers in assessing the effectiveness of alternative policies that aim at increasing both profitability and user satisfaction. To that end, the research is narrowed down into three complementary research goals: (1) to characterize typical user behavior and to model the workload patterns derived from it in cloud storage services; (2) to investigate the impact of content sharing on the costs of service providers and on the level of user activity; and (3) to model cost-benefit tradeoffs between end users and the service provider. We have addressed these issues by investigating the traffic generated by users of Dropbox, which is currently one of the most popular cloud storage services. Using data collected passively from four different networks, we have obtained the following contributions related to each research goals:

(1) We propose a hierarchical model that captures user sessions, file system modifications and content sharing patterns. We parameterize our model using measurements gathered from our datasets. Next, we use the proposed model to drive the development of CloudGen, a new synthetic workload generator that allows the simulation of the network traffic created by cloud storage services in various realistic scenarios. We validate CloudGen by comparing synthetic traces with the actual data. We then show its applicability by investigating the impact of the continuing growth in cloud storage popularity on bandwidth consumption. Our results indicate that a

hypothetical 4-fold increase in both user population and content sharing could lead to 30 times more network traffic.

(2) Using our datasets, we show that a large fraction of downloads generated by Dropbox users (57–70%) is associated with content shared among multiple devices. Moreover, despite optimizations such as the `LAN Sync` protocol, up to 25% of Dropbox download traffic refers to content replicas. We then evaluate an alternative synchronization architecture that uses caches to offload storage servers from such downloads. Our experiments show that the approach cost-effectively avoids most repetitive downloads, benefiting service providers, the network and end users. Moreover, we quantify the relationship between content sharing and user activity levels in Dropbox. We show that the user activity (i.e., data volume synchronized with the cloud) is highly correlated with the amount of shared folders (the measured Spearman correlation coefficient was 0.73-0.98 in the four networks) which indicates the most active users are likely to share more content in Dropbox.

(3) We develop a model to analyze cost-benefit tradeoffs for service providers and end users. Our model is based on utility functions that capture, in an abstract level, the satisfaction of both parties, the service provider and users, in various scenarios scenarios. Then, we apply our model to evaluate alternative policies for content sharing in cloud storage services. We consider two alternative policies for the current services sharing architecture, which count on user collaboration to reduce providers' costs via P2P architecture. Our results show that these policies are advantageous for providers and users, leading to utility improvements of up to 39% for both parties with respect to current settings. Best utility improvements are achieved with a limited number of collaborators (around 30% of the eligible devices). Gains are more relevant in scenarios where users share lots of contents, but those collaborating still need to contribute with only a small amount of resources – e.g., no more than 23% of the Internet bandwidth they are willing to offer to the provider.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In the modern human society, essential supply services such as water, electricity and gas as well as telephony and Internet are frequently accessed. Consumers pay for these public utility services based on their usage or a flat rate tariff. As early as in the 60's, visionary computer scientists such as John McCarthy and Leonard Kleinrock foresaw the potential for computation to also be an essential service for people and thus be provided as a public utility [Buyya et al., 2008; Zhang et al., 2010; Bestavros and Krieger, 2014]. Recently, these predictions are achieved by the concept of cloud computing, which can be summarized as an architecture and business model in which computational resources (e.g., networks, servers, storage, applications, services, etc.) can be rapidly provisioned and released with minimal management effort or service provider interaction [NIST, 2011].

Cloud computing has made software even more attractive. Now, developers with innovative ideas can count on the availability of hardware infrastructure to deploy highly scalable Internet applications. In addition, the "elasticity" of cloud computing allows such applications to increase dynamically hardware resources to serve load surges and release these resources when no more needed. In other words, applications based on the cloud computing model, also called *cloud services*, are delivered over the Internet and supported by an infrastructure of various data centers spread all over the world. Important advantages that cloud services offer to users are high availability and scalability. This means that a cloud service may be available for their subscribers through the Internet at any time and provide computational resources (e.g., processing or storage) as large as subscribers need or can pay for [Armbrust et al., 2010].

Recently, providing online storage for end users (i.e., people or enterprises) has become an increasing popular cloud service [Drago et al., 2012; Bocchi et al.,

2015b]. A typical (and multiple purposes) cloud storage service [Marshall et al., 2012; Gracia-Tinedo et al., 2015] might provide end users with a convenient and reliable way to store files on the cloud (*backup*) as well as synchronize data among multiple devices, such as personal computers (PCs), tablets and smartphones (*file synchronization*). In addition, it might offer the means for a user to easily share files with other users, such that these files are automatically synchronized among all user devices in real time (*collaborative work*).

Cloud storage services currently attract a large interest of industry. The high public success has pushed many providers to this market, and among well-known services we can cite Dropbox, Google Drive, OneDrive, Box, iCloud, SugarSync and Carbonite[1]. However, it is the well-established players, such as Dropbox, and giants like Google and Microsoft that face a fierce competition for customers. A measurement study conducted on European Internet service providers in 2015 [Bocchi et al., 2015b] showed Dropbox was the most popular service, being observed in 68% of monitored households. On the other hand, Google Drive and OneDrive showed a fast growth thanks to solutions which are more and more integrated into PCs or mobile operating systems. As of late 2017, cloud storage services have still shown an increasing popularity. For example, Dropbox claims to have 500 million registered users[2], whereas Google drive have passed this mark with more than 800 million registered users[3].

Cloud storage services have also attracted the attention of researchers, not only due to their high popularity among users but also due to the impact generated on local network traffic [Gonçalves et al., 2014a; Bocchi et al., 2015b]. Among the existing services, Dropbox has been one of the most studied ones and such attention may be explained by two important aspects. First, Dropbox presents the largest number of technical capabilities, e.g., compression, deduplication, device-to-device synchronization [Li et al., 2014; Bocchi et al., 2015a]. Second, Dropbox has a large insertion in academic environments, as observed in traffic measurement studies. For example, the total volume of Dropbox traffic was equivalent to one third of the YouTube traffic on European universities networks [Drago et al., 2012], whereas, Dropbox was estimated at 4% of all traffic of a university in Brazil [Duarte et al., 2015] and 71% of all access to storage servers from a university in Portugal [Oliveira et al., 2015].

[1] See an exhaustive list of providers and their web pages in `https://en.wikipedia.org/wiki/Comparison_of_online_backup_services`

[2] `https://www.dropbox.com/news/company-info`

[3] `https://www.youtube.com/watch?v=Y2VF8tmLFHw`

1.1 Motivation

Despite such popularity of cloud storage services, little is known about the underlying client processes that generate workload to these services. As mentioned previously, a typical cloud storage service offers users multiple functionalities such as backup, file synchronization and collaborative work. These functionalities bring new issues for the existing workload models based on well-known Internet services, such as e-commerce [Krishnamurthy et al., 2006], video streaming [Borges et al., 2012] and online social networks [Benevenuto et al., 2012]. Unlike observed for those other types of services, content stored in cloud storage is often modified by one or more users and replicated among their devices as fast as possible to reflect the user interaction in file synchronization or collaboration networks. Previous studies of user behavior in cloud storage services focused mostly on identifying performance bottlenecks as well as proposing benchmarks [Drago et al., 2012; Li et al., 2014; Gracia-Tinedo et al., 2015; Bocchi et al., 2015b; Liu et al., 2013]. Modeling the workload patterns derived from user behavior, which is key to analyze system performance, costs as well as the future user satisfaction for these services, has not been tackled yet.

Another important aspect of cloud storage services regards cost-effective architectures to support file synchronization and collaborative work. Studies conducted by computer-supported cooperative work (CSCW) community provide evidence that functionalities for sharing content (social file sharing) increases user activity [Shami et al., 2011; Marshall et al., 2012]. In fact, cloud storage providers foster content sharing functionalities as they might attract users to the service as well as increasing their data volume in the cloud. On the other hand, such functionalities pose extra costs to providers, as more data transference to/from the cloud will be required to synchronize content shared among multiple user devices. Although new architectural designs to optimize data synchronization with the cloud have been proposed [Vrable et al., 2012; Bessani et al., 2014; Li et al., 2014], none of them evaluates the impact of collaborative work on cloud storage service costs and on user behavior (i.e., activity) patterns. Dropbox has employed the `LAN Sync` protocol aiming to reduce content sharing traffic with device-to-device synchronization [Dee, 2015]. Nevertheless, there is still a lack of studies evaluating how effective this protocol really is. These are important issues to be investigated in cloud storage services, since cost-effective architectures for file synchronization and collaborative work have the potential to benefit not only service providers, but also local networks and end users.

Finally, cloud storage services have also an economic dimension which encompasses the pricing/incentive strategies adopted by the service, the resource

demands imposed by the users when interacting with the service, and the competition among multiple providers for attracting users. These aspects raise an interesting issue related to the tradeoffs between costs and benefits of cloud storage services for both providers and end users, given current pricing models adopted by popular services. For example, most Dropbox users use a free version of the service with limited storage, while the service offers an option to pay for extra space if desired[4]. Although attractive to users, such pricing models pressure providers to offer more competitive services. In fact, some providers have left the market (e.g., UbuntuOne, Wualla), possibly, due to high costs of outsourcing infrastructure [Silber, 2015; Lam, 2015]. Most existing studies analyze this issue from the perspective of either the user [Naldi and Mastroeni, 2013; Naldi, 2014; Yeo et al., 2014; Shin et al., 2014] or the provider [Wu et al., 2013; Wang et al., 2012], but not both, thus offering a limited interpretation. A prior effort that jointly analyzed user and provider perspectives [Lin and Tzeng, 2014] does not take into account typical user behavior patterns and the roles of file synchronization and collaborative work, which are key components to a cost-benefits analysis.

All the aforementioned issues related to cloud storage services have as central pillar the common patterns of user behavior. In this dissertation, we are specially interested in exploiting the impact of this behavior on costs and benefits of this type of service for both users and providers. Our assumption is that a solid understanding of such patterns can guide the design of solutions to improve services in terms of architecture and storage/data transference costs to better meet user satisfaction. Towards building such understanding, we rely on a rich set of data provided by cloud storage services capturing user behavior patterns ranging from *technical features* (e.g., sessions durations, volume of data transferences, file inter-modification time) to *social features* (e.g., number of shared folders, number of contacts in collaborative work[5], and number of devices associated with shared folders).

In sum, the guiding question we evaluate throughout this dissertation is: given a set of features that represent user behavior in cloud storage services – particularly technical features and social features – can service providers leverage such features to decrease their cost (or, in other words, improve their profit) while still improving users satisfaction (e.g., increasing users benefit in the service)?

In the following section, we narrow this guiding question down as a problem statement and specific research goals.

[4] `http://www.economist.com/blogs/babbage/2012/12/dropbox`

[5] The number of contacts of a given user may be obtained by counting all distinct users associated with its shared folders.

1.2 Problem Statement

The problem we propose to tackle is the following: given a set of users U and a cloud storage service provider p that employs a set of policies, such as variable pricing/incentive strategies (e.g., free service up to X bytes, s dollars from X to Y bytes, no transference limit for file synchronization and collaborative work among users, etc), how should this provider assess the effectiveness of policies that aim at increasing both profitability and user's satisfaction?

To answer this question, we intend to (1) characterize and model typical user behavior patterns and the associated workload; (2) understand the impact/advantages of user social features (e.g., number of shared folders per user, number of devices associated with a shared folder) on/to service; and (3) analyze the cost-benefit tradeoffs to provide the service while meeting the interests of both users and providers. In this way, our goal throughout this dissertation is to develop new models and tools to tackle these three issues. In other words, we shall develop models that will take into consideration realistic workloads to support providers in choosing the most cost-effective policies.

1.3 Research Goals

The general goal described above is narrowed down into the following three complementary research goals (RGs):

RG1 - To Model User Behavior and Workload Patterns: Our first research goal focuses on characterizing and modeling typical user behavior (and the workload patterns derived from it) in cloud storage services. Specifically, we aim at modeling user behavior with respect to technical features which regard user sessions, content updates within sessions, and volume of data transmitted (upload/download) over the Internet. We also intend to model social features which emerge from the *content propagation network* during file synchronization and collaborative work as shared folders and user devices associated with shared folders. The last step towards achieving this RG consists in implementing a synthetic workload generator based on the proposed model, thus capturing, realistically, the aforementioned aspects of user behavior. We believe this workload generator will be a valuable tool to simulate realistic scenarios, providing, for example, reasonable estimates of resource (storage/data transference) demands for a provider given a target number of users. As such, it will also be used to drive our steps towards achieving RG2 and RG3.

RG2 - To analyze the Effects of Content Sharing: As our second goal, we aim at deepening our investigation on service functionalities related to content sharing. The key question we intend to address in RG2 is: *to which extent file synchronization and collaborative work in cloud storage impacts on the user activity and the costs of service providers (i.e., resource consumption)?* Recent studies have already observed that downloads due to file synchronization and collaborative work account for higher traffic volume than uploads in cloud storage [Bocchi et al., 2015b; Gracia-Tinedo et al., 2015]. However, most providers do not implement any distributed synchronization architecture to reduce data transference cost due to download traffic [Bocchi et al., 2015a]. Towards our RG2, we intend to investigate whether the traffic caused by content sharing is significant for providers and, if so, how cost-effective an alternative architecture that take advantage of content sharing to save data transference to/from the cloud can be. We also intend to investigate the correlation between social features and the level of user activity given by some technical features (e.g., volume of content synchronized with the cloud per user).

RG3 - To Model the Tradeoffs Between User and Service Provider Interests: Building on the conclusions of RG1 and RG2, we will analyze the cost-benefit tradeoffs which result from the interplay between end users and the service provider. Given the problem statement of this dissertation (*how should providers assess the effectiveness of policies that aim at increasing both profitability and user's satisfaction?*), we intend to develop analytic models, specifically utility functions [Lin and Tzeng, 2014; Shen and Li, 2015], which capture the costs and the benefits of cloud storage services for both parties. For example, the provider's utility is a function of its revenue and cost, which may depend on the percentage of paying users, funding groups and all users resource demands. The user utility, on the other hand, captures the user satisfaction with the service, and may be expressed as a function of technical features and/or social features and the service cost. Such models will enable the analysis of the provider and users' utility improvements over different service policies and realistic scenarios.

1.4 Contributions and Outline of this Dissertation

In Figure 1.1 we show how the research goals of this dissertation relate to each other. We also emphasize the chapters in which each research goal is addressed. The rest of this dissertation is organized as follows. Chapter 2 provides a background required for understanding the context of the dissertation. Chapter 3 discusses related work that

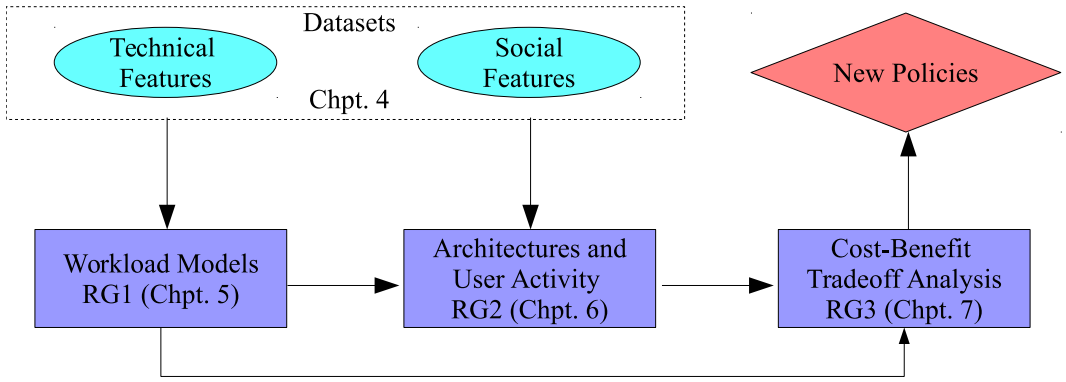


Figure 1.1: Pictorial representation of our Research Goals.

inspired our research goals. Our expected contributions are organized in the following chapters:

- **Chapter 4** focuses on technical issues to collect data from cloud storage services. The development of the proposed RGs requires the analysis of real data so as to produce realistic representations. Thus, data collection from real cloud storage services is a key step in our research. As the stored content is private and synchronization protocols are mostly proprietary, the current knowledge of how cloud storage services work is limited, which makes data collection very challenging. Yet, Drago et al. [2012] developed methods to obtain data of Dropbox, which is one of the currently most popular cloud storage service, from traffic measurements. Such methods respect the privacy of users and services, as data is collected passively and remains anonymized. In this dissertation, we improve this method by proposing a post-processing data methodology, that estimates the volume of updates made on content stored by users on Dropbox folders from TCP flows. We use data related to Dropbox traffic collected in four large networks.

- **Chapter 5** regards our study on **RG1**. Specifically, we show our proposal for a novel hierarchical model of user behavior in cloud storage services. This model is based on a thorough investigation of the Dropbox datasets described in Chapter 4, however, focusing on actions driven by user behavior. Thus, the model provides realistic representations of the workload generated by general users of cloud storage services, specifically capturing user technical and social features. We use the proposed model to drive the development of CloudGen, a new synthetic workload generator that allows the simulation of the network traffic created by cloud storage services in various realistic scenarios. We validate CloudGen by

comparing synthetic traces with actual data from operational networks. We then show its applicability by investigating the impact of the continuing growth in cloud storage popularity on data transference consumption.

- **Chapter 6** regards our study on **RG2**. We first tackle the issue of how content sharing impacts the costs of a service provider. Cloud storage offers a popular environment for users to store and share content. Yet, content sharing by multiple users located within the same network leads to multiple downloads of a single content for synchronizing devices. These downloads create extra load on the servers. We investigate the traffic generated by Dropbox and the fraction of downloads associated with content shared among multiple devices located within the same network, using the datasets described in Chapter 4. Moreover, we evaluate the feasibility of an alternative synchronization architecture that uses caches to offload storage servers from such downloads. As a final part of RG2, we investigate to which extent the social features related to content sharing contribute to increase user activity. We believe this investigation is important as a growing level of user activity might yield increased service provider profit, then compensating the costs associated with content sharing. In this sense, we analyze the relationship between social features (e.g., number of shared folders, devices, contacts) and the activity level of users given by technical features (e.g., volume of content synchronized by users with the cloud).

- **Chapter 7** regards our study on **RG3**. We build upon our RG1 and RG2 efforts, specifically user technical/social features and alternative synchronization architectures, to propose a general model for the costs and benefits of cloud storage considering both users and providers. Our model is based on utility functions that capture, in an abstract level, the satisfaction of the service provider and users, i.e., the benefits minus the costs of the service for each party. The greater the utilities are, the more satisfied providers and users become. Then, we apply the model to evaluate alternative policies for content sharing in cloud storage. We consider alternative policies for the current service sharing architecture, which count on user collaboration to reduce providers' costs via a Peer-to-Peer (P2P) architecture. Ultimately, the proposed model allows us assessing the effectiveness of those service alternative policies in order to improve both the provider and users' utilities in various realistic scenarios.

Finally, Chapter 8 concludes this dissertation and presents a discussion on directions for future work.

1.5 Overview of Current Results

Our current results are summarized in the following publications:

1. Gonçalves, G., Drago, I., Couto da Silva, A. P., Borges Vieira, A., & Almeida, J. M. (2014). Modeling the Dropbox client behavior. In IEEE International Conference on Communications - ICC **(RG1)**.

2. Gonçalves, G., Drago, I., Couto da Silva, A. P., Borges Vieira, A., & Almeida, J. M. (2014) Caracterização e Modelagem da Carga de Trabalho do Dropbox. In 32^o Brazilian Symposium on Networks and Distributed Systems - SBRC **(RG1)**.

3. Gonçalves, G., Drago, I., Couto da Silva, A. P., Borges Vieira, A., & Almeida, J. M. (2015) Analisando o Impacto de Compartilhamentos no Dropbox em uma Rede Acadêmica. In 33^o Brazilian Symposium on Networks and Distributed Systems - SBRC **(RG2)**.

4. Gonçalves, G., Drago, I., Couto da Silva, A. P., Borges Vieira, A., Almeida, J. M., & Melia M. (2016). Workload Models and Performance Evaluation of Cloud Storage Services. Elsevier Computer Networks **(RG1)**.

5. Gonçalves, G., Couto da Silva, A. P., Borges Vieira, A., & Almeida, J. M. (2016) Trabalho Colaborativo em Serviços de Armazenamento na Nuvem: Uma Análise do Dropbox. In 34^o Brazilian Symposium on Networks and Distributed Systems - SBRC. **(RG2)**

6. Gonçalves, G., Drago, I., Couto da Silva, A. P., Borges Vieira, A., & Almeida, J. M. (2016). On the Impact of Content Sharing on the Bandwidth Consumption of Cloud Storage. IEEE Internet Computing. **(RG2)**

7. Gonçalves, G., Drago, I., Couto da Silva, A. P., Borges Vieira, A., & Almeida, J. M. (2016). Analyzing Costs and Benefits of Content Sharing in Cloud Storage. In ACM SIGCOMM Workshop on Fostering Latin-American Research in Data Communication Networks (LANCOMM 2016). **(RG3)**

8. Gonçalves, G., Drago, I., Couto da Silva, A. P., Borges Vieira, A., & Almeida, J. M. (2017). Cost-Benefit Tradeoffs of Content Sharing in Personal Cloud Storage. In IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2017). **(RG3)**

Chapter 2

Background

In this chapter, we discuss core aspects of the architecture of cloud storage services providing the basis to model user behavior and workload patterns. We first introduce the concepts of cloud computing and cloud services according to the technical literature in Section 2.1. In Section 2.2, we provide an overview of the typical architecture of cloud storage services. Finally, in Section 2.3, we focus on a popular service of this kind (Dropbox), to report the main architectural aspects of a typical cloud storage service. As such, these aspects will be used to drive the methodology we adopt to collect cloud storage traffic data, presented in the next chapter.

2.1 Cloud Computing and Cloud Services

Cloud computing is typically defined as an architecture and business model, both at the hardware and the software levels, which provide computing resources, e.g., processing, memory, storage and applications, delivered over the Internet [Armbrust et al., 2010]. These resources can be rapidly provisioned and released with minimal management efforts or interaction between resource provider and consumer [NIST, 2011].

One of the main purposes behind cloud computing is to provide computing resources for the general public[1] through a utility business model like water or energy [Buyya et al., 2008]. The term "cloud computing" started to gain popularity mainly from 2006 after being used by the Google CEO Eric Schimit to

[1] Although computing resources can also be provided for a specific group of users, i.e., a private cloud.

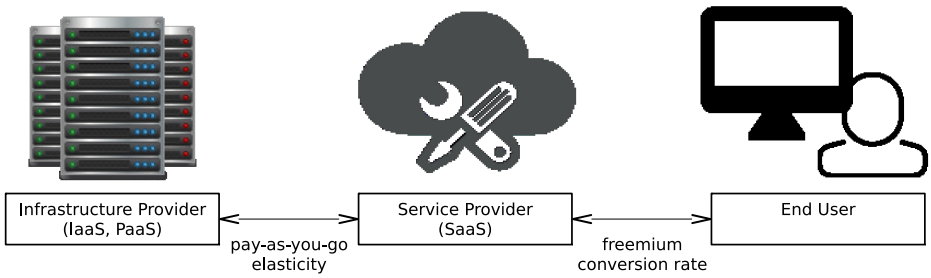


Figure 2.1: Representation of the relationship between providers and consumers in the cloud computing model.

describe the business model of providing the aforementioned resources through the Internet [Zhang et al., 2010]. Since then, cloud computing has consolidated thanks to the development of technologies such as grid computing [Chetty and Buyya, 2002] and virtualization [Graupner et al., 2003]. The former is a distributed computing paradigm for coordinating networked resources in order to offer protocols that enable shared computation or storage over long distances. The latter abstracts the details of physical hardware providing resources for high-level applications. Ultimately, cloud computing leverages both technologies to achieve the goal of providing computing resources as a utility.

The cloud computing business model provides software and hardware as services which are available on demand, also called cloud services. These services can be grouped into three categories: infrastructure as a service (IaaS), e.g., Amazon EC2 or RackSpace[2]; platform as a service (PaaS), which refers to lower level software resources such as development frameworks (e.g., Google App Engine[3]) or a framework for reselling infrastructure resources (e.g., VMs) at lower cost [Vieira et al., 2013]; and software as a service (SaaS), which is a higher level software. Examples of SaaS are cloud storage services such as Dropbox or Google Drive[4], although traditional business applications have also adopted SaaS through cloud computing model, for example SAP Business byDesign[5] and Salesforce.com[6].

The relationship between provider and consumer of IaaS, PaaS and SaaS is not often clearly defined. Nevertheless, most of the researchers adopt the representation shown in Figure 2.1 [Armbrust et al., 2010; Zhang et al., 2010]. For the sake of simplicity, we will adopt the terms “infrastructure provider” and “service provider”

[2] `http://aws.amazon.com/ec2`, `http://www.rackspace.com/cloud/servers`
[3] `https://cloud.google.com/appengine/docs`
[4] `https://www.dropbox.com/about`, `https://www.google.com/drive`
[5] `http://go.sap.com/product/enterprise-management/business-bydesign.html`
[6] `https://www.salesforce.com/blog/2013/01/what-is-crm-your-business-nerve-center.html`

as shown in this figure. Note that IaaS and PaaS providers are often parts of the same organization (e.g., Google and Google App Engine respectively) or the PaaS focuses on providing infrastructure resources [Vieira et al., 2013]. As such, they can be called the infrastructure providers (or cloud providers). People or organizations that use this infrastructure via utility computing to provide SaaS are called service providers (or cloud consumers). In some cases, an infrastructure provider might also host its own customer service, e.g., Google and Google Drive respectively. At its rightmost side, Figure 2.1 shows end users that use cloud services via Web interfaces or client applications.

Most of the research taking place in the technical aspects of cloud computing considers one of two (or both) types of relationships shown in Figure 2.1. The relationship between infrastructure and service providers is mainly characterized by the dynamic resource provisioning and short-term payment [Zhang et al., 2014a; Wang et al., 2014; Vieira et al., 2014]. This business model, well-known as *pay-as-you-go* [Armbrust et al., 2010], allows cloud consumers to use computing resources on demand as needed to follow their load surges (infinite resources appearance), and release them when no more needed. Those resources are typically paid on a short-term basis, e.g., processors per hour, storage/data transference per used volume. Alternatively, consumers can rent resources on a long-term basis (e.g., physical or virtual machines per years), thus reducing per-usage charges[7].

According to Armbrust et al. [2010], one of the main advantages of the "pay-as-you-go" model is resources "elasticity", i.e., add/remove resources for any computing task quickly (within minutes). This cloud computing ability enables consumers to match resources closely to their service workload and to increase profitability. In other words, elasticity can avoid service's potential revenue loss, when users are not served during a peak load (underprovisioning) as well as resources waste during non peak times (overprovisioning).

The relationship between service providers and end users follows a different dynamic. In general, providers offer a free service version with limited resources, expecting that a fraction of their registered users will pay for the full service. This business model well-known as "freemium", i.e., a combination of "free" and "premium", has become the dominant one among service providers, in particular, Internet start-ups and smartphone applications [Kumar, 2014]. In this model, free users get basic functionalities at no cost and no limited-term offers[8], and can access

[7] We discuss different payment models for cloud resources in Chapter 3 Section 3.3.1.

[8] Note that the freemium model is different from the 30-day free trials, which customers have become wary of cumbersome cancellation process.

richer functionalities for a subscription fee, becoming a premium user. A monthly or yearly fixed fee is the common price adopted by the majority of providers for premium users [Naldi and Mastroeni, 2013].

According to Kumar [2014], one of the main advantages of the freemium model is to allow new companies to increase their user base without expending resources on costly ad campaigns or a traditional sales force. On the other hand, the service provider has to maintain its *conversion rate*, i.e., the percentage of free users that have upgraded to a premium plan, at a moderate level. The authors' research shows that successful companies cases, which adopt freemium model targeting a large market (millions of registered users), reach a conversion rate between 2-5%, thus being considered a moderate rate. Moreover, freemium companies may face fall and rise of their conversion rate as they expand the user base to include people that are more price-sensitive, i.e., users who see less value in the service.

In this dissertation, we study the relationship between service providers and end users. In particular, we focus on cloud storage services, which is one type of cloud service that has attracted a lot of interest among end users [Drago et al., 2012; Bocchi et al., 2015b], thus being one of the most representative examples of such relationship [Shin et al., 2014]. Moreover, cloud storage services provide us means to study three important aspects of the relationship between service providers and end users: workload patterns, distributed architectures for file synchronization and sharing, and cost-benefit tradeoffs of cloud service. We have appointed these aspects as guiding motivations for our research goals in Chapter 1.

2.2 Cloud Storage Services Typical Architecture

The main functionalities that users expect from a cloud storage service, according to researches performed by the human-computer interaction community [Marshall et al., 2012; Voida et al., 2013; Massey et al., 2014], are: backup (copying data to secondary storage, e.g., cloud data centers), file synchronization (sharing data with oneself on different devices), and collaborative work (sharing data with others). In this section, we discuss common architectural aspects adopted by most cloud storage services to offer these three functionalities. Our observations are supported by studies that dissect internal designs of popular services such as Dropbox, Google Drive, MS OneDrive [Wang et al., 2012; Drago et al., 2012; Bocchi et al., 2015a], and also an

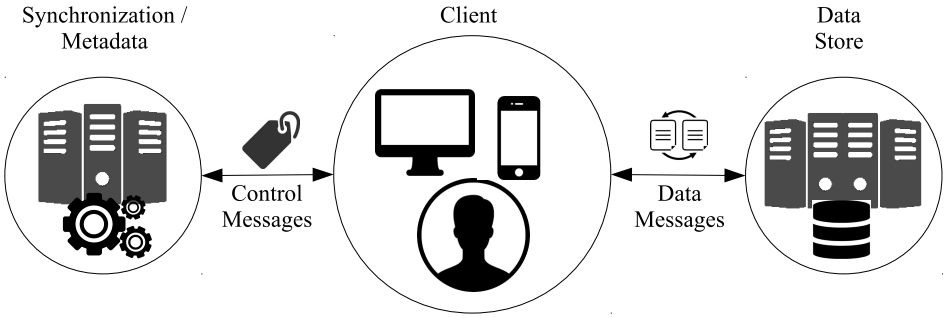


Figure 2.2: Architectural design of a typical cloud storage service.

open source service, less popular, named UbuntuOne [Gracia-Tinedo et al., 2015]. All of them offer the three mentioned functionalities.

The architectural design of a typical cloud storage service presents three components: (i) clients, (ii) synchronization or metadata service, and (iii) data store [Drago et al., 2012; Gracia-Tinedo et al., 2015]. Figure 2.2 presents an overview of this architecture. *Clients* play a central role in this architecture as they generate the workload for the service. This component is represented by PC or mobile devices that use the service via client applications or Web interfaces developed by the providers. The *Synchronization* component, at the left side of Figure 2.2, typically consists of processing tasks related to the management of user files and their logical representation (meta-data). At the right side, Figure 2.2 shows the infrastructure of data centers representing the *data store* component. This infrastructure can easily scale out storage capacity thanks to the "pay-as-you-go" cloud business model, then reducing the costly risks of storage resources underutilization or saturation.

Some service providers, e.g., Dropbox and UbuntuOne, handle their own infrastructure only for the synchronization component, whereas user files may be stored separately in infrastructure providers like Amazon (i.e., an outsourced data store component). Wang et al. [2012] observed that Dropbox used the Amazon infrastructure to also perform processing tasks (Amazon EC2 service) such as file hash keys computations, before sending these files to the outsourced storage (Amazon S3 service). At the time of this writing, Dropbox is moving to its custom-built infrastructure [Akhil Gupta, 2016], whereas service providers like Microsoft and Google already take advantage of their own infrastructure of data centers to host the synchronization and data store components of their end-user storage services, respectively, OneDrive and Google Drive.

Figure 2.2 also highlights out the communication among the components. It focuses on two main types of messages: *control* and *data* messages. Services

typically adopt HTTPS to encrypt all communication among components for both types. Control messages are established between a client and the synchronization components for tasks such as authentication, meta-data control, and notification of file modifications. The majority of services maintains an open TCP connection between client terminals and synchronization servers, in which file modifications are announced by clients or servers as they occur via the open connection. In addition, keep-alive messages are exchanged periodically only to guarantee that the notification channel remains open.

In turn, data messages transmission, established between client terminals and storage servers, can be optimized by specific techniques as those listed by Bocchi et al. [2015a]: (i) chunking, i.e., splitting large files into a maximum size data unit; (ii) bundling, i.e., the transmission of multiple small files as a single object; (iii) deduplication, i.e., avoiding re-transmitting content already available on servers; (iv) delta encoding, i.e., transmitting only modified portions of files; (v) compression, i.e., compressing files before transmission; and (vi) device-to-device (or peer-to-peer) synchronization, i.e., exchange files among devices without involving the data store component. Dropbox is the only service that applies all these techniques, according to benchmarks provided by Bocchi et al. [2015a].

Most of the techniques aforementioned, i.e., items i-v, focus on improving performance of data transference between users devices and the cloud. Previous studies [Drago et al., 2012; Li et al., 2014; Bocchi et al., 2015a] show such techniques reached relevant results. For example, Li et al. [2014] conducted active measurements in five popular services and observed that compression is able to reduce 24% of that traffic, whereas Drago et al. [2012] observed bundling deployment in Dropbox reduced such traffic overhead and, as a consequence, increased mean throughput by 65%. On the other hand, only one technique (device to device synchronization) addressed the traffic due to content sharing among devices in cloud storage services. To our knowledge, no previous studies tackled the performance of this technique via traffic measurement of real services. Thus, we consider this later our line of optimization for data transference, as we are going to show in Chapters 6 and 7.

Finally, it is worth mentioning the impact of the geographic locations of the data store component on service performance. Measurement studies conducted by Gracia-Tinedo et al. [2013] show that location greatly impacts data transmission speed from/to the cloud. It means that the farther the end user is from the data store, the longer the data transmission period will be. Amongst popular services, Google Drive is the only service in which user data transmissions are redirected to a (possibly) nearer data store, thanks to Google's content delivery network (CDN) infrastructure.

Dropbox, in contrast, used to redirect all worldwide user data transmissions to Amazon's data centers in the United States[9]. In this case, this service applies the aforementioned data transmission optimizations to overcome issues as long TCP Round Trip Times (RTT) due to larger distances between users and data store.

2.3 Dropbox Overview

In this section, we focus on a popular service, namely Dropbox, to report the main architectural aspects of a typical cloud storage service. First, we overview Dropbox operation by summarizing its architecture (Section 2.3.1), and next, we discuss its file synchronization protocols (Section 2.3.2).

2.3.1 Architecture

Dropbox relies on thousands of servers that are split into two major groups: *control servers* (i.e., synchronization component) and *storage servers* (i.e., data store component). Control servers are responsible for: (i) user authentication; (ii) management of the Dropbox file system metadata; and (iii) management of device sessions. Storage servers are in charge of the actual storage of data[10].

Users can link multiple devices to Dropbox using both PC and mobile clients. Web interfaces are also available, providing an easy way to access files in the cloud. We will only discuss PC client in this section, since it was responsible for more than 75% of the Dropbox traffic in 2014 [Bocchi et al., 2015b][11]. Every user registered with Dropbox is assigned a unique *user ID*. Every time Dropbox is installed on a PC, it provides the PC a unique *device ID*. Users need to select an initial folder in their PCs from where files are kept synchronized with the remote Dropbox servers. Users might decide to share content with other users. This is achieved by selecting a particular folder that is shown to all participants in the sharing as a *shared folder*.

Internally, Dropbox treats the initial folders selected at configuration time and all shared folders indistinctly. Both initial folders and shared folders are the root of independent directory trees, where actual files are stored, and are called *namespaces* in the Dropbox system. Each namespace is identified by a unique *namespace ID*. Namespaces are usually synchronized with all devices of the user and, for shared folders, with all devices of all users participating in the sharing. Each namespace is also

[9] By the time that our datasets were collected.

[10] This component was outsourced to the Amazon cloud at the time of our data collection.

[11] There are some differences when considering different types of clients, which may impact workload modeling, we discuss them in Chapter 5.

attached to a monotonically increasing *journal ID (JID)*, representing the namespace latest version [Koorapati, 2014].

We define an *update* as the steps that a device needs to take to move a namespace from a JID to its next JID value. Updates include files that have been added to/modified in the namespace and all metadata and commands that manipulate the namespace, such as to delete files and create folders.

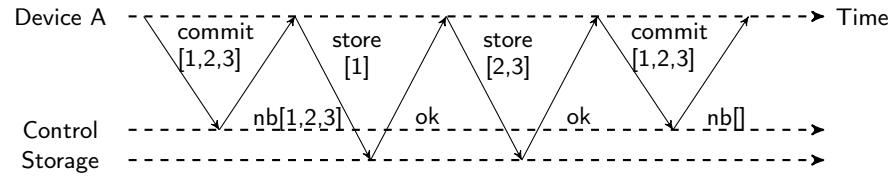
Devices using the Dropbox PC client usually keep a local copy of all files present in the user's namespaces. The addition of any content in a namespace triggers content propagation: all devices having the PC client and registering the namespace retrieve the content immediately if on-line, or as soon as they come back on-line. Dropbox controls the status of namespaces by means of synchronization protocols as we describe next.

2.3.2 File Synchronization Protocols

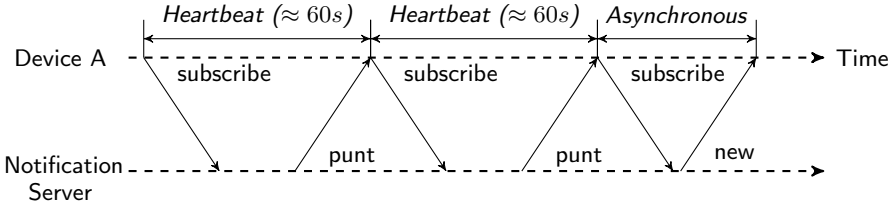
Our work takes as reference the protocols used by Dropbox in its client v2.8.4. A device connecting to Dropbox contacts control servers to perform user authentication and to send its list of namespaces and respective journal IDs. This step allows servers to decide whether the device is updated or if it needs to be synchronized. If outdated, the device executes storage operations (exemplified next), until it becomes synchronized with the cloud. After that, the device finishes the initial synchronization phase and enters the steady state.

When a Dropbox device identifies new local modifications (e.g., new files, or file modifications), it starts a synchronization cycle. The device first performs several local operations to prepare files for synchronization. For example, Dropbox splits files into chunks of up to 4 MB, and calculates content hashes for each of them. Chunks are compressed locally, and they might be grouped into bundles to improve performance. The Dropbox client uses both single chunk and bundles of chunks (i.e., bundling) transferences. The events that trigger a unique or both transferences mode in the client are not clear [Drago et al., 2012]. Such detail, however, does not affect our methodology to collect data from the service, as described in Chapter 4.

As soon as chunks/bundles are ready for transmission, the device behaves as depicted in Figure 2.3a. It first contacts a Dropbox control node, sending a list of new chunk hashes – see the *commit* request in the left-hand side of Figure 2.3a. The servers answer with the subset of hashes that is not yet uploaded (see *need blocks (nb)* reply). Note that servers can skip requesting chunks they already have, a mechanism known as client-side deduplication [Mulazzani et al., 2011]. If any hash is unknown to servers, the device executes several storage operations directly with the storage servers (see



(a) Control and Storage Protocols



(b) Notification Protocol

Figure 2.3: Basic operation of Dropbox client protocols.

store requests). Finally, the device contacts again the Dropbox control node, repeating the initial procedure to complete the transaction.

Since the login and until going off-line, devices keep open a connection with a Dropbox control server, which we call *notification* server. Notification servers are in charge of spreading information about updates performed in namespaces. For instance, when a partner performs modifications in a shared folder, notifications of updates are sent to all devices participating in the sharing. Devices then contact the Dropbox storage and control centers to retrieve updates.

The notification protocol is illustrated in Figure 2.3b. As soon as a device connects to Dropbox, it contacts a notification server (see *subscribe* request). Contrary to other Dropbox communications that have always been performed over SSL/TLS, the traffic to notification servers used to be exchanged in plain text HTTP during the period our datasets were collected (see Chapter 4)[12]. Every message sent to the notification servers contains a device ID and a user ID, as well as the full list of namespaces in the client with their respective journal IDs.

Notification servers answer after approximately 60 seconds if no modifications in the namespaces are performed in the meanwhile (see *punt* reply). The arrival of an answer triggers the client to resend a *subscribe* request, thus acting like a heartbeat protocol. If any modifications are detected in a namespace, the servers instead answer to the pending request asynchronously, with a *new* message. The client then reacts to the notification, starting a new synchronization, as illustrated in Figure 2.3a.

We note that by observing traffic to notification servers, it is possible to identify when each namespace is updated. In this dissertation, a key aspect of our data

[12] Dropbox have encrypted all traffic since mid-2014.

collection methodology relies on exploring these messages to trace the evolution of namespaces of Dropbox clients and reconstruct client behavior.

2.3.3 LAN Sync Protocol

Dropbox deploys a protocol for synchronizing files among devices in the same Local Area Network (LAN), called `LAN Sync` [Dee, 2015]. Devices connected to the same LAN can use the `LAN Sync` protocol to exchange shared files instead of retrieving them from the cloud, thus saving both network and server resources.

The protocol operation is twofold: (i) UDP broadcasts are sent out periodically by all devices in the LAN. The broadcast messages contain information about all namespaces in the devices. Any devices in the vicinity can then form a list of possible neighbors devices for future synchronizations. (ii) Devices receiving a notification of update in a namespace (see the notification protocol in previous section) check their lists of neighbors before starting a synchronization cycle with the cloud. Device to device synchronization takes place if the namespace with new updates is already updated in any other devices in the LAN.

Data synchronization in `LAN Sync` is accomplished over HTTPS. Dropbox makes sure that only clients authenticated for each namespace can request data blocks to each other. Then, Dropbox generates a key/certificate pair for each namespace, and devices receive these pairs when they login to Dropbox. Figure 2.4 illustrates data synchronization between two devices using `LAN Sync`. When a device requests blocks of a namespace to any device of its list, both devices have to establish a connection authenticated with the same certificate of the required namespace (see authentication (auth) for namespace (NM) X). If the authentication is succeeded, the method *head* can be used for checking if the required block exists, whereas the method *get* actually retrieves the block (see *head*, *get* for blocks 1 and 2 and their respective replies).

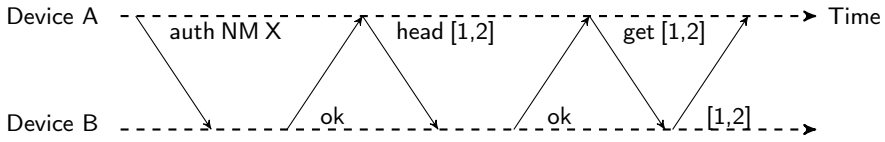


Figure 2.4: Device to device data synchronization with the `LAN Sync` protocol.

Devices with several namespaces in common, potentially from different users, can make use of `LAN Sync` to download files among them in the same LAN, saving time and data transference compared to downloading these files from Dropbox storage servers. However, this protocol works only for devices in the same LAN sub-network

(i.e., same broadcast domain[13]), which is not a common scenario in large networks (e.g., a university campus). Moreover, devices must be on-line simultaneously to benefit from `LAN Sync`. We will return to discuss this protocol in Chapter 6.

2.4 Summary

In this chapter, we presented the background knowledge required to understand the work developed in the following chapters. We discussed the fundamentals of cloud storage services grouped into three key parts: (1) we defined the general concept of cloud service based on cloud computing model and enforcing the relationship between the service provider and the infrastructure provider and between the service provider and the end user (see Figure 2.1); (2) we described the typical architecture of a multi purpose cloud storage service regarding its main components (user, metadata and data store) and types of communication (control and data messages); and (3) we explained the architecture internals of a real cloud storage service (Dropbox), focusing on core aspects of its communication protocols (Section 2.3).

These three aspects we have discussed in this chapter provide the basis to understand our methodology to collect data in next chapter and our workload model presented in Chapter 5.

[13]By the time of writing.

Chapter 3

Related Work

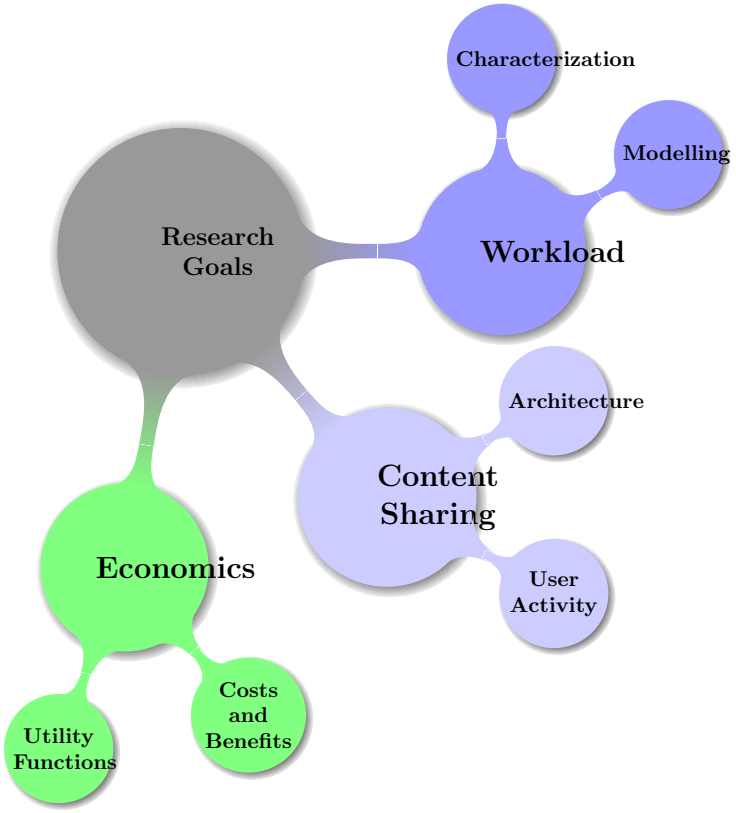


Figure 3.1: Mind mapping of the topics related to this dissertation.

In this chapter, we present a discussion of related work that is fundamental to the understanding of this dissertation. We discuss previous efforts related to each of the three research goals presented in Chapter 1, which can be grouped into the topics shown in Figure 3.1. In Section 3.1, we discuss studies related to the characterization and modeling of cloud storage service workloads. Afterwards, in Section 3.2, we shift

our focus to studies on content sharing in cloud storage regarding architectural designs and measurements of its effects on user activity. Finally, in Section 3.3, we discuss previous efforts to analyze cost-benefit tradeoffs and utility functions considering service providers and end users.

3.1 Services Workloads

In this section, we review related work regarding the first research goal (RG1) of this dissertation which focuses on the workload patterns of cloud storage services. We start by discussing previous characterizations and performance analysis efforts in Section 3.1.1. We then review previous efforts to model the workload of general Internet services which are closely related to cloud storage in Section 3.1.2.

3.1.1 Cloud Storage Services Characterization and Performance Analysis

Cloud storage services have received large attention from the research community. Drago et al. [2012] were the first to present an in-depth characterization of typical usage and performance bottlenecks of these services. The authors studied Dropbox architecture internals and developed a methodology to collect data of this service from traffic measurements in edge networks, e.g., large networks maintained by an organization, such as an Internet service provider (ISP) or a university. The proposed methodology respects privacy concerns of users and services since data is collected passively and remains anonymized. In this dissertation, we partly rely on their methodology to collect and characterize data from Dropbox. However, our main goal is to model user behavior. In this sense, we build upon the methodology developed in the work of Drago et al. [2012] by proposing a post-processing data mechanism to estimate the volume of content modifications (see Chapter 4). This mechanism provided us with data to characterize and model user behavior in cloud storage (see Chapter 5).

Other studies have presented new measurement and characterizations of cloud services workloads. Gracia-Tinedo et al. [2015] characterized user behavior in the service Ubuntu One (U1) based on data collected by the provider itself. Some user characteristics observed for U1 agree with our observations on Dropbox user behavior, presented in Chapter 5. For example, as shown by Gracia-Tinedo et al. [2015] for U1, we also observe the vast majority of small files against a few very large files (heavy

tail distribution showed in Dropbox). Moreover, as shown by authors, the flow in user modifications do exhibit periods of burstiness followed by long inter-modification times. On the other hand, Gracia-Tinedo et al. [2015] reported that only a small fraction of users (1.8%) share content in U1, which differs from our observation in Chapter 5. Unlike Dropbox, U1 is used more as a backup service rather than a platform for collaborative work.

Bocchi et al. [2015b] compared the usage patterns of cloud storage services on PC clients and mobile terminals through passive measurements in residential ISPs. The analyses focused on Dropbox, as it presented (at the time of study) the largest number of users with different terminals. The authors observed an increasing volume of upload from mobile terminals compared to previous measurements [Drago et al., 2012], possibly, associated with the growing production of multimedia content (e.g., photos and videos). On the other hand, larger volumes of downloads were observed from PC clients due to its automatic synchronization of content from the cloud. The authors concluded that PC clients still were responsible for a larger fraction of the traffic for cloud storage, although the number of users accessing this service from mobile terminals was significant. According to their measurement performed in 2014, users synchronized (i.e., uploaded and downloaded) 1442 GB, 231 GB and 215 GB from PC, mobile and web, respectively. Note that PC was responsible for more than 75% of the Dropbox traffic.

Another body of study focuses on developing cloud storage service benchmarks. In these studies, authors use primarily active measurements, which are experiments in a controlled environment to evaluate some functionality of the service. For example, Hu et al. [2010] analyzed four services, namely Mozy, Carbonite, Dropbox and CrashPlan by comparing their traffic, data transference delay, and CPU usage. They observed that service performance varies substantially with factors such as file size, data compression and de-duplication. Gracia-Tinedo et al. [2013] benchmarked three cloud storage services (Dropbox, Box, and SugarSync) using their public APIs. The authors characterized the mean upload and download transfer speeds of these services and analyzed the impact of geographic location on them.

Other benchmark studies focused on the operations users typically perform in cloud storage services. Li et al. [2013] analyzed the traffic overuse problem in Dropbox, which happens when users frequently modify files, and proposed an efficient batched synchronization mechanism to handle massive data interaction. In another study [Li et al., 2014], the same authors collected information about the types of files and operations performed by a dozen of volunteers in popular cloud storage services. From this dataset, they proposed a metric to quantify the service traffic usage efficiency

(TUE). The analysis of the TUE for different services revealed that compression and de-duplication can reduce the traffic between the cloud and user devices by up to 24%. Bocchi et al. [2015a] developed a framework to benchmark cloud storage services. They relied on an automated testing methodology to compare service capabilities and its implications on eleven different services. We discussed some capabilities regarding data transmission optimizations that services can adopt in Section 2.2. All such benchmark studies provide important aspects related to the performance of different cloud storage services. However, they relied only on active experiments in controlled environments, then lacking characteristics of user behavior.

Other studies performed active measurements with the target of developing alternative protocols to improve the performance of cloud storage services. Chen et al. [2014] proposed a framework that collects semantic information from files (e.g., file type) in order to improve the storage performance in data centers. Zhang et al. [2014b] pointed out that current services fail at preventing corruption and consistency of data as they rely on local file systems to update user data in the cloud. The authors proposed a mechanism to ensure data consistency from images (views) of the whole file system. Cui et al. [2015] proposed QuickSync, a framework that optimizes data synchronization by combining de-duplication, delta encoding and bundling techniques, all them parametrized by the size of chunk, in turn, data blocks depends on network conditions (network-aware chunking). Our user behavior model described in Chapter 5 contributes to such studies with a workload generator that can be used to evaluate new cloud storage protocols in realistic scenarios.

In sum, the characterization conducted by the studies presented in this section focused mostly on identifying service performance bottlenecks. Unlike them, in this dissertation, we have characterized cloud storage services with the purpose of modeling user behavior and the underlying client processes that generate workload for the service. Moreover, most of those prior studies have a clear concern on the traffic generated by services to transfer user files to/from the cloud. The workload patterns produced using our model can be directly mapped to network traffic, besides capturing various other related aspects, e.g., user inter-sessions times and files inter-modification times.

3.1.2 Workload Models

As far as we know, our study is the first that proposes models to generate synthetic workloads for cloud storage services. Nevertheless, the literature for models

and workload generators of Internet services closely related to cloud storage is rich. In the following we discuss the main differences of those studies to ours.

Several models and workload generators have been proposed for web request streams, and they may be represented by well known studies such as Mosberger and Jin [1998]; Barford and Crovella [1998]; Busari and Williamson [2002]; Krishnamurthy et al. [2006]. Two of them, model user behavior and its impacts on network traffic and, as such, are closer (in objective) to our work. SURGE [Barford and Crovella, 1998] is a synthetic workload generator that employs the concept of user equivalents to generate traffic to web servers, while SWAT [Krishnamurthy et al., 2006] considers user sessions and the dependence of content requests within sessions to generate synthetic workloads to web servers. Neither SURGE nor SWAT, however, include mechanisms to represent the dynamics of content generation (i.e., user generated content – UGC) and propagation of this content to other users (content sharing) as in cloud storage services. Hence, both approaches are insufficient to cover the particularities of such services.

Another line of research related to workload generators concerns the modeling of file-system characteristics. Mitzenmacher [2004] proposed a generative user model to explain the behavior of file size distributions. According to this model, the final size of a file results from successive modifications of its initial size (i.e., size rescaling), a process equivalent to multiplication of various Log-normal distributed random variables. This process yields a distribution for file sizes in which the body closely matches a Log-normal distribution and the tail resembles a Pareto distribution. The volume of uploads in Dropbox, which we have characterized to parametrize our user behavior model (see Chapter 5), fits these two distributions. Thus, our characterization contributes to validade Mitzenmacher's model hypothesis about file sizes, given that uploads in Dropbox are generated from files that users add or modify in namespaces.

Models to generate media streaming services also present some characteristics similar to cloud storage services. The studies presented by Almeida et al. [2001]; Jin and Bestavros [2001]; Tang et al. [2003] aim at generating synthetic workloads of media objects that include client sessions, arrival patterns and seasonal access patterns. However, similarly to SURGE and SWAT, these models do not consider UGC. On the other hand, a more recent media streaming workload model [Abad et al., 2013], which considers UGC, focuses mainly on the popularity of media objects, without including aspects fundamental to cloud storage performance, such as workload volumes and user sessions.

Regarding UGC, online social networks (OSNs) as Facebook and social media streaming sites as Youtube are popular Internet services which also share some

similarities in terms of user behavior patterns to cloud storage services. For example, Benevenuto et al. [2012] observed that session durations and inter-session times in OSNs follow heavy tail distributions, similarly to our characterization of Dropbox shown in Chapter 5. By comparing our measurements to studies on Youtube [Zink et al., 2009], we also have observed that both services present roughly similar data transference volumes: the mean volumes are around 7 MB. Despite such similarities, user behavior patterns in OSNs and social media streaming services are quite different from those in cloud storage services. For example, the most frequent activities in OSNs are browsing and comments [Cha et al., 2009; Benevenuto et al., 2012], such that some contents reach high popularity in terms of views and replies. By contrast, the most common user activity in cloud storage services is content synchronization, i.e., upload and download of files which may be modified over time by one or more users and replicated at real time among their devices.

More closely related to our work, some prior studies proposed hierarchical approaches to model the workload of e-business [Menascé et al., 2003] and media streaming [Costa et al., 2004; Veloso et al., 2006; Borges et al., 2012] services. According to Calzarossa et al. [2016], hierarchical decomposition of the workload is a powerful modeling approach in which changes to the characteristics of higher layers are automatically mapped and captured by the lower layers. Menascé et al. [2003] proposed a hierarchical model to study the interactions of the customers with e-commerce websites. Their model decomposes e-business workload into three layers: user sessions, application functions and HTTP requests. Each layer is described in terms of specific components such that the lower layers capture changes in user behavior (the highest layer). Following a similar approach, Costa et al. [2004]; Veloso et al. [2006]; Borges et al. [2012] proposed hierarchical models to capture the behavior of users in streaming services. These models added a pair of states (ON/OFF) to the user session layer to represent active and idle periods of video clients.

Inspired by the above hierarchical models, we here propose what we believe to be the first effort towards modeling cloud storage services workload [Calzarossa et al., 2016]. Table 3.1 summarizes the aforementioned models and our proposal taking into consideration the main functionalities of a typical cloud storage service (see Chapter 2 Section 2.2). Unlike previous efforts, our model, which is fully described in Chapter 5, captures content modifications which lead to data exchanges in the network, as well as content propagated over the network of shared folders and among the several devices of a user.

Table 3.1: Internet services models and workload generators (important efforts of the literature) closely related to cloud storage: comparison taking into consideration the main functionalities of a typical cloud storage service.

Work	User's Session	Modifications in Content	Content Propagation Network
Barford and Crovella [1998]	x	x	
Krishnamurthy et al. [2006]	x	x	
Mitzenmacher [2004]		x	
Almeida et al. [2001]	x		
Jin and Bestavros [2001]	x	x	
Tang et al. [2003]	x	x	
Abad et al. [2013]		x	
Zink et al. [2009]	x		
Benevenuto et al. [2012]	x		
Menascé et al. [2003]	x		
Costa et al. [2004]	x		
Veloso et al. [2006]	x		
Borges et al. [2012]	x		
Our proposal (Chapter 5)	x	x	x

Other studies characterized and modeled workloads of cloud infrastructure providers based on publicly available traces of Google data centers[1]. Mishra et al. [2010] proposed a methodology for task classification that consists in identifying groups of tasks (workloads). The authors used task duration, CPU and memory usage as dimensions for clustering workloads into different groups. The authors found 8 different clusters. Moreno et al. [2014] extended the task classification of Mishra et al. [2010] by including the user behavior dimensions: submission rate, requested CPU and memory. The authors showed that the high diversity of tasks is due to user behavior, then they identified six clusters considering user behavior in addition to three clusters considering tasks characteristics, and tasks clusters are associated with user behavior clusters. Those authors' studies are based on the aggregated workload of various services. Unlike them, the workload model proposed in this dissertation provides measures for individual user behavior, i.e., user equivalent requests, as proposed by Barford and Crovella [1998], in which various user equivalents will approximate the aggregated workload (see our model validation in Chapter 5 Section 5.4.2).

Finally, our workload model for cloud storage services contributes to efforts in studying cloud computing workloads in general. The workload classificators

[1] `http://googleresearch.blogspot.com.br/2011/11/more-google-cluster-data.html`

proposed by Mishra et al. [2010] and Moreno et al. [2014] can be extended to analyze the data input/output (I/O) dimension when clustering workloads. Our workload model, in turn, can be applied to generate tasks with high I/O usage, as it is a typical characteristic of cloud storage services. Delimitrou et al. [2011] proposed a workload generator that recreates I/O loads in data centers, based on the inter-arrival times of storage access seen in application traces. Our workload model will be able to generate such traces, thanks to the inter-modification time component (see Chapter 5 Section 5.2). Similarly, tools like CloudSim [Calheiros et al., 2011] and CloudAnalyst [Wickremasinghe et al., 2010] aim at simulating the effect of different scheduling algorithms on infrastructure providers. Our model can help such efforts in reproducing realistic workloads of cloud computing, since it models the behavior of end-users and consequent tasks generated by them, including the network traffic, for one of the largest currently deployed cloud services (i.e., Dropbox).

3.2 Content Sharing in Distributed Systems

In this section, we review the technical literature related to the second research goal (RG2) of this dissertation, which focuses on the effect of content sharing on cloud storage services. This functionality have been consolidated thanks to various researches on distributed systems, such as data synchronization architectures. We discuss different architecture models for distributed systems that can support content sharing in cloud storage services (Section 3.2.1). Afterwards, we discuss research efforts to measure the impact of this functionality and their derived social features on user activity (Section 3.2.2).

3.2.1 Data Synchronization Architectures

The typical cloud storage service architecture presented in Chapter 2 (see Figure 2.2) may adopt different distributed approaches to provide file synchronization and collaborative work for end users over the Internet. Example of approaches include the centralized client-server model, web caching, content distribution network (CDN), and the decentralized peer-to-peer model (P2P).

The centralized model has been adopted by most providers [Bocchi et al., 2015a]. In this model, all user devices send file updates/retrievals only to/from cloud storage servers (the data store component) located in a specific geographic region. According

to Marshall et al. [2012], the centralized model made it easier for end users to obtain data consistency in file synchronization and collaborative work operations. For example, users in different locations can perform collaborative work, being or not on-line simultaneously, counting on cloud stored folders to obtain the most recent version of files. Moreover, these folders actuate as an intermediary data repository that maintains file versions of each user device, thus enabling detection of conflict when users edit the same content simultaneously[2]. Previously to cloud storage services, the main strategy adopted by users to accomplish these tasks was to send content attached in emails, which may not be a convenient way when several pieces of content are changed by multiple users at the same time [Massey et al., 2014].

However, the centralized model has its issues regarding performance and costs. From the end user perspective, this model impacts mainly on synchronization speed. As we have discussed in Chapter 2, larger distances between globally located users and a centralized cloud store component yield longer times for data synchronization, as appointed by measurement studies of Gracia-Tinedo et al. [2013] and Bocchi et al. [2015a]. From the service provider perspective, network traffic due to file synchronization and collaborative work, indeed, imposes high costs in terms of data transference payments to/from infrastructure providers [Li et al., 2014]. For example, Dropbox, in order to serve customers globally, still keeps part of data store component on Amazon cloud [Akhil Gupta, 2016], which charges for the outbound traffic (i.e., traffic from the cloud to the user devices) in addition to storage space. Such concerns with costs are particularly important given that retrieving data from the data store component may be as expensive as storing data[3].

Web caching and CDNs are classical solutions to offload servers in the centralized model. Web caching has been proposed in the 90's for achieving improved client performance (e.g., faster load time of web pages and file downloads) and reduced network cost (e.g., traffic across different ISPs) [Rabinovich and Spatschek, 2002]. In a classical setup, it consists of hardware components deployed within a local network (e.g., ISPs) to store (cache) cacheable HTTP contents[4] accessed by the customers within this network. CDNs, in turn, implement geographically distributed servers to cache a frequently accessed content (CDNs cache only content for which they are paid).

[2] Dropbox keeps track of file versions and create a file copy to appoint conflicted versions, whereas Google Drive supports on-line synchronous editing, i.e., various users can view their changes in the same file simultaneously.

[3] By the time of this writing, Amazon charges \$0.09/GB/month for outbound traffic and \$0.05/GB/month for data storing in magnetic disks.

[4] This refers to HTTP content that has longer Time-To-Live (TTL), thus it maybe be requested (reused) by many clients when cached, e.g., news and multimedia content

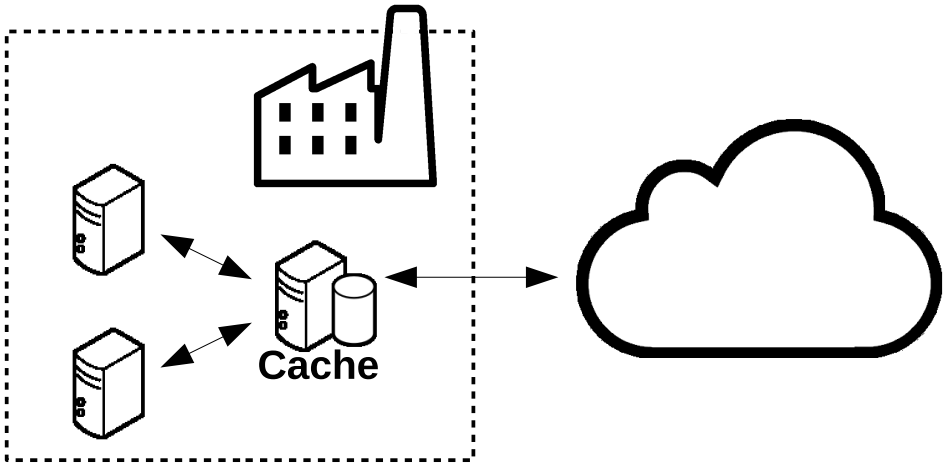


Figure 3.2: Cloud-based file storage architecture with caches for enterprises.

Thus, rather than having millions of users accessing a single (central) server platform, it is possible to spread this load across multiple server nodes [Peterson and Davie, 2012].

Web caching is an approach simpler than CDN to offload the centralized model, as it is set in a local network that aggregates various service costumers. Some studies [Vrable et al., 2012; Bessani et al., 2014] as well as commercial products[5] have proposed cloud-based file system architectures that considers caches in order to reduce download traffic and content access latency for enterprises. Figure 3.2 illustrates the general idea of such architecture. It consists in a cache component placed in the network infrastructure of the organization, acting as a file server to multiple clients. The cache provides access to content frequently modified by users, pushing file updates to the cloud, but enabling the retrieval of files locally whenever possible. The use of cloud-based file systems for the university environment have also been studied. Liu et al. [2013] analyzed access pattern in traces of a large university file system. The authors observed that a small portion of files are often accessed in this environment. Through a trace-driven simulation, they showed that caches located closer to users can provide faster access to such files, then improving the user experience.

The CDN model has already proved to be an efficient mean to distribute content over the Internet [Erman et al., 2009]. Google Drive is a popular cloud storage service that applies CDN, according to a benchmark study performed by Bocchi et al. [2015a]. This study showed that Google Drive round trip time (RTT) is around 10-fold lower than that observed in Dropbox and Onedrive. This results in shorter synchronization times for Google Drive without resorting to transmission optimizations as compression

[5]They are known as cloud storage gateways and offered by companies as Nasuni (`http://www.nasuni.com/PanzuraCloudFS`) and Panzura CloudFS (`http://panzura.com`).

and deduplication. Bocchi et al. [2015a] also observed that the data store component of One Drive, a popular cloud storage service provided by Microsoft, is supported by a CDN of four edge nodes. In comparison, the same study reported that Google Drive uses more than thirty nodes spread in six continents. The service benefits with shorter RTT, however, comes with a high cost typically feasible for big Internet companies as Google, which count on a large CDN not only to provide storage but other different services as content search, video (Youtube), email, etc.

Netflix is a world company, which is specialized in streaming video-on-demand over the Internet. Netflix built its own CDN to achieve this specific task. The company invite ISPs around the word to participate in its CDN, and the qualified ISPs receive a cache node, named *Open Connect Appliance (OCA)*, to connect to this CDN. There are two OCA types: the storage node with large storage and medium throughput (10 TB SSD and 240 TB HDD) and the flash node (4 TB flash) with limited storage and high throughput. Each node type is allocated to a ISP according to a set ISP characteristics, e.g., number of Netflix's subscribers, content popularity and proximity to an Internet exchange point. Netflix uses the offload metric, i.e., bytes served from ideal location over total bytes served, to evaluate the performance of its CDN. The company reports almost all its content is served from the local OCAs [Ken Florance, 2016]. This has the dual benefit of reducing the ISP's cost of operation and ensuring the best possible Netflix experience for their subscribers.

Given the benefits of web caches and CDNs, no previous studies analyzed whether the traffic caused by content sharing in cloud storage is significant for adopting such distributed architectures. If so, given observed content sharing patterns, is the deployment of caches in networks such as universities or ISPs advantageous for the service provider? The aforementioned studies [Vrable et al., 2012; Liu et al., 2013; Bessani et al., 2014] and companies examples (Google Drive and Netflix) show the benefits of such distributed architecture for users. We, however, focus on an issue not yet investigated that concerns how cost-effective is such architecture for the service provider. We present this investigation in Chapter 6 regarding the RG2 of the dissertation.

A large body of research on file sharing focuses on architectures based on the peer-to-peer (P2P) model [Lee, 2003]. One of the main advantages of this model, compared to the centralized one, is data transference savings and communication resilience [Shen et al., 2011]. For example, users collaborating in the same document can synchronize updates among themselves without intermediation of cloud control servers for reasons of data transference economy and Internet failure of control servers. The `LAN Sync` protocol implemented in Dropbox (see Chapter 2) adopts a P2P model within

a local area network (LAN), even though this protocol requires server intermediation, e.g., to notify devices (peers) about new updates in shared files. The advantages of a P2P architecture, however, depend on peers being online at the same time. Scenarios where peers sharing a file are not online simultaneously to synchronize updates may be handled via the centralized model, as usually performed in cloud storage services.

Early research on P2P models related to file synchronization tackled the challenge of ensuring data consistency across devices with weak connectivity, i.e., devices often alternating between online and offline states - churn [Stutzbach and Rejaie, 2006a] - without intermediation of central servers. Cimbiosys [Salmon et al., 2009] is a framework that enable P2P synchronization of files among multiple devices aiming to guarantee the *eventual consistency model* [Tanenbaum and Van Steen, 2007]. To agree with this model, Cimbiosys coordinates replicas of a specific file across all user devices over time in order to guarantee that, eventually, all accesses to a replica will return the last updated value. Recent studies [Perkins et al., 2015; Chandrashekhara et al., 2015] have improved Cimbiosys to ensure data consistency across cloud storage services over mobile applications. However, these studies also apply the centralized model of synchronization with the cloud.

In a recent effort to enhance the P2P model for cloud storage services, Chaabouni et al. [2014, 2016] tackled the problem of the efficient distribution of shared files among end-user devices. Authors proposed an automatic method to switch from centralized to P2P model upon detection of a certain critical mass demand on a specific content (i.e., high number of peers requesting a content) in order to avoid bottlenecks and save data transference to/from the cloud. The challenge of this method, as in *LAN-Sync*, is to handle the short period in which peers are on-line simultaneously to exchange data. Supposing this period is long and most of shared files have large size, which is a hypothetical scenario proposed by authors, the method can provide improvements over the centralized method in the download time that exceed 65%.

In summary, we have observed that content retrieval in cloud storage services is an issue that requires further investigation and improvement. Two important aspects that must be considered are the cost and the speed to transmit content from the cloud to user devices. Although alternatives to the traditional centralized file synchronization model have been proposed for enterprises, there is still a lack of research to evaluate how cost-effective such alternatives are, given realistic scenarios observed in popular services as Dropbox. In this sense, our contribution (described in Chapter 6) is to provide an analysis of repetitive downloads of same contents from the cloud (and thus data transference waste) to users within the same network, and whether the introduction

of network caches would reduce the volume of these downloads in a cost-effective way, even in scenarios where `LAN Sync` is unable to reduce them.

3.2.2 Evaluating User Activity

Intuitively, functionalities of cloud storage services as file synchronization and collaborative work, which increase the interaction among users and enable social features, might contribute to increase the user activity, avoiding their departure from the service. Evaluating the impact of these functionalities and their derived social features on user activity is an important issue for a service provider that aims to attract users and increase its profit, e.g., by having users storing more data in the service. Previous research efforts with this purpose have adopted either a qualitative analysis extracted from user interviews or a measurement of the user quality of experience (QoE). In the following, we discuss these studies and point out our contribution for this evaluation.

Marshall et al. [2012] developed a prototype of cloud storage service named Cimetric and analyzed how users explore the following functionalities: (i) backup of files to the cloud, (ii) synchronization of files with different devices, and (iii) sharing of files with other users. The design of Cimetric consists of general hybrid cloud storage service with synchronization and sharing functionalities via P2P architecture and the option of backup files to the cloud. After monitoring the use of Cimetric by a local community of 12 distinct people involved in 9 collaborative research activities during one year, the authors concluded that synchronization and sharing are functionalities more required by users than backup, which was mostly required when the P2P approach could not support user activities (e.g., due to collaborators not being online at the same time and organizational firewalls barriers).

Palviainen and Rezaei [2015] analyzed the user experience and practices regarding the use of popular cloud storage services (Dropbox, Google Drive, One Drive, and ICloud). The authors conducted interviews with 19 distinct people divided into two rounds in different dates (separated by two years) in order to analyze the changes of social interaction that might happen over time in the services. They concluded that: (1) content sharing and multiple devices synchronization are the primary motivations for the users to start using a certain cloud storage service, and (2) those are also the most satisfying experience reported by users, followed by backup and ease of use. The authors also investigated the potential of integrating new functionalities for user interaction, such as tagging, private messaging, rating, and user status, in current

services. Interestingly, the majority of users rejected such integration. The authors reasoned that users have already managed this type of functionalities in a multitude of services such as OSNs, and thus they preferred not to add them to cloud storage.

Other studies have conducted qualitative analysis focusing on usage problems regarding file synchronization and collaborative work. Example of problems are, misuse of write/read rights on shared content (e.g., deleting shared folders) [Marshall and Tang, 2012], conflicts with multiple digital identities (e.g., work and hobby profiles) when operating shared folders [Voida et al., 2013], difficulties of users with file versioning [Massey et al., 2014], and limited support for tracking files and user activities for collaborative work [Nebeling et al., 2015]. Despite such issues, all these studies advocate that file synchronization and collaborative work are essential functionalities for cloud storage services that should be improved by providers. Ultimately, all qualitative analyses we have discussed so far provide evidence that such functionalities are important requirements for service adoption by users. However, these studies do not aim to quantify the relationship of such functionalities with user activity. In other words, they do not propose measurable metrics, which allow service providers to evaluate how synchronization and collaborative work as well as their derived social features impact user activity.

Some studies [Hobfeld et al., 2012; Casas et al., 2013; Amrehn et al., 2013; Casas and Schatz, 2014] have advocated QoE perceived by users as guidance for managing quality in cloud computing, in particular cloud storage services. Although QoE is perceived subjectively by the end user, Hobfeld et al. [2012] argued that a typical approach for assessing it is to calculate the mean opinion scores (MOS) out of subjective tests. Thus, MOS expresses the opinions of individual users aggregated and meant to reflect the opinion of an average user. Hobfeld et al. [2012] also proposed mapping QoE to other measurable variables, such as network quality of service (QoS), CPU, memory and storage usages, in order to estimate and monitor QoE by service providers.

Amrehn et al. [2013] conducted experiments with 52 participants in a controlled laboratory environment to investigate QoE during the regular usage of Dropbox. The authors correlated waiting time with MOS in order to quantify QoE in this service for different tasks as initialization, storage (i.e., upload), retrieval (i.e., download) and multi-device synchronization (both upload and download). They concluded that the end-user perception of QoE is dependent on the task, and in general, users were more tolerant to storage than retrieval operations. Casas et al. [2013] as well as Casas and Schatz [2014] conducted similar experiments to correlate QoE to QoS measurements in Dropbox. Their studies considered bandwidth as the most perceived QoS metric for users in terms of QoE, and showed the correspondence of both metrics. The authors

argued that such correspondence provides means (i.e., measurable QoE parameters) to guide ISPs, service providers and end users to establish bandwidth requirements for a satisfactory usage of the service. These studies, though important in terms of proposing measurable metrics for user activity, do not deal with the interaction among various users (i.e., social features).

Jeners and Prinz [2014] proposed performance metrics to support the analysis of a large enterprise system developed for collaborative creation and edition of documents over the Internet. The authors proposed metrics, such as activity, productivity and cooperativity, which aim to analyze the overall behavior of a group of employees working on projects as well as the level of collaboration in the group (i.e., combination of the metrics with states low/high). Despite the focus on interaction between users and collaborative work, the study by Jeners and Prinz [2014] does not correlate any metric indicating users activities in terms of system resources consumption (CPU, memory, storage, data transference) to the level of interaction among multiple users.

In sum, most prior analysis of user interaction and its derived social features in cloud storage services have been performed by means of qualitative studies. Those studies do not quantify the impact of these features on the level of user activity in the service. On the other hand, prior QoE studies have proposed quantitative approaches to evaluate user activity. Yet those studies do not tackle user interaction and social features. In this dissertation, we bridge this two extremes by evaluating the impact of social features on user activity in a cloud storage service in different networks. To this goal, we propose a different approach, not yet adopted by aforementioned studies, that relies on analyzing the activity of users from network traffic collected passively from the service in different networks. We provide detailed explanation of our approach and the results of our investigation in Chapter 6.

3.3 Economic Analysis

In this section we review studies related to our third research goal (RG3), which regards modeling the tradeoffs between end users and service provider interests. First, we discuss studies that analyzed such tradeoffs taking into consideration either the service provider or end users perspectives separately in cloud storage services (Section 3.3.1). Next, we discuss some studies that analyzed both perspectives jointly, where the interest of each party was modeled as a utility function.

3.3.1 Cost-Benefit Analysis of Cloud Storage Services

The costs and benefits of cloud storage services for end users have been addressed by few prior studies. The key aspects investigated by these studies are pricing policies [Naldi and Mastroeni, 2013], willingness to pay for the service [Shin et al., 2014] and adoption of multiple service providers [Naldi, 2014; Yeo et al., 2014].

For instance, Naldi and Mastroeni [2013] surveyed the cloud storage packages offered by major providers aiming at proposing a method to compare those services based only on their pricing strategies. The authors observed that the majority of providers offer a free service with a limited storage space and the option to subscribe to a paid plan, i.e., the freemium business model (see Chapter 2 Session 2.2). In general, providers adopt a bundling price policy, i.e., the service sells packages of a maximum storage capacity associated with fixed monthly/yearly fees. The authors argued that even considering only price (i.e., disregarding technical capabilities [Bocchi et al., 2015a]), comparing services prices is a complex task for end users. To perform such comparison, the authors proposed two approaches: pointwise comparison (services prices were grouped into ranges of maximum capacity) and Pareto dominance (services prices were decomposed into two parts: fixed fee and marginal price per volume unit). The authors argued that both analyses will enable end users to select a prospective provider in terms of service price.

Shin et al. [2014] conducted a quantitative analysis to show the adoption behavior of end users with respect to packages offered by main cloud services providers in Korea. The authors surveyed 400 people in order to collect their preferences regarding core attributes of services. Based on this data collection, the authors analyzed the marginal willingness-to-pay the service and the relative importance of core attributes. Among analyzed attributes, service fee and reliability (i.e., service availability) are the must-be attributes. As main result, the authors concluded that users are more willing to pay for higher reliability than larger storage capacity. Moreover, the authors also concluded that multiple devices synchronization is the attribute with the lowest willingness-to-pay. Collaborative work, which was shown to be the most required functionality by users according to qualitative analyses discussed in previous section [Marshall et al., 2012; Palviainen and Rezaei, 2015], was not considered in this study.

Subscription to multiple cloud storage services, also known as “cloud of clouds” [Bessani et al., 2014], has been an approach for end users to prevent data

lock-in[6] in a given provider. Naldi [2014] argued that storing data on multiple services provides higher reliability than a single one, but does not guarantee that data are always accessible. Extending the price analysis of his previous work [Naldi and Mastroeni, 2013], the author modeled the costs to combine multiple services and an insurance policy, aiming at guaranteeing service availability and risk coverage against cloud storage unavailability. Yeo et al. [2014] proposed a framework to aggregate storage capacity from multiple service providers. The main goal of this framework is to provide an economic benefit for end users, given by free storage quote offered by several services. This framework, however, requires some management tasks, for example, users have to keep track of files within preferred services due to specific functionalities such as collaborations, in addition to account authentication on multiple services. The authors argued that the service without costs (i.e., non-monthly fee) for end users compensates such management tasks.

The aforementioned studies pose important benefits for end users, however, they do not take into consideration the service provider interest. Other prior research efforts focused on a cost-benefit analysis for the perspective of service providers. For example, Wu et al. [2013] developed a framework (SPANstore) to combine multiple infrastructure providers with focus on minimizing cost and decreasing latency of service. The authors argued that there are large divergences among prices of infrastructure providers (e.g., Amazon, Google and MS Azure) for storage and data transference. SPANstore addresses the tradeoff between storage, latency and data transference costs to replicate data in closer data centers. To accomplish this, the authors formulated an optimization problem including storage, latency and data transference variables.

Wang et al. [2012] analyzed the main costs to run a cloud storage service like Dropbox. The authors argued that computing capacity (e.g., due to encryption and computation of data block hash keys) should also be considered in cloud storage services costs, in addition to storage, data transference and latency cost factors. According to authors, the service should tackle such processing tasks in exclusive virtual machines (VMs) in order to decrease file synchronization delay. Thus, the authors formulated an optimization problem to minimize the number of VMs necessary to perform processing and data synchronization tasks separately and decrease delay. We intend to build upon these prior models to develop new cost models and analysis for our RG3, which consider, for example, the data transference costs for content sharing.

Karunakaran and Sundarraj [2015] surveyed the pricing policies adopted by main infrastructure providers and proposed an organization of these policies into *static* and

[6] When users cannot easily extract their data from one service provider to transfer to another provider, mainly in terms of monetary costs, time or data transference cost [Armbrust et al., 2010].

Table 3.2: Pricing models categories considering prices adopted by the main infrastructure providers.

Pricing Model	Descriptions	Scheme	Usage Example
Static	Resource has an up-front price to be provisioned	on-demand	consumers are charged per hour of resource usage
		reserved	consumers pay a constant price and can use the resource for any hour given the subscribed period
Dynamic	Resource price fluctuates periodically depending on the infrastructure supply and demand.	bids	consumers place bids to obtain the resource

dynamic pricing models, as shown in Table 3.2. The authors argued that infrastructure providers are moving away from a simple, static pricing model to the dynamic one, which will require more complex decisions of consumers (e.g., cloud service providers) in order to obtain monetary savings. Karunakaran and Sundarraj used the Amazon Spot Instance service, which follows the dynamic price model, to analyze the behavior of consumers. The authors observed that a significant fraction of consumers bid values above the on-demand scheme (see Table 3.2) to increase their chances of winning, as final price of resource may be lower than their bids[7]. According to the authors' analyses, such behavior is not advantageous, as consumers' job-completion time (or job interruptions) do not have significant reduction as bids get higher (beyond on demand scheme price). The main lesson of this study is the need for better analysis of cloud computing market to help consumers in making more strategic decisions. We intend to conduct similar analysis for our RG3, however, with focus on issues regarding the relationship between service providers and end users for cloud storage services.

3.3.2 Utility Functions

The studies we have discussed in the previous section provide an analysis of costs and benefits limited to either the provider or the end user perspective. An alternative approach is to analyze both perspectives jointly as a conflict of interest problem, where the interest of each party involved in the problem is modeled as a utility function.

[7] In Amazon Spot Instance, consumers only pay the "spot-price" irrespective of how high is their bid value. Spot-price is defined dynamically depending on the infrastructure supply and demand (`http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html`).

Utility is an important concept in economics and game theory, that has been widely applied to analyze the relationship between cloud service providers and users [Xu and Li, 2013; Joe-Wong et al., 2015; Shen and Li, 2015; Lin and Tzeng, 2014]. In the following, we discuss some studies that propose this kind of analysis (we have found only one study related to cloud storage services). It is worth mention that the utility concept applied in those studies is also known, in economics, as *marginal utility*, as it represents consumers additional satisfaction obtained from consuming one additional unit (or increasing units) of the same good [Pindyck and Rubinfeld, 2013].

Xu and Li [2013] studied prices of cloud infrastructure resources. The authors proposed a model to analyze theoretically the infrastructure provider revenue maximization, given by the difference between consumers' utility and the cloud resource price. Consumers' utility is a function of resource consumption with the same shape for all consumers. However, this function is weighted by the consumer's utility level, which vary across consumers, i.e., classes of consumers with different valuation for a given cloud resource. The cloud resource price, in turn, is a linear function of the resource unit price and quantity of units. We build upon Xu and Li's study to develop the users' utility function of our cost-benefit model (RG3). Regarding the provider's utility, different from those authors, we consider a service provider that outsources storage and data transference resources from a cloud infrastructure provider. This is the case of Dropbox,[8] U1 [Gracia-Tinedo et al., 2015] and other general cloud services as Netflix and Airbnb.[9]

Joe-Wong et al. [2015] used utility functions to analyze a current practice in the United States mobile data market, that is content providers to subsidize users' cost of mobile data, relying on ads or freemium subscription to make money. To achieve this analysis, authors proposed utility functions to represent benefits minus costs of three agents that interact in the issue: the content provider, users and the ISP. Authors' analyses show all three parties can benefit from sponsored data, although users achieve higher utility improvements than content providers. The proposed model also can support content providers decision about which and how much content to sponsor.

Shen and Li [2015] proposed utility functions to investigate a win-win scenario in which tenants compete for bandwidth of a provider. Such competition leaded to providers and consumers reduced utility, as consumers have low network utilization, preventing providers to meet service level agreements (SLA). To deal with this issue, authors proposed a dynamic bandwidth pricing policy which establishes three

[8] At the time of our data collection.

[9] See an exhaustive list of Amazon infrastructure customers in `https://aws.amazon.com/solutions/case-studies/all/`

usage groups: reserved bandwidth, congested links and uncongested links. Unit prices for reserved bandwidth is greater than for congested links, which in turn is greater than for uncongested links. Authors' analyses showed the policy enables a harmonious environment, where a consumer has incentive to use uncongested links, then cooperating to increase its utility as well as other consumers and provider's utilities. Although relevant to our work, Shen and Li [2015] study as well as the afore mentioned Joe-Wong et al. [2015] study do not take into account peculiarities of cloud storage services, such as content sharing.

The only prior effort to analyze users and provider's utility in cloud storage services [Lin and Tzeng, 2014] focuses on the cost to replicate users' data into multiple disks. Authors observed that replication incurs overhead to infrastructure provider as well as auditing replication incurs costs to consumers. Thus, consumers prefer less auditing, whereas infrastructure provider prefer less replication. However, data loss due to non replicated data yields decreased utility for both parties: the infrastructure provider receives SLA punishments, whereas the service provider is punished due to data unavailability. To tackle this problem, the authors developed utility functions for both infrastructure and consumers that consider, in an abstract level, the benefits, costs and punishments for each party. The authors' model allows a rational analysis of the most important aspects of the conflict, i.e., the number of replicates and discount for reduced replication that satisfy both infrastructure and consumers.

In sum, we have observed that utility function model is an important approach to tradeoff analyses that includes multiple agents (e.g., service provider and users) with conflicting interests in a given situation. We have adopted utility functions for modeling costs and benefits of cloud storage services for both providers and end users, as proposed for our RG3. Different from Lin and Tzeng [2014], we take into consideration the role of data transfer costs and content sharing, which are key features to cloud storage services as Dropbox, Google Drive and Onedrive, which focus on end users. Additionally, we have applied our model to evaluate two alternative policies for the current cloud storage sharing architecture. We describe in Chapter 7 our efforts and results towards this goal.

3.4 Summary

In this chapter we reviewed previous studies that provide background and motivation for the three RGs proposed in this dissertation.

In Section 3.1, we discussed studies about workload characterization and modeling for cloud storage services (RG1). We observed that state-of-art studies have focused mostly on identifying services performance bottlenecks, and although the literature for workload models of Internet services is wide, models of user behavior of cloud storage services and its derived workload are still lacking. Table 3.3 summarizes the most important related work regarding RG1, comparing them with our contribution. Details about our contribution towards RG1 is discussed in Chapter 5.

In Section 3.2, we discussed the literature related to the impact of content sharing and its derived social features on cloud storage services (RG2). This covers research efforts on alternative synchronization models such as caches, CDNs and P2P, as well as studies on the evaluations of the activity of users of existing services. Related to these research efforts we have proposed new analyses and measurement techniques to evaluate the impact of data sharing on cloud storage services costs as well as the importance of such functionality for end users. Table 3.4 summarizes the most important related work regarding RG2, comparing them with our contribution. We present our contributions for RG2 in more detail in Chapter 6.

Finally, in Section 3.3, we have discussed prior efforts to analyze the costs and benefits of cloud storage services (RG3). Thus, we have focused on studies that model conflicts of interest among agents with utility functions, which might provide us technical knowledge to model costs and benefits of cloud storage services for the provider and users. Table 3.5 summarizes the most important related work regarding RG3, comparing them with our contribution. Our results towards that goal are provided in Chapter 7.

In the next Chapter, we present the methodology we used to collect data to support our investigation, and we provide an overview of our collected datasets.

Table 3.3: The most important related work regarding RG1 and our contribution related to the work.

Related Work	Work focus/contributions	Our contribution related to the work
Drago et al. [2012] Gracia-Tinedo et al. [2015]	Cloud storage services typical usage and performance analyses from passive measurement / server side log	User behavior model and workload generator based on characterization (statistical distributions) of main users' actions in cloud storage services
Bocchi et al. [2015b]	Usage patterns of cloud storage services on PC and mobile terminals	
Hu et al. [2010] Gracia-Tinedo et al. [2013] Bocchi et al. [2015a]	Cloud storage service benchmarks	
Li et al. [2013] Li et al. [2014]	Traffic usage efficiency analysis of Cloud storage services	
Chen et al. [2014] Zhang et al. [2014b] Cui et al. [2015]	Protocols to improve the performance of cloud storage services based on usage patterns	
Barford and Crovella [1998] Krishnamurthy et al. [2006]	User behavior models and network traffic generators for web services	Workload model for user generated content (large volume as text, multimedia, backup files) propagated over content sharing networks
[Mitzenmacher, 2004]	Files sizes generative model	
Almeida et al. [2001] Jin and Bestavros [2001] Tang et al. [2003] Abad et al. [2013]	Synthetic workload generator for media objects	
Cha et al. [2009] Zink et al. [2009] Benevenuto et al. [2012]	User-generated content models for on-line social networks	
Menascé et al. [2003] Costa et al. [2004] Veloso et al. [2006] Borges et al. [2012]	Hierarchical user behavior models for video streaming and e-business workload generation	
Mishra et al. [2010] Moreno et al. [2014] Delimitrou et al. [2011] Wickremasinghe et al. [2010] Calheiros et al. [2011]	Data centers aggregated workload analyses	Cloud storage services workload generator based on a given quantity of users (i.e., user equivalent workload)

Table 3.4: The most important related work regarding RG2 and our contribution related to the work.

Related Work	Work focus/contributions	Our contribution related to the work
Erman et al. [2009]	Web caches performance analysis based on ISP traffic traces	Investigation of content sharing impact on traffic of a cloud storage service based on campuses and ISPs traffic traces. Performance analysis of a cache based architecture that aim at reducing traffic (downloads) due to content sharing
Vrable et al. [2012] Bessani et al. [2014]	Cloud based file system proposals for supporting content sharing in enterprises	
Liu et al. [2013]	Campus storage system analysis based on server side access log	
Chaabouni et al. [2014] Chaabouni et al. [2016]	Performance analysis of a hybrid cloud-P2P storage service based on server side logs	
Marshall et al. [2012] Palviainen and Rezaei [2015] Marshall and Tang [2012] Voida et al. [2013] Massey et al. [2014] Nebeling et al. [2015]	Content sharing usage analyses in popular cloud storage services based on qualitative research	Investigation of social features impact on user activity of cloud storage based on campuses and ISPs traffic traces
Casas et al. [2013] Amrehn et al. [2013] Casas and Schatz [2014]	Correlation between user QoE and QoS for cloud storage services	
Jeners and Prinz [2014]	User activity evaluation methodology for systems designed for collaborative work	

Table 3.5: The most important related work regarding RG3 and our contribution related to the work.

Related Work	Work focus/contributions	Our contribution related to the work
Naldi and Mastroeni [2013]	Structural pricing model proposal for cloud storage services	Alternative content sharing policies for improving cloud storage users benefits
Shin et al. [2014]	Analysis of user willingness to pay cloud storage services	
Naldi [2014] Yeo et al. [2014]	Model and framework proposals to analyze the benefits of adopting multiple cloud storage service providers	
Wu et al. [2013] Wang et al. [2012] Karunakaran and Sundarraj [2015]	Strategies for cloud infrastructure consumers (companies or people that rent VM/storage) to obtain lower prices	Alternative content sharing policies for reducing cloud storage provider transference cost
Xu and Li [2013] Joe-Wong et al. [2015] Shen and Li [2015] Lin and Tzeng [2014]	Modeling conflicts of interest among agents (cloud infrastructure providers and consumers) with utility functions	A general model for analyzing tradeoffs between costs and benefits of both users and providers in cloud storage

Chapter 4

Data Collection

In this chapter, we describe our data collection methodology that is based on Dropbox. We focus on Dropbox because it has been the most popular cloud storage service until the time of this writing [Drago et al., 2012; Bocchi et al., 2015b; Duarte et al., 2015; Oliveira et al., 2015] and Dropbox architectural design has been well known among researchers thanks to different benchmark studies [Hu et al., 2010; Wang et al., 2012; Gracia-Tinedo et al., 2013; Li et al., 2014; Bocchi et al., 2015a]. Moreover, Dropbox presents the most important functionalities for a storage service from the end user perspective: backup, file synchronization and collaborative work [Marshall et al., 2012; Palviainen and Rezaei, 2015]. These functionalities are explored by our three research goals (see Chapter 1), which will be tackled in the subsequent chapters.

We organize this chapter into three sections, noting that some technical terms used along these sections will refer to the Dropbox architecture overview presented in Chapter 2 Section 2.3. First, in Section 4.1, we describe how we perform passive measurements to collect data related to Dropbox from edge networks. Afterwards, in Section 4.2, we present our datasets collected in four different edge networks, which includes university campuses and Internet Service Provider (ISP) networks. Finally, in Section 4.3, we describe our methodology to post-process data, which covers mechanisms to reconstruct Dropbox user sessions and to extract information about data transmission and sharing.

4.1 Methodology

We rely on the open source Tstat tool [Finamore et al., 2011] to perform passive measurements and collect data related to Dropbox. We installed Tstat on vantage points of different edge networks as large university campuses and ISPs. Tstat monitors

each TCP connection, exposing flow level information, including: (i) anonymized client IP addresses; (ii) timestamps of the first and the last packets with payload in each flow; (iii) the amount of exchanged data; and (iv) the Fully Qualified Domain Name (FQDN) the client resolved via DNS queries prior to open the flow [Bermudez et al., 2012].

We adopt a methodology similar to the one in [Drago et al., 2012] to filter and classify Dropbox traffic. Specifically, we take advantage of the FQDNs exported by Tstat to filter records related to the several Dropbox servers. To this end, we isolate flow records related to the storage of data by looking for FQDNs containing *dl-client*.dropbox.com*. Similarly, we use *notify*.dropbox.com* to filter flow records associated with the notification of updates and heartbeat messages (see Chapter 2 Section 2.3.2).

In addition to TCP level flows, we use the HTTP monitoring plug-in of Tstat to export, simultaneously to the flows, information seen in heartbeat messages. For each heartbeat message, we gather: (i) the device/host ID; (ii) each namespace ID registered in the device; (iii) the latest journal ID of each namespace; (iv) the user ID; and (v) the timestamp of the message. Figure 4.1 shows an example of heartbeat message piece, which we extract such information: the HTTP get field of the message. Note each information is separated by the character "&", and the *ns_map* field contains each namespace ID followed by its latest journal ID. A heartbeat message also includes a *nid* field to identify the client session ID.

```
/subscribe?host_int=1111111111&ns_map=111111111_11111111111111,
111111111_11111111111111111111111_1111111111111,111111111_1111
1111111111,111111111_11111111111111&user_id=111111111&nid=11111
11111111111111&ts=1111111111
```

Figure 4.1: Example of information transmitted in heartbeat messages: original numeric keys for each information ID were replaced by number ones.

4.2 Datasets

We have collected traffic in four different networks, including University campuses and ISP networks. Table 4.1 summarizes our datasets, presenting the number of Dropbox devices, the Dropbox traffic volume, the number of unique namespaces[1], the number of times the journal ID (JID) of a namespace has changed, and the data capture period.

[1] Recall Dropbox treats the initial folder selected by user at configuration time and all shared folders indistinctly. They are all called *namespaces* as we describe in Chapter 2 Section 2.3

Table 4.1: Overview of our datasets.

Name	Devices	Upload (TB)	Download (TB)	Namespaces	Changes[1]	Period
Campus-1	10,609	1.7	3.8	34,343	2,690,617	04/14–06/14
Campus-2	1,987	0.6	1.0	9,861	507,394	04/14–06/14
PoP-1	9,478	3.9	6.8	28,899	1,657,744	10/13–04/14
PoP-2	3,376	2.4	3.7	12,050	1,306,045	07/13–05/14
Total	25,540	8.6	15.3	85,153	6,161,800	–

[1] Number of times the journal ID of a namespace has changed.

Campus-1 and Campus-2 datasets have been collected at border routers of two distinct campus networks. Campus-1 includes traffic of a large South American university with user population (including faculty, student and staff) of $\approx 57,000$ people. Campus-2 serves around $\approx 15,000$ users in a European university. Both datasets include traffic generated by wired workstations in research and administrative offices, in addition to WiFi networks in Campus-1 and students houses in Campus-2.

PoP-1 and PoP-2 have been collected at distinct Points of Presence (PoPs) of a European ISP, aggregating more than $25,000$ and $5,000$ households, respectively. ADSL lines provide access to 80% of the users in PoP-1, while Fiber To The Home (FTTH) is available to the remaining 20% of users in PoP-1 and to all users in PoP-2. Ethernet and/or WiFi are used at home to connect end users' devices, such as PCs or smartphones.

In total, we have observed more than 20 TB of data exchanges with Dropbox servers, 25 k unique Dropbox devices and 85 k unique namespaces during more than 1 year of data collection. Our datasets provide not only a large-scale view in terms of number of users/devices, but also a rich and longitudinal picture of the Dropbox usage. A key contribution of our work is the release of anonymized data to the public, aiming to foster new research and independent validations of our work. We invite interested readers to contact us in order to access the datasets.

4.3 Data Preparation

Our data capture resulted in two distinct and independent types of datasets for each analyzed network: (i) flow information that includes the data volume exchanged by clients with Dropbox storage servers (i.e., data messages); (ii) heartbeat messages (i.e., control messages) with metadata related to the Dropbox namespaces. However, these two data sources have no common keys to provide an association between notifications of updates in namespaces and storage flows (i.e., data volume). Even more, we observe no one-to-one correspondence between notifications and flows. Indeed, different

notifications might be exchanged in a single TCP flow for data volume, whereas a single notification might be split across several TCP flows.

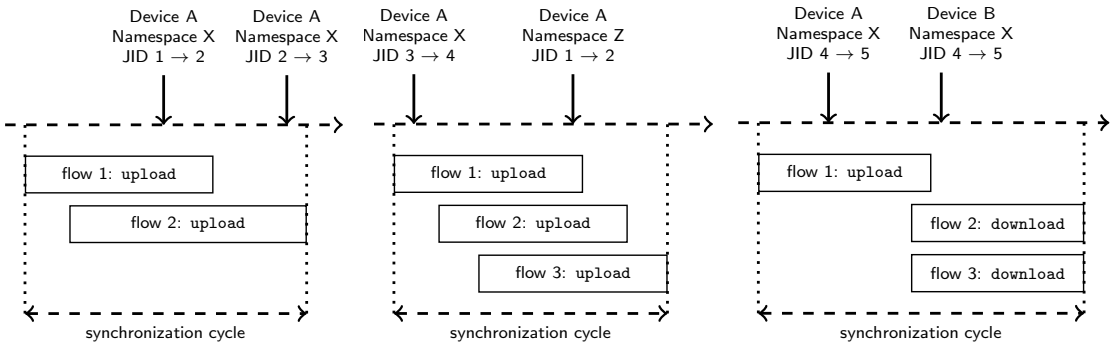
Our research goal one (RG1) aims at characterizing and modeling the behavior of users in cloud storage services. As we will show in Chapter 5, our model requires knowledge about (i) how frequently are contents (i.e., namespaces) updated, and (ii) how much data is updated (i.e., uploaded and/or downloaded) to/from Dropbox servers. Dropbox clients exchange notifications with servers once a minute when on-line. When a namespace is updated, additional Dropbox traffic appears in the network, and a notification message announcing the new JID is sent out by the client. We thus develop a method to estimate the size of each update, correlating flow volumes with notification messages. The method works in three steps.

First, we group all flows and heartbeat messages per client IP address. However, IP addresses provide only a coarse identification of user devices in Dropbox, since different devices may be present behind NAT – e.g., when sharing the same Internet access line at home. We leverage the heartbeat messages to identify precisely when each device becomes active and inactive, even if several devices are simultaneously connected from a single IP address.

Second, we group all flows overlapping in time for each anonymized IP address. We define the total overlapping period as a synchronization cycle. Thus, during a synchronization cycle, at least one device is performing update(s), i.e., uploading or downloading content. Considering the Dropbox protocols, notifications of updates must appear during synchronization cycles or shortly before/after their begin/end. We thus form a list of notifications – from the same anonymized IP address – announced during each synchronization cycle, considering a small threshold (60 seconds, in-line with the heartbeat interval) for merging notifications before/after the synchronization cycle.

Third, we evaluate each synchronization cycle to decide whether notifications represent uploads or downloads, and then to assign volumes to the notifications. We first label each data flow in the synchronization cycle as upload or download. We are able to do so because Dropbox typically performs either uploads or downloads in a single TCP flow, opening parallel flows when both operations need to be performed simultaneously [Bocchi et al., 2015a]. Therefore, we label flows simply by inspecting their downstream and upstream volumes, e.g., an upload flow has upstream much higher than downstream volume. Then, we inspect metadata in notifications (e.g., device and namespace IDs) to decide how to distribute upload and download volumes among notifications.

Figure 4.2 provides the most common cases we notice while assigning volumes to notifications. Figure 4.2a presents a scenario where a Device A performs several uploads



(a) Single NS (b) Multiple NS (c) NAT

Figure 4.2: Examples of the most common cases processed by our method to label notifications and to estimate the transferred volumes (NS stands for namespace).

to a single namespace. In this case we label all notifications as upload. Moreover, the total synchronization cycle volume is equally divided among the notifications. We apply the same methodology for symmetric cases, in which a unique device performs several downloads from a single namespace.

Often, a device updates multiple namespaces simultaneously. This case is illustrated in Figure 4.2b, in which Device A updates Namespaces X and Z during the same synchronization cycle. It occurs typically after a device connects to the Internet: if modifications in several namespaces are performed while the device is off-line, all namespaces must be synchronized at once. We are able to label all notifications as upload, but we cannot match single flows to notifications. Thus, for simplicity, we equally divide the total volume among notifications. Again, we apply the same methodology in symmetric cases with only downloads.

Finally, Figure 4.2c presents a NAT scenario, e.g., when Dropbox is used to synchronize various devices in the same house with a single IP address. Note that we observe upload and download flows from the same address, and notifications for different devices, but always for a single namespace. We infer the label of notifications based on timestamps: clearly, the first notification comes from the device uploading content, while remaining notifications come from devices downloading content. Thus, we assign the total upload volume to the first notification, and divide the total download volume equally among other devices.

Some corner cases prevent us from estimating volume for notifications. We ignore flows and notifications in complex NAT scenarios, where multiple devices modify multiple namespaces simultaneously. We also ignore a small number of orphan notifications and orphan flows – i.e., we see notifications, but no simultaneous storage flows, or vice-verse. Orphan notifications happen because the Dropbox client may

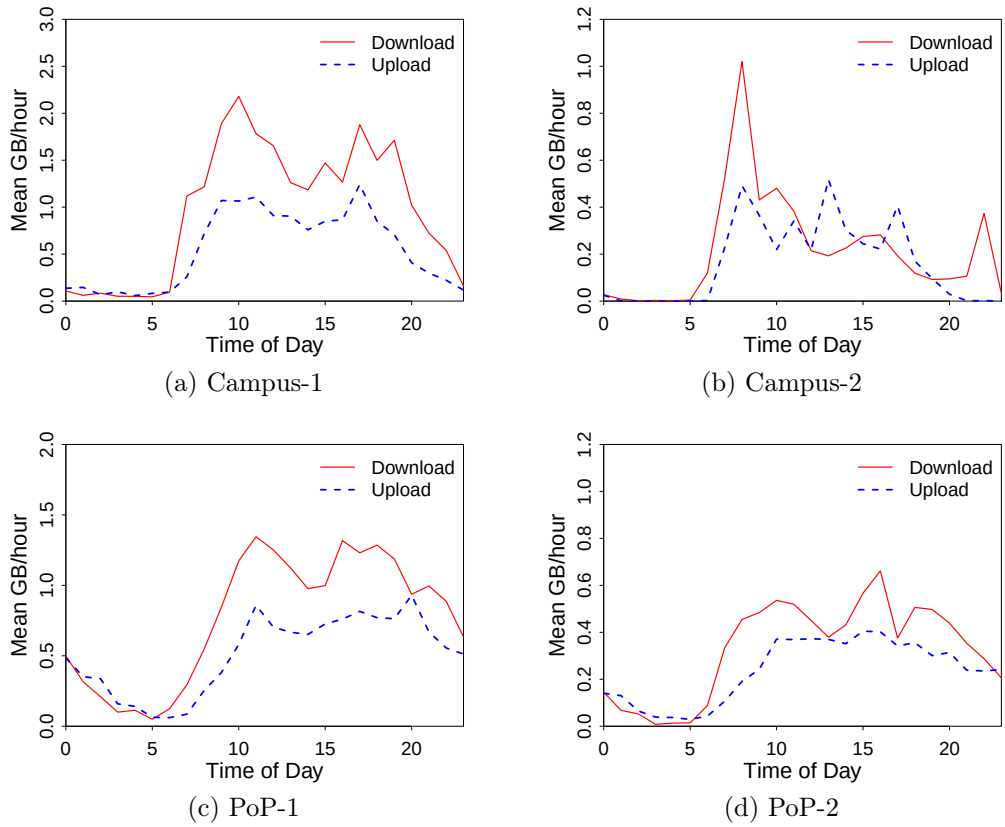
Table 4.2: Uploads and downloads volumes estimated for Dropbox by the data preparation method, considering the most common cases. Percentages refer to total upload and download volumes for each dataset shown in Table 4.1.

	Upload (GB)				Download (GB)				Total
Dataset	Single NS	Mult. NS	NAT	Total	Single NS	Mult. NS	NAT	Total	Traffic
Campus-1	1,137	141	25	1,304 (77%)	1,688	518	76	2,282 (60%)	3,586 (65%)
Campus-2	254	69	4	326 (53%)	349	90	3	443 (44%)	769 (47%)
PoP-1	2,135	217	252	2,604 (67%)	2,957	680	272	3,909 (57%)	6,514 (61%)
PoP-2	1,367	229	254	1,850 (77%)	1,569	569	419	2,558 (68%)	4,409 (72%)
Total	4,893	656	535	6,084 (71%)	6,564	1,858	771	9,193 (60%)	15,277 (64%)

generate no workload to storage servers when users perform local modifications, such as when files are deleted or renamed or when the `LAN Sync` actuates. Orphan flows and orphan notifications are also an outcome of artifacts in our data capture: packet loss during the capture might cause heartbeat messages to be missed. Finally, since we characterize namespace updates by observing Journal ID *transitions*, we ignore data flows that happen during the first time a device is seen in our captures.

Despite these artifacts, our method is able to associate most volume to notifications. We choose to be conservative and use only the data volume related to the cases illustrated in Figure 4.2, which we are confident about the volume associated with notifications. Table 4.2 shows percentages for each of these cases and the total volumes of upload and download as well as total traffic for the four networks. We associate 77% (60%) of the total volume of upload (download) with notifications in Campus-1, and 53% (44%) in Campus-2. Concerning PoP datasets, we associate 67% (57%) of the total volume of upload (download) in PoP-1, whereas that percentages are 77% (68%) in PoP-2. Percentages for downloads are consistently lower than for uploads thanks to new devices registered with Dropbox during our captures – those typically download lots of contents during the initial synchronization cycle and we ignore this volume. Differences in percentages across datasets are in-line with the presence of complex NAT scenarios in each network and the packet loss rate in the respective probes. Overall, we retain 64% of total Dropbox traffic for our analysis as shown in Table 4.2.

Figure 4.3 shows the bandwidth of Dropbox processed data for each network, given by GB per hour. Each curve represents the mean upload and download volumes per time of day, considering every day of each network period. As one can observe, the download is higher than the upload most of the time in all networks, even considering that the discarded download volume was higher than the upload one. The figure also shows the times of day when the service has the highest volumes, i.e., service peak times. In campuses, we observe the largest bandwidth peak in the morning and a



(a) Campus-1 (b) Campus-2

(c) PoP-1 (d) PoP-2

Figure 4.3: Bandwidth of Dropbox (processed data): curves represent the mean upload and download volumes per time of day for all days of each network period (note different scale for y-axes).

sharp drop at night. The download volume at night in Campus-2 is due to the use of Dropbox in students houses of this campus. In PoPs, in turn, we observe bandwidth peaks that are not very distinct and roughly distributed over the day. Notice upload and download volumes, at night, in PoPs are higher than in campuses. However, the largest bandwidth peaks in PoPs also occur during in the morning or afternoon. We conjecture it reflects the use of cloud storage for professional activity of users during daytime, e.g., home office, in those environments. The highest bandwidth peaks reach 5.6, 2.8, 2.9 and 1.4 mean GB/hour ($\pm$ 0.29, 0.27, 0.21 and 0.07 with 95% confidence intervals), all of them for download, in Campus-1, Campus-2, PoP-1 and PoP-2 respectively. Those mean peaks values are in agreement with each network population size.

4.4 Ethical Considerations

Datasets collected from networks, i.e., statistics about TCP flows and heartbeat messages, do not include any information that could reveal user identities, or information about the content users store in Dropbox. All payload data are discarded directly in the probe. All Dropbox IDs are numeric keys that offer no hints about users' identities. Additionally, any information about IP addresses, even anonymized before data preparation for analysis, cannot be made publicly available, according to the Non-Disclosure Agreement with the organizations' boards that provided those datasets.

Following the aforementioned procedure, our methodology to collect data is in agreement with the guideline proposed by Partridge and Allman [2016], which establishes as the main rule: *data collected by measurements studies cannot cause any tangible harm to any person's well being.* In other words, the dataset and results used in the study can not reveal any private or confidential information about individuals.

4.5 Summary

In this chapter, we described the methodology we adopted to collect and process Dropbox traffic in four different networks. We can summarize this process as follows:

- We collected a long-term dataset divided into two parts: one that summarizes all Dropbox flows seen in the network, and another one that registers each transition of the version number of namespaces (JID) present in client devices.
- We proposed a pre-processing methodology which takes these two datasets and estimates the volume of each update (i.e., namespace JID transition) by matching an update with the traffic volume that happened close in time.
- There were some cases which prevented us to accurately match the flows with the corresponding notification, such as many devices behind NAT and the first synchronization of devices. Having to face such cases, we chose to be conservative and use only data when we are confident about the volume of the update.

The resulting dataset is used to characterize and model user behavior in Chapter 5, to analyze data sharing and user activity in Chapter 6, and our cost-benefit analysis of cloud storage service in Chapter 7.

Chapter 5

Modeling User Behavior and Workload Patterns

In this chapter, we address our first research goal (RG1), which regards modeling user behavior and workload patterns of cloud storage services.

Cloud storage is currently very popular with many companies offering this kind of service, including worldwide providers such as Dropbox, Microsoft and Google. These companies as well as providers entering the market could greatly benefit from a deep understanding of typical workload patterns their services have to face in order to develop cost-effective solutions. Yet, despite recent studies of usage and performance of these services, the underlying processes that generate workload for them have not been deeply studied.

This chapter presents a thorough investigation of the workload generated by Dropbox customers. We propose a hierarchical model that captures user sessions, file system modifications and content sharing patterns. First, we describe our model assumptions in Section 5.1 and its components in Section 5.2. We then parameterize each component of our model from the datasets collected in four different networks (see Chapter 4) in Section 5.3. Next, in Section 5.4, we use the proposed model to drive the development of CloudGen, a new synthetic workload generator that allows the simulation of the network traffic created by cloud storage services in various realistic scenarios. Moreover, we validate CloudGen by comparing synthetic traces with actual data. Finally, in Section 5.5, we show CloudGen applicability by investigating the impact of the continuing growth in cloud storage popularity on data transference.

5.1 Model Assumptions

We target two aspects to build a model taking into consideration the complexity of the Dropbox client functioning in a realistic fashion. First, our model needs to represent users' behavior, including mechanisms to describe the arrival and departure of their devices, as well as the dynamics behind updates of Dropbox namespaces by the users. Second, from the users' behavior, the model needs to estimate the resulting workload seen in the network.

We make some assumptions to simplify the model. First, we assume Dropbox devices generate new content (i.e., updates in namespace) *independently* of any other device. That is, we assume the appearance of new contents that need to be uploaded to Dropbox servers is a process that can be modeled per device, without considering other devices from the same or other Dropbox users.

Second, we assume that the upload of content to servers and its propagation to several devices follow the Dropbox protocols described in Chapter 2 (Section 2.3). This implies our model follows two major design principles: (i) a device producing new content uploads it to the cloud immediately; (ii) all devices registering a namespace download updates performed elsewhere as soon as this is available at the cloud.

Third, we assume content uploaded to a namespace is always propagated to all other devices registering the namespace. In practice, users' actions might prevent a content from reaching other devices. For instance, users could manually pause content propagation in specific shared folders. More important, devices that stay off-line for long periods might return on-line after previously uploaded files are removed from Dropbox. We simplify our model by avoiding describing the evolution of namespaces in a per-file granularity. The model thus provides an upper bound for downloads caused by content propagation.

Finally, in this work we refrain from modeling the physical network or servers. We assume that data transmission or server delays are negligible. Modeling network conditions is out of our scope, and could be achieved by coupling our model to well-known network simulation approaches, such as the simulator developed by Yuksel et al. [2000].

5.2 Proposed Hierarchical Model

We propose a model composed by two parts: (i) the working dynamics of each Dropbox device in isolation; and (ii) the set of namespaces that is used to propagate content among devices.

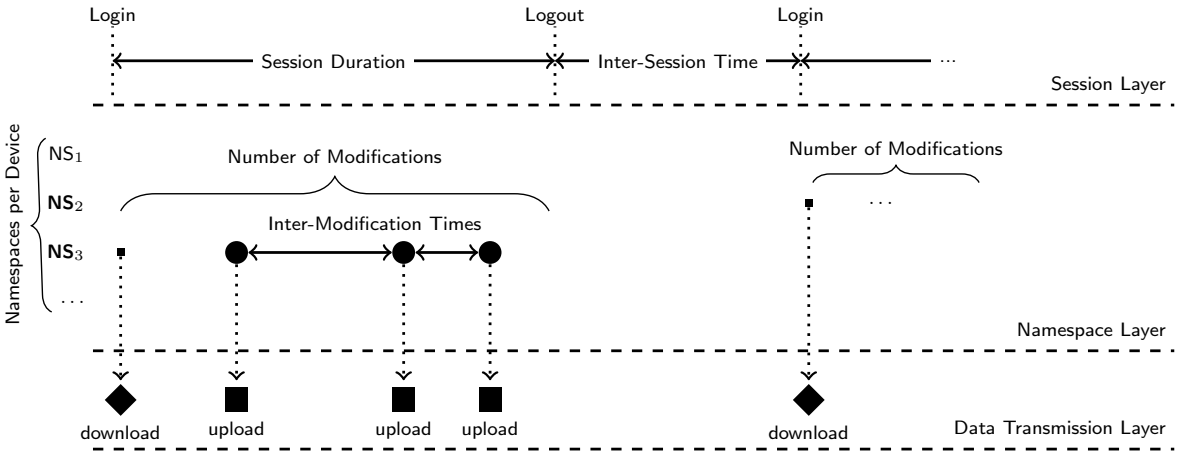


Figure 5.1: Hierarchical model of the cloud storage client behavior.

5.2.1 Modeling Device Behavior

Figure 5.1 depicts how our model represents a single client (i.e., Dropbox device). It captures three fundamental aspects of cloud storage clients: (i) the sessions of devices – i.e., devices becoming on-line and off-line; (ii) the frequency people make modifications in the file system that result in storage workload; (iii) the transmission of data from/to the cloud. Each aspect is encoded in a layer forming a hierarchical model for the cloud storage client behavior.

The *session layer* is the highest one in the hierarchy, describing the activity of devices. Thus, it is the one closest to the behavior of end users. While devices are active, they might generate workload to storage servers, which is encoded in lower layers of the model. Yet, active devices generate workload to control servers, e.g., to notify updates in namespaces. A session starts when the device logs in, and it lasts until the device logs out. We call the duration of this interval the *session duration*. The time between the client logout and its next login we refer to as the *inter-session time*. Note that Dropbox usually connects automatically in background in PC clients, and thus the inter-session time typically happens when devices are disconnected.

The *namespace layer* encodes the modifications happening in namespaces. Each device registers a number of namespaces, characterized by the *namespaces per device* component. Devices then may start several synchronization cycles. Synchronization cycles start because new content is produced locally and needs to be uploaded to the cloud, or because new content is produced elsewhere and needs to be downloaded by the device.

The start of uploads is regulated by three components: the *namespace selection*, the *number of modifications* and the *inter-modification time*. The first determines

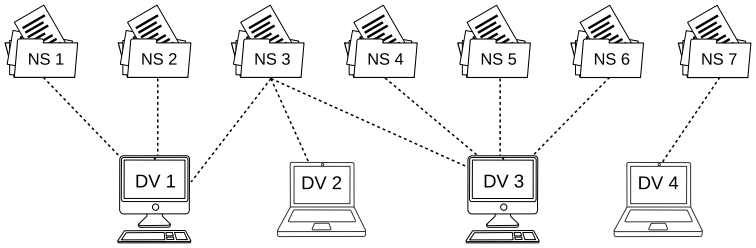


Figure 5.2: Example of the content propagation network (NS and DV stands for namespace and device respectively).

which namespaces of the device have modifications that result in uploads during the session. The second represents the total number of uploads during the session. The third models the interval between consecutive uploads. Downloads are governed by the behavior of peer devices in the content propagation network, which we will describe next.

Finally, at the *data transmission layer*, we characterize the size of the workloads locally produced at the device, which are sent to servers. The *upload volume* is the single component in this layer.

5.2.2 Modeling the Content Propagation Network

The modification (or addition) of any content in a namespace generates uploads, which, in turn, may trigger the content propagation, i.e., downloads. Content propagation is guided by a network representing how namespaces are shared among the several devices of a single user, or among devices of different users. Two components in the namespace layer describe the content propagation network.

First, the previously mentioned *namespaces per device* component specifies how many namespaces devices register. Second, a namespace is visible by a number of devices, characterized by the *devices per namespace* component. This scheme is represented in Figure 5.2, in which seven namespaces and four devices are depicted. Namespaces and devices have a many-to-many relationship, where each namespace is associated with at least one device and vice-verse.

When a device starts a new session, it first retrieves all updates in shared namespaces while it has been off-line. For instance, if a device has shared folders with third-parties, all pending updates in those shared folders are retrieved. This behavior is represented in Figure 5.1, in which we can see downloads happening when sessions start. Whenever a device uploads content to a namespace, the content propagation network is consulted to schedule content propagation. All peer devices that are on-line

retrieve the updates immediately, whereas off-line devices are scheduled to update when their current inter-session time expires.

5.3 Characterization

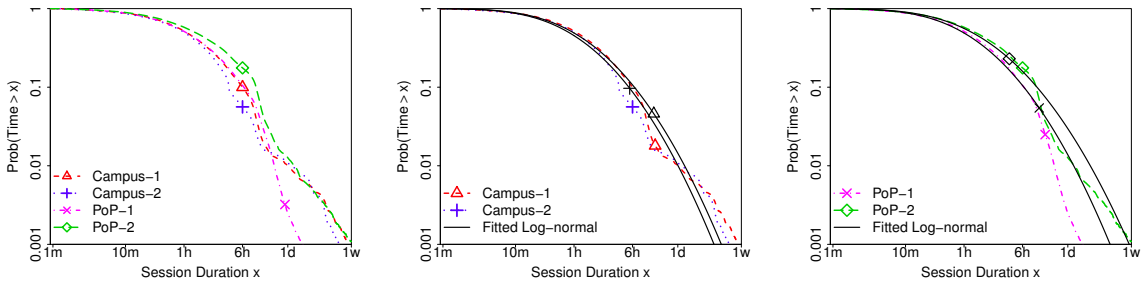
We characterize each model component presenting the statistical distribution that best fits our measurements. To our knowledge, no prior work fitted statistical distributions to the workload generated by users in cloud storage services. We focus on the working days in our data due to seasonal activity observed during weekends or holidays in campuses.[1] In this way, we apply the same characterization methodology for both campuses and PoPs datasets.

To achieve this characterization, we proposed a methodology organized into three steps. First, we use the Maximum-Likelihood Estimation (MLE) method [Venables and Ripley, 2002] to fit a set of well-known distributions of literature for each component from this component measured data. MLE is an optimization method that search for the parameters values of a given distribution that best fit the provided data. We then consider the following distributions to perform MLE: Normal, Log-Normal, Exponential, Cauchy, Gamma, Logistic, Log-Logistic, Beta, Uniform, Weibull, Pareto (continuous measured values); and Poisson, Binomial, Negative Binomial, Geometric and Hypergeometric (discrete measured values).

Second, we select three among the best-fitted curves (i.e., distributions) by MLE for each component. The criteria for this selection is the value the Kolmogorov-Smirnov distance statistic (KS) [D'Agostino and Stephens, 1986] (for continuous distributions) and the least square errors (LSE) [Jain, 1991] (for discrete distributions) of the fitted curves. The smaller those values are the more representative a distribution is for the measured data. We then select the three distributions with the smallest KS/LSE values. Third, the final decision of what candidate distribution best fits each component is taken by comparing visually the obtained best-fitted curves with the measured data curves both at the body and at the tail to support our decisions.

We show results next, while parameters of the best-fitted distributions are described in Appendix A.

[1] Modeling such seasonality is outside the present scope.



(a) All (b) Best Fit – Campuses (c) Best Fit – PoPs

Figure 5.3: Session duration. Note the logarithmic axes. Distributions are similar in different scenarios despite differences in the tails. The best fitted distribution is always Log-normal.

5.3.1 Session Layer

Session Duration:

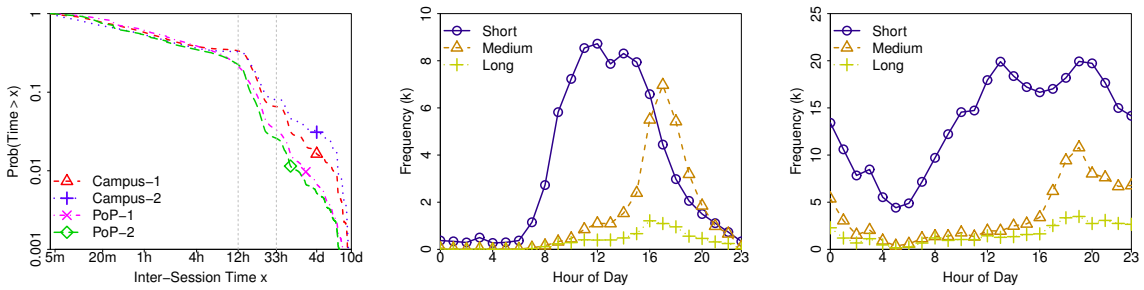
Figure 5.3 describes sessions duration by plotting the empirical Complementary Cumulative Distribution Functions (CCDFs) for each dataset. Figure 5.3a contrasts the datasets, whereas Figures 5.3b and 5.3c show separate plots for campuses and PoPs, including lines for the best-fitted distributions.

Focusing on Figure 5.3a, note how the majority of Dropbox sessions are short. The duration follows similar patterns in all datasets – e.g., at least 42% of the sessions are shorter than one hour. However, we observe that sessions are slightly shorter in campuses and PoP-1 than in PoP-2. Indeed, 90% of the sessions in campuses and in PoP-1 are under 6 hours, whereas 90% of the sessions in PoP-2 last up to 9 hours. This can be explained by the sole presence of FTTH users in PoP-2. Our results suggest that FTTH users tend to enjoy longer on-line periods than others. Note also how the number of very long sessions is smaller in PoP-1, where ADSL lines seem to provide less stable connectivity to users.

Despite the variations, we find that the Log-normal distribution describes well the data from all sites. Interestingly, Log-normal distributions have been used to model client active periods in other systems, such as in Peer-to-Peer [Stutzbach and Rejaie, 2006b].

Inter-Session Time:

In contrast to sessions duration, we find much higher variability in inter-session times. Our data indicate that inter-sessions are characterized by a mix of distributions that hardly fits a single model well. Intuitively, this happens due to the routine of users



(a) Inter-Session Time (b) Per Hour (Campus-1) (c) Per Hour (PoP-1)

Figure 5.4: Typical ranges composing the inter-session time distribution and the frequency we observe each range per time of the day in Campus-1 and PoP-1.

in working days in different environments, which includes short breaks during the day and long breaks overnight or across days. Moreover, a relevant portion of devices (20–40%) connects roughly daily to the service, suggesting the existence of different classes of users.[2]

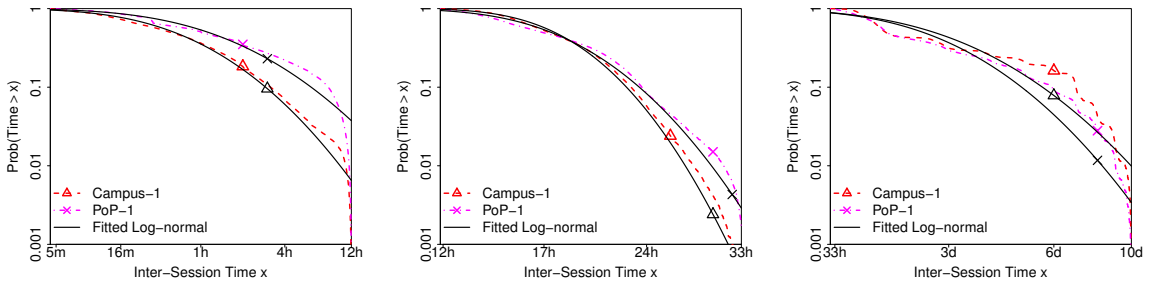
These patterns are illustrated in Figure 5.4a, which depicts CCDFs of inter-session times in the four datasets (note the logarithmic scales). We see a majority of *short inter-sessions* (67–76%) that are shorter than 12 hours, compatible with intraday breaks. A significant percentage of *medium inter-sessions* (20–28%) lasts between 12 and 33 hours. Those define primarily the overnight off-line intervals of devices. Finally, a small percentage of *long inter-sessions* with more than 33 hours ($\approx 5\%$) appears in the tails of the distributions, characterizing long breaks of devices.[3]

Interestingly, the intraday routine in each environment determines when inter-session ranges appear in practice. Figures 5.4b and 5.4c depict the frequency we observe each range in Campus-1 and PoP-1, respectively, considering the time of the day inter-sessions start. We can see that short inter-sessions are more common between 8:00 and 18:00 in Campus-1, but between 8:00 and 23:00 in PoP-1. Medium and long inter-sessions peak in the evenings in both cases, when devices disconnect to return next day or some days later respectively. The inter-session peak is in line with reduction of upload and download volumes, which we show in Figure 4.3, from a preliminary analysis of datasets in Section 4.3.

We opt for breaking the measurements into three ranges to determine the best-fitted distributions. We also vary the probability of observing each range when

[2] We leave for future work an investigation about classes of users in the service.

[3] Note that we ignore inter-sessions longer than 10 days to reduce effects of seasonality. Those are less than 3% of the inter-sessions in the original data.



(a) Short (b) Medium (c) Long

Figure 5.5: Inter-session times per range and best-fitted Log-normal distributions.

generating workloads in Section 5.4, taking into consideration the frequency of each range throughout the day, as shown in Figures 5.4b and 5.4c.

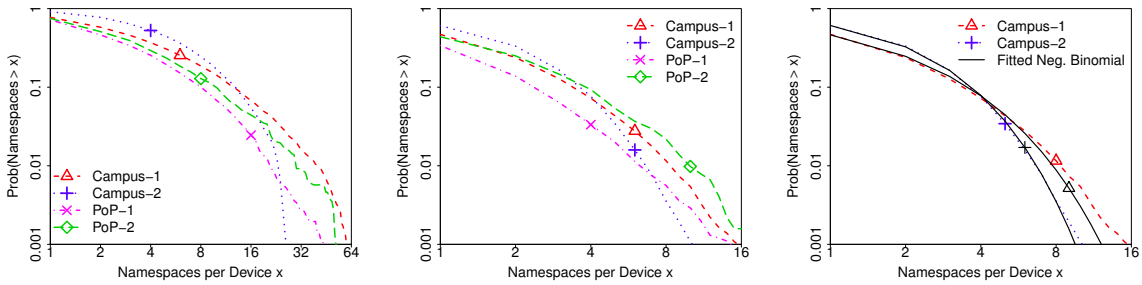
Figure 5.5 shows CCDFs with respective best fits for each inter-session range in Campus-1 and PoP-1. We observe similar behavior for CCDFs of Campus-2 and PoP-2. Log-normal is the best choice for all datasets in the three ranges. However, note how long inter-sessions present peculiar curves, pointing again to a complex process that is hard to be described with a simple model. Even if fits are not tight for this range, it has little impact on generated workloads, thanks to the low frequency (5%) long inter-sessions occurring in practice.

5.3.2 Namespace Layer

Namespaces per Device:

Figure 5.6a shows the CCDF of the number of namespaces per device. Results are in-line with findings of Drago et al. [2012]. The number of devices with a single namespace is small: around 10–22% in campus networks, and 25–35% in residential networks. Devices in campuses as well as those of users enjoying fast Internet connectivity at home have higher number of namespaces. Compare the line for PoP-2 (FTTH) with the one for PoP-1 (FTTH and ADSL) in the figure.

We however notice the existence of temporal locality when users modify namespaces. Indeed, although the number of namespaces per device is usually high, users tend to keep working on few namespaces in short time intervals. This effect is illustrated in Figure 5.6b, in which we plot the distribution of the number of namespaces per device, considering only namespaces that have at least one modification in arbitrary 1-week long intervals of our datasets, and ignoring devices with no modifications. We can see that most devices among those that have modifications produce new workloads



(a) All (b) Per Week (c) Best Fit – Campuses

Figure 5.6: Namespaces per device considering all namespaces (a) and only those modified in 1-week long intervals (b) with best fits in campuses (c). Note differences in x-axes.

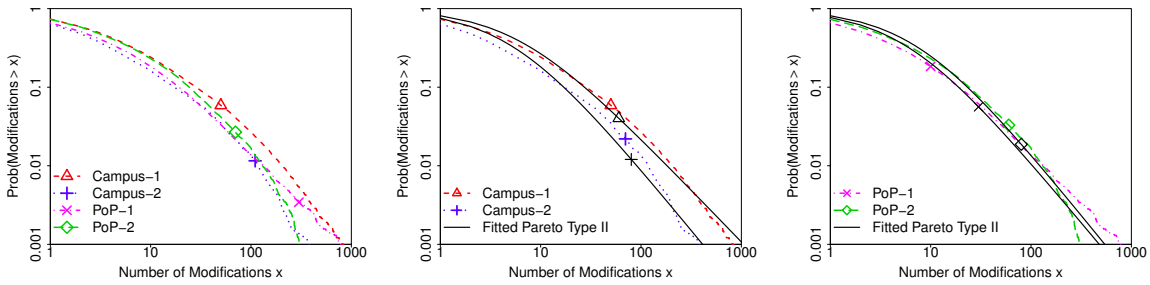
in a single namespace in a full week (i.e., 40–53% in campuses and 56–66% in PoPs), while 80% of the devices modify no more than 3 (campuses) or 2 (PoPs) namespaces per week.

In this chapter, we are interested in the workload in short time intervals (e.g., weeks), we refrain from characterizing the temporal locality of namespaces. For the practical proposal of generating workload, we compute the number of namespaces per device based on namespaces for which we observe modifications in one week periods, as in Figure 5.6b. Despite distinct patterns between campus and residential networks, Figure 5.6c shows that Negative Binomial distributions are the best fit in campus datasets. PoP datasets are omitted as they provide the same conclusion.

Namespace Selection:

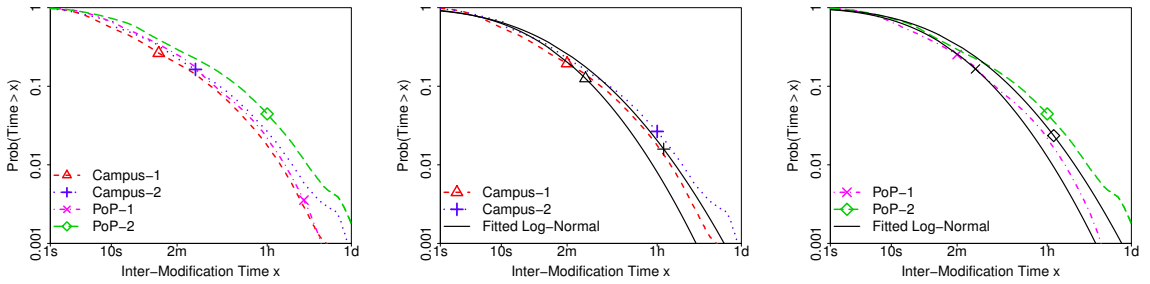
We derive the methodology to select which namespaces upload content during a session. We find that the vast majority of sessions (85–93%) have no uploads at all. Clients connect to Dropbox servers, check for possible updates, but do not modify any files locally, i.e., they do not upload content. The remaining sessions typically have modifications in a single namespace, with less than 2% of the sessions experiencing updates in multiple namespaces. This is expected, since the temporal locality implies that each namespace is changed in restricted periods of time.

Thus, for simplicity, we determine whether a session will see modifications by computing the probability of modifications in *any* namespaces (e.g., 15% in Campus-1 and 7% in PoP-1). Then, we select a random namespace for each session that has modifications.



(a) All (b) Best Fit – Campuses (c) Best Fit – PoPs

Figure 5.7: Number of modifications. Pareto Type II distributions are the best candidate.



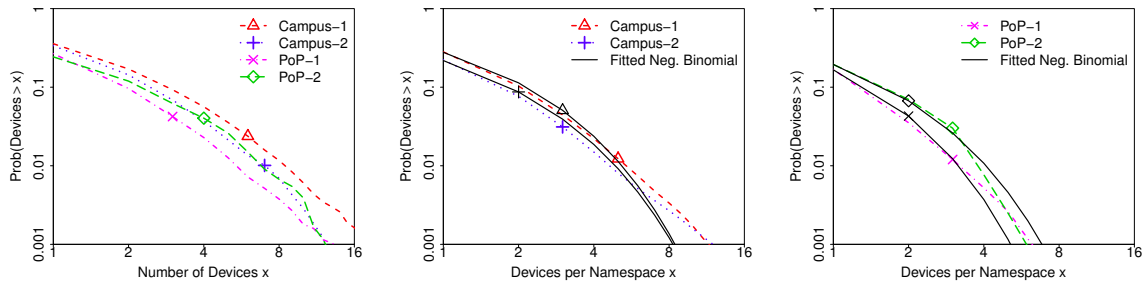
(a) All (b) Best Fit – Campuses (c) Best Fit – PoPs

Figure 5.8: Inter-modification time. Log-normal distributions are the best candidate.

Number of Modifications:

We characterized the 7–15% of sessions that have at least one modification in namespaces. Figure 5.7 shows CCDFs of the number of modifications per session.

In general, users make few modifications in namespaces per session. Figure 5.7a shows that, among sessions that have at least one modification, around 26% have a single modification, whereas 75% of them have no more than 10 modifications. However, we also observe a non-negligible number of sessions with lots of modifications. Focusing on the tail of the distributions in Figure 5.7a, notice that around 2% of the sessions have more than 100 modifications. Such heavy tailed distributions are well characterized by Pareto models. Indeed, we can see in Figures 5.7b and 5.7c that Pareto Type II distributions fit our data satisfactorily, despite differences in parameters per dataset.



(a) All (b) Best Fit – Campuses (c) Best Fit – PoPs

Figure 5.9: Devices per namespace (logarithmic scales). Most namespaces are accessed by a single device in the network. Negative Binomial distributions best fit our data.

Inter-Modification Time:

Figure 5.8 shows CCDFs for inter-modification times, which are computed considering sessions with at least two modifications. Figure 5.8a shows that short time intervals are frequent between modifications. For instance, at least 60% of the inter-modification times are shorter than 1 minute in all sites. This indicates that namespaces tend to present bursts of modifications. Indeed, sessions with large numbers of modifications are characterized by such bursts. However, we also observe inter-modification times of hours, which seems to represent the interval between bursts in long-lived sessions. The combination of inter- and intra-burst intervals result in distributions that are strongly skewed towards small values. As we can see in Figures 5.8b and 5.8c, which depict CCDFs together with the best fits, Log-normal distributions describe our data best.

Devices per Namespace:

The last component of the namespace layer characterizes how many devices have access to each namespace. This component also represents the popularity of a given namespace in the cloud storage service, as it provides the number of devices which a content (i.e., modification or update) in the namespace is propagated to. Our data provide the means to estimate the number of devices per namespace for the user population in the networks we monitor. Namespaces may have higher number of devices thanks to devices connected outside the studied networks. Thus, our characterization provides a lower-bound for the actual popularity of a namespace or content generated in this namespace.

Figure 5.9a shows CCDFs of the number of devices per namespace computed for four networks. In contrast to the number of namespaces per device, devices per

namespace does *not* vary greatly when observing all namespaces registered in the devices or only those users actually modify in a limited time interval – i.e., it is not affected by the temporal locality of namespaces. Thus, results in the figure are computed based on all namespaces observed in each dataset.

We find that the number of devices per namespace in campuses is higher than in PoPs: 33–36% of namespaces are shared by multiple local devices in campuses, whereas that percentage is 24–26% in PoPs. The probability of observing a namespace with high popularity in campuses is then slightly greater than in residential networks, as we can see in Figure 5.9a. For example, 1% of namespaces in Campus-1 propagate content for more than 8 devices in this network.

To understand the popularity of namespaces in campuses and PoPs further, we analyze the number of devices per namespace considering which users own the devices – i.e., disregarding devices of the same user. We observe about 17% of namespaces are shared by multiple local users in campuses, whereas that percentage is 7–8% in PoPs. Thus, the percentage of namespaces registered in devices of different users is much lower in residential networks. These results reinforce that content sharing is more important in the academic scenario, where a more consistent community of users is aggregated. Our residential datasets, instead, capture a scenario in which multiple devices of few users in a household are synchronized.

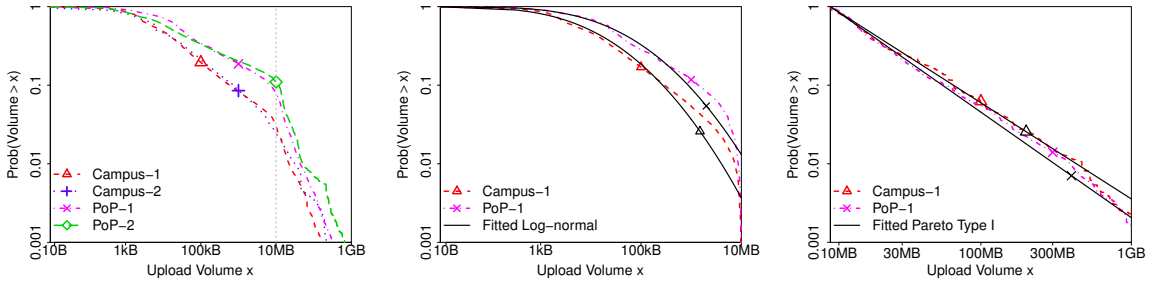
Despite differences in parameters, we find that Negative Binomial distributions are the best fit for the number of devices per namespace in all sites, as we show in Figure 5.9b and Figure 5.9c.

5.3.3 Data Transmission Layer

Finally, we study the volume of upload workloads. We characterize the variable using only the subset of notifications of updates for which we are confident about the upload volume – i.e., those cases described in Chapter 4 Section 4.3 (see Figure 4.2). We expect to select a random subset of notifications, thus providing a good estimation for the size of each workload.

Figure 5.10a shows CCDFs of upload sizes. Interestingly, uploads are larger in PoPs than in campuses. We conjecture that this happens because users at home often rely on cloud storage for backups and for uploading multimedia files, while a high number of small uploads is seen in campus due to collaborative work.

The figure clearly shows knees in the distributions at around 10 MB, which divide them into distinct curves for all datasets. Uploads with volume under 10 MB are the majority, e.g., 97% and 89% in Campus-1 and PoP-1, respectively. This suggests



(a) All (b) Best Fit $\leq$ 10 MB (c) Best Fit $>$ 10 MB

Figure 5.10: Upload volumes. Data are best-fitted by different models at the body (Log-normal) and at the tail (Pareto Type I). Uploads at PoPs are larger than in campuses.

to break the measured data into two ranges, determining the best fit for each range independently. Figure 5.10b shows the empirical distributions and best fits for the measured volumes up to 10 MB, whereas Figure 5.10c shows curves for the volumes greater than 10 MB. Only Campus-1 and PoP-1 are shown to improve visualization. In all datasets, Log-normal distributions are the ones that better describe our data up to 10 MB (see Figure 5.10b). On the other hand, uploads with volume greater than 10 MB are best fitted by the Pareto Type I distribution (see Figure 5.10c). These results are in agreement with the generative model for file sizes proposed by Mitzenmacher [2004], as discussed in Chapter 3 Section 3.1.2.

5.3.4 Discussion

Our client behavior model has several components. At the highest session layer, client behavior is represented in terms of session duration and inter-session time. At the namespace layer, the model captures the modifications in namespaces by selecting a number of namespaces in each session and using the number of modifications and the inter-modification time to schedule modifications. Modifications are propagated by consulting the content propagation network. Finally, at the lowest data transmission layer, modifications are represented by the upload volume they produce.

While we have designed the model and characterized its parameters using Dropbox PC clients as a reference, we believe the model is mostly applicable to other cloud storage services and types of terminals (e.g., mobile terminals such as smartphones). This is because most of the model components capture actions of the Dropbox PC client that are driven by human behavior – e.g., users turning on and off the PC and connecting to Dropbox.

It is however clear that the parameters that characterize the components likely vary in other scenarios. For example, the typical usage of Dropbox on PCs and on mobile terminals are known to be different [Bocchi et al., 2015b]. Thus, even if Dropbox clients would be implemented exactly in the same way in all terminals, variables such as session duration would naturally differ among terminals.

Moreover, since we assume the synchronization protocol works as in Dropbox (see Section 5.1), minor tweaks may be necessary to adapt our model to other cloud storage clients (e.g., Google Drive or OneDrive). A clear example is in the content propagation network: our model spreads updates in shared folders automatically when peer-devices are on-line. This is a typical behavior of Dropbox PC clients that is not seen in every alternative. Yet, some tuning of the propagation logic would adapt the model to other protocols. For example, our model can adopt different levels of propagation for each update, where not all but only a fraction of on-line devices receive the update.

Finally, we highlight that despite variations in the estimated parameters, explained by differences in usage scenarios, we found the same distributions for each component of our model in four datasets. These results suggest our model captures, to a good extent, the behavior of users when using the Dropbox client.

5.4 CloudGen– Cloud Storage Traffic Generator

In this section, we present CloudGen, a synthetic workload generator for cloud storage. We overview CloudGen in Section 5.4.1, and validate it in Section 5.4.2.

5.4.1 Overview

We implement CloudGen based on our user behavior hierarchical model for the cloud storage client. CloudGen is also inspired by Dropbox user's behavior and content propagation network, which is similar to other cloud storage clients, e.g., Microsoft Onedrive. In the following, we discuss algorithms that show the main procedures and heuristics we applied to such implementation.

Algorithm 1 shows the initial procedures of CloudGen. This algorithm receives as input parameters for the set of distributions characterizing the model, as well as execution parameters, such as the synthetic workload duration, the target number of devices and namespaces. These parameters can be freely set, for example, to determine the population size to be evaluated. As outcome, CloudGen produces a synthetic trace indicating when each device is active and inactive, the series of modifications in namespaces (i.e., updates), and the data volume of modifications. Therefore, we

envisage the use of CloudGen in a variety of scenarios, e.g., to evaluate the traffic users in a network produce, or to estimate the workload a given number of servers (or VMs) provisioned by a service provider can face in different scenarios.

Algorithm 1: CloudGen

```
   Data: d, n, p, P, G as # of devices, # of namespaces, period by # of days, parameters of
         probability distributions and the content propagation network respectively
   Result: synthetic trace output
 1 // load the network from a trace
 2 if G = ∅ then
 3     G ← GenerateNetwork(d, n, P)
 4 // Priority queue of events and we start without any events
 5 Q ← ∅
 6 // simulation start at a day (whatever day in UNIX Epoch time)
 7 t ← START
 8 // start up the event simulation by creating device on-line events
 9 for each device d ∈ D[G] do
10     insert(Q, DeviceON(t, d))
11 // we iterate over events until the simulation time reaches the limit (days)
12 while t < START + p × 24 × 3600 do
13     e ← Extract-Min(Q)
14     t ← e.time
15     for each event i ∈ Execute(e) do
16         Insert(Q, i)
```

CloudGen starts by setting up the content propagation network, i.e., the structure that represents how namespaces are shared among devices (see Figure 5.2). CloudGen can either receive a manually defined network as input parameter or generate a network. Algorithm 2 shows the function *GenerateNetwork*, which receives as input parameters the number of devices (d), the number of distinct namespaces (n) and the set of parameters of probability distributions. As outcome, this function produces a bipartite graph $G = (D, N, E)$ composed of sets of devices, namespaces and the edges connecting both. This function follows two steps. First, it derives the number of distinct namespaces (n) from the number of devices (d), using the mean number of namespaces per device (n_d) and the mean number of devices per namespace (d_n), when value of n is not provided (lines 1-4). The expected number of distinct namespaces is given by $n = d \times n_d/d_n$. As an example, our data provide an average ratio of distinct namespaces per device (i.e., n_d/d_n) of 1.36 and 1.45 for Campus-1 and PoP-1, respectively.

Second, the function *GenerateNetwork* associates devices to namespaces. For that, it generates a random network based on method proposed by Goh et al. [2001]. Basically, this method works as follow: (i) the function first picks one namespace for each device to make sure all devices have at least one namespace (lines 14-21); then (ii) it complements the number of devices of each namespace (sampled from

the distribution of devices per namespace), by selecting random devices with the probability proportional to the number of namespaces in the devices (sampled from the distribution of namespaces per device), as shown in lines 22-42.

Algorithm 2: GenerateNetwork

```
   Data: d, n, P as # of devices, # of namespaces and parameters of probability distributions.
   Result: G = (D, N, E) a bipartite graph for network of devices, namespaces and their edges.
 1 if n = ∅ then
 2     n_d ← NegBinomialMean( P[NMS_PER_DEV] )
 3     d_n ← NegBinomialMean( P[DEVS_PER_NM] )
 4     n ← d × n_d/d_n
 5 N ← Namespace(n)
 6 // compute probability of edges leaving namespaces
 7 L ← ∅
 8 for n ∈ N do
 9     // sample the number of devices per namespace
10     degree[n] ← NegBinomialRandNumb( P[DEVS_PER_NM])
11     for i ← 1 to degree[n] do
12         Insert( L, n )
13 S_n ← Stack( shuffle( L ) )
14 D ← Device(d)
15 // create set of undirected edges from devices to namespace and vice-versa
16 E ← ∅
17 for d ∈ D do
18     degree[d] ← NegBinomialRandNumb( P[NMS_PER_DEV] )
19     // pick a random namespace for each device
20     v ← pop(S_n)
21     E ← E ∪ (d, v)
22 // then we complement the number of devices per namespace
23 // we skip devices with degree 1 in a first pass, since they just got 1 namespace
24 r ← 1
25 // we might have less edges leaving devices than necessary
26 while not StackEmpty(S_n) do
27     // compute probability of edges leaving devices from stack S_d
28     L ← List()
29     for d ∈ D do
30         for i ← 1 to degree[d]-r do
31             insert( L, d )
32     S_d ← Stack( shuffle( L ) )
33     while not(StackEmpty(S_n) or StackEmpty(S_d)) do
34         u ←Pop(S_n)
35         v ←Pop(S_d)
36         // we are lazy and skip the unfortunate repetitions
37         if not (u ∈ Adj[v]) then
38             E ← E ∪ (u, v)
39         else
40             Push(S_n, u)
41     // now we use devices with degree 1 to complement devices per namespace
42     r ← 0
```

Now we focus again on Algorithm 1 from line 4. Notice that CloudGen generates workloads following a classic discrete-event simulation approach. It creates a sequence of events to mark devices on-line and off-line (i.e., sessions login and logout). For that, a priority queue is used to schedule events, starting from an on-line event for each

device. As soon as extracted from the queue and executed, the on-line event triggers other associated events to be enqueued. The times of on-line and off-line events are determined by sampling the sessions duration and the inter-session time distributions, until the next on-line event of all devices falls outside the synthetic trace duration. In this case, oldest events are extracted first from the priority queue, which guarantees execution of events by their creation time order.

Algorithm 3 shows execution of an on-line event of device. Notice the event starts after sampling an intersession time (line 3). In this way, devices present distinct starting point for on-line events. Modifications events can happen depending on the probability of occurring modifications in the session. On the other hand, each on-line event is always associated with a device off-line event. Algorithm 4 shows execution of a device off-line event, which, in turn, trigger the next on-line event of the device (the last line).

Algorithm 3: Execution of On-line Event of Device

Data: d, t, P as the device, time stamp and probability distributions parameters associated with the On-line event
Result: L as a list of new events associated with the current event

```
1  // On-line event starts after the intersession duration
2  r ← IntersessionRange(t)
3  t ← t+ LognormalRandNum( P[INTERSESSION,r] )
4  d.on ← True
5  // On-line event time is used to schedule synchronizations for the device
6  d.onTime ← t
7  print( "DevOnline", t, d.id )
8  // start the list of events without any associated event
9  L ← ∅
10 // sample the session duration
11 s ← LognormalRandNum( P[SESSION] )
12 // check whether we add modifications events
13 if random.uniform() > P[MODIFICATION_PROB] then
14     // pick up the random namespace
15     n ← Random(Adj[d])
16     // generate a list of modification events
17     M ← Modifications(t, d, P, s, n)
18     L ← L ∪ M
19 // append to the list the device off-line event that starts after session duration
20 insert( L, DeviceOFF(t + s, d, P) )
```

Algorithm 4: Execution of Off-line Event of Device

Data: d, t, P as the device, time stamp and probability distributions parameters associated with the Off-line event
Result: L as a list of new events associated with the current event

```
1 d.off ← True
2 print( "DevOffline", t, d.id )
3 // start the list of events without any associated event
4 L ← ∅
5 // append to the list the device on-line event
6 Insert( L, DeviceON(t, d) )
```

We still focus on Algorithm 3 to discuss some procedures of execution of a device on-line event. It is worth noting, to compute inter-session times, CloudGen samples from the inter-session ranges (i.e., short, medium or long) based on their probabilities throughout the day (line 2). However, CloudGen enforces only medium inter-sessions or long inter-sessions in night periods.[4] As such, CloudGen mimics the behavior shown in Figure 5.4, which results in a sharp decrease in the number of new sessions at night, and multiple-day breaks for some devices. Note that while CloudGen captures intraday variations, thanks to the procedure to select inter-session times, other seasonal patterns (e.g., weekly or yearly) are intentionally left out for the sake of simplicity.

For each on-line event, CloudGen decides whether new events are required by following our namespace selection strategy (lines 12-18): first, it evaluates the probability that any namespaces in the session are modified. When modifications should occur, CloudGen randomly selects a namespace to execute modifications. CloudGen then determines how many modifications to place in the session with the function $Modifications$. It does so by matching the distribution of sessions duration with the distribution of the number of modifications per session. Basically, the longer sessions are, the higher are the chances they experience lots of modifications. Then, the series of modification events is scheduled using the distribution of inter-modification times in a best-effort fashion: samples of inter-modification times determine the time of the next modification in the session; if the session duration is reached before scheduling all modifications, the remaining modifications are scheduled starting from the session beginning again.

Algorithm 5: Execution of Modification Event of Device

Data: d, t, n, P as the device, time stamp, namespace and probability distributions parameters associated with the modification in namespace.
Result: L as a list of new events associated with the current event

```
1  // determine the volume of the change
2  v ← VolFromMixDist(P[MOD_VOLUME])
3  print( "Upload", t, d.id, n, v )
4  // prepare content propagation list of events, i.e., downloads output
5  L ← ∅
6  for each device i in Adj[n] do
7      if i ≠ d then
8          // if on, propagate content now
9          if i.on then
10             Insert(L, SyncDevice(t, i, v))
11             // if off, scheduled propagation
12         else
13             Insert(L, SyncDevice(d.onTime, i, v))
```

[4] At the time of this writing, CloudGen uses a parameter to represent a percentage of devices that access the service daily, and enforces medium inter-sessions in night periods for those devices. For future work, we intend to extend CloudGen to represent classes of users from this parameter.

Finally, Algorithm 5 shows execution of a modification event. CloudGen assigns the upload volume to each modification and, eventually, schedules download events using the content propagation network and the information about on-line status of peer devices. Modifications that happen in shared namespaces are then propagated to all peer devices. The synthetic trace generated also include a sequential version (JID) for each modification in namespaces. However, we omit those details in Algorithm 5 for the sake of clarity, although such procedures are included in CloudGen source code. CloudGen is available as open source software at `http://cloudgen.sourceforge.net`.

5.4.2 Validation

Methodology:

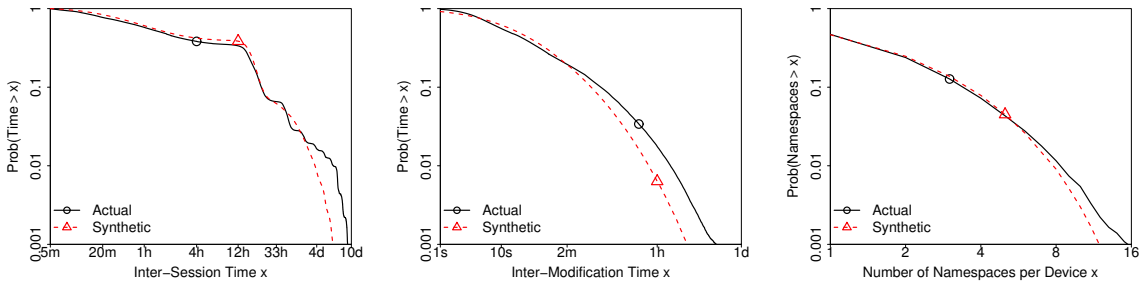
We validate CloudGen by comparing synthetic and actual traces. To help readability, we show validation results only for Campus-1 and PoP-1, representing campuses and ISP respectively, since similar results are obtained for all other datasets. We first determine the input number of devices. To remove possible effects of the registration and removal of devices to/from Dropbox, we pick several weeks of data in each dataset and count the number of active devices per week. Then, we execute CloudGen entering the mean number of active devices per week in each dataset, together with the respective distribution parameters characterized in the previous section.

We set CloudGen to generate synthetic traces of two weeks, to make sure all ranges of inter-session times are used in the simulation. Moreover, we discard the initial week to remove simulation warm-up effects, as well as weekends since our model does not include such seasonality. We repeat the procedure sixteen times for each dataset, to obtain different samples of synthetic traces.

We then contrast synthetic traces to real datasets. We focus on two types of metrics: (i) *Direct metrics* to validate that synthetic traces follow the distributions that define our model – e.g., we compute synthetic inter-session times to analyze the impact of simplifications we made to build CloudGen; (ii) *Indirect metrics* that are not explicitly encoded in our model – e.g., we use the aggregated workload devices produce in pre-determined periods (e.g., per day) to check whether synthetic traces approximate well the workload users create to a cloud storage service.

Direct Metrics:

We recompute all components of our model characterized in Section 5.3, now using synthetic traces, to validate CloudGen implementation. We show results only



(a) Inter-Session Time (b) Inter-Modification Time (c) Namespaces per Device

Figure 5.11: Comparison of model components in actual and synthetic traces in Campus-1.

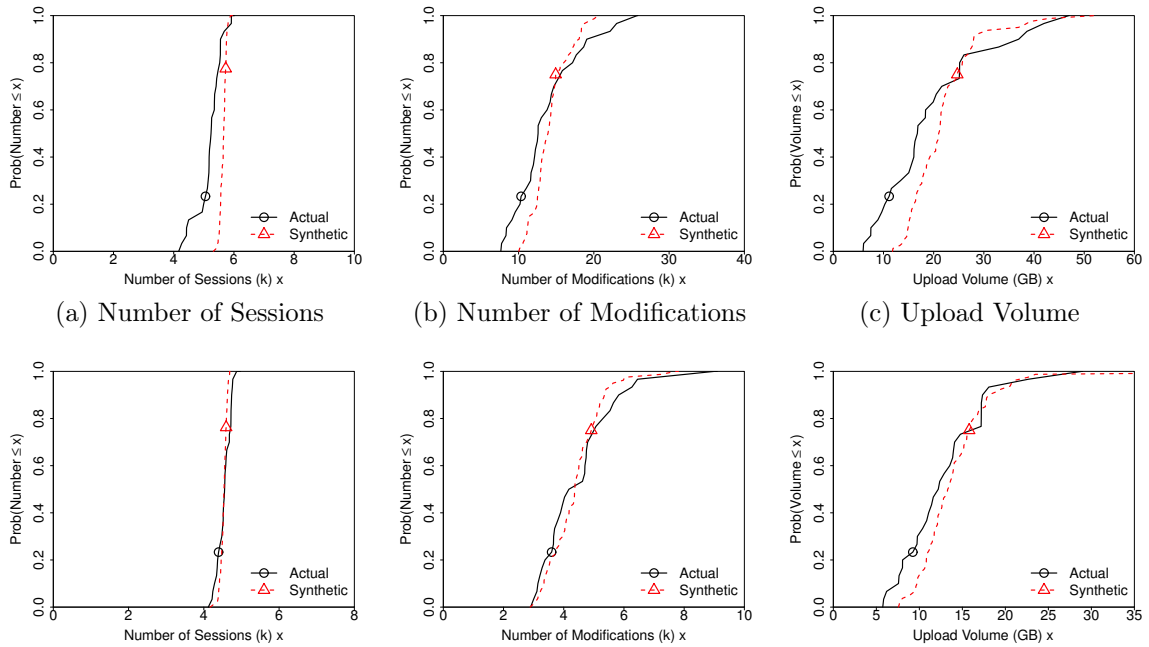
for Campus-1 in this first test, since other datasets lead to the same conclusions. We also omit plots for sessions duration, number of modifications per session, upload volume and devices per namespace, since these variables are used directly by CloudGen, without any specific design decision to simplify implementation.

Figure 5.11 summarizes other metrics. We depict in each plot the empirical distribution found in actual traces, along with the distribution extracted from the synthetic traces.

Inter-sessions times are depicted in Figures 5.11a. We can see that synthetic and actual distributions are tightly close to each other. Synthetic traces present the three-range pattern seen in actual data, with correct proportions among the ranges. Thus, we can conclude that our strategy to select inter-session ranges works well in practice.

Figure 5.11b depicts inter-modification times. Since inter-modification times are used only to distribute modifications in the sessions, and we adopt heuristics to match the number of modifications and the sessions duration, inter-modification times are prone to errors due to simplifications in our method. Indeed, synthetic traces present slightly more short inter-modification times than the actual ones. Still, the artifact affects less than 10% of the measurements, noticeable in the tail of distributions.

Finally, Figure 5.11c presents the number of namespaces per device, which is used indirectly as a weight when sampling devices for the content propagation network. We can see that synthetic and actual traces slightly diverge in the tail of the distributions. This difference is, however, compatible with the distribution characterized for the variable. In fact, our method to create content propagation networks approximates well both distributions that characterize the Dropbox network.



(a) Number of Sessions (b) Number of Modifications (c) Upload Volume

(d) Number of Sessions (e) Number of Modifications (f) Upload Volume

Figure 5.12: CDFs of indirect metrics (per day) for synthetic and actual traces in Campus-1 (1st. row) and PoP-1 (2nd. row). CloudGen slightly overestimates the workload.

Overall, we conclude that our implementation is satisfactory and produces synthetic traces similar to datasets used as input to our model.

Indirect Metrics:

Next we validate CloudGen focusing on three indirect metrics: the number of sessions, the number of modifications in namespaces and the total upload volume. We compute the metrics after aggregating the activity of all devices per day. Such metrics summarize the overall workload the network and servers need to handle in a time interval, given a fixed population of devices. Our goal is to understand whether CloudGen realistically reproduces the metrics when it is fed with the number of devices seen in actual traces.

Figure 5.12 presents CDFs for the indirect metrics. Top plots are relative to Campus-1, while bottom plots depict results for PoP-1. Other datasets are omitted again since they show similar patterns.

Figures 5.12a and 5.12d show that the number of sessions per day in synthetic traces is close to actual values. Focusing on Figure 5.12d, note how the synthetic distribution captures the overall trend in actual data for PoP-1. Mean values for

synthetic and actual traces are 4,542 and 4,551 in PoP-1, respectively. The number of sessions per day is slightly off actual values in Campus-1: mean of 5,649 for the synthetic trace and 5,219 for the actual one (i.e., synthetic overestimates by 8%), and we see more variability in actual traces. This happens because (for simplicity) CloudGen generates similar workloads for all weekdays, while campus routine clearly varies during the week – e.g., fewer devices are seen active on Wednesdays and Fridays in Campus-1. Thus, in environments with very high mobility of users, such as in campus networks, CloudGen approximates well mean values, but smooths the number of sessions per day.

Figures 5.12b and 5.12e show distributions for the number of modifications per day. Synthetic traces follow the overall trend in actual data again, in particular regarding mean values. The mean numbers of modifications per day are 14,174 (synthetic) and 13,884 (actual) in Campus-1 (synthetic overestimates by 2%), and 4,286 (synthetic) and 4,530 (actual) in PoP-1 (synthetic underestimates by 5%). Although mean values are closer in Campus-1 than in PoP-1, more variability can be noticed in Campus-1 in Figure 5.12b, once more, thanks to the heterogeneous users' routine in the environment throughout the week.

Figures 5.12c and 5.12f show distributions for the total upload volume per day in Campus-1 and PoP-1, respectively. We can see that CloudGen overestimates the upload volume per day. Mean volumes per day are 22 GB (synthetic) and 20 GB (actual) in Campus-1 (synthetic overestimates by 10%), and 14 GB (synthetic) and 13 GB (actual) in PoP-1 (synthetic overestimates by 8%). Synthetic mean volumes are slightly higher than actual mean volumes, despite the visual similarity of the distributions in the figures, due to the tails of the synthetic volumes – i.e., few days with very high workloads in synthetic traces. Yet, synthetic values are very close to the actual ones.

We conclude that synthetic workloads capture the main trends in actual traces. CloudGen provides conservative estimates of the upload traffic of a population of devices, because of the heavy tailed Pareto distributions characterized for the number and volume of uploads. CloudGen reconstructs reasonably well the aggregated workload in typical campus and residential networks, even though its design is based on a per-client workload model, which is challenge for a complex system as Dropbox. CloudGen is a valuable tool, helping systems administrators or developers to easily provision traffic requirements for cloud storage services.

5.5 Future Scenarios for Cloud Storage Usage

In this section, we illustrate the use of CloudGen by exploring how increases in popularity of cloud storage services affect network traffic. Section 5.5.1 describes the evaluated scenarios, whereas Section 5.5.2 presents results of CloudGen application.

5.5.1 Scenarios

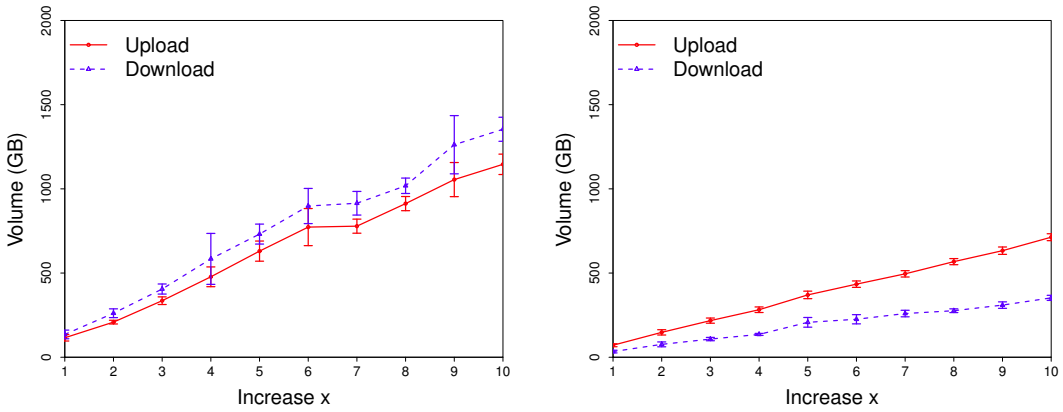
Our goal is to obtain insights on the traffic volume in near future scenarios where, due to the increasing popularity of cloud storage, we will see a larger number of users adopting such services, more devices being registered and, eventually, more content sharing through shared namespaces. We define three scenarios to explore how network traffic would behave with those trends:

1. *Larger Population*: This scenario studies how Dropbox user population growth affects network traffic. We increase the number of devices, evaluating how the network traffic changes with up to 10-times more devices than what is observed nowadays in edge networks.

2. *Larger Sharing*: We simulate a scenario where users share more content through cloud storage. We achieve that by increasing the mean number of devices per namespace. More precisely, we manually determine the parameter r of the Negative Binomial distribution (see Appendix A) that leads to the desired mean number of devices per namespace of each experiment.

3. *Larger Population and Sharing*: We combine the previous two scenarios, to simulate a future where more devices are registered to Dropbox, while users also become more prone to share content.

All remaining CloudGen parameters are kept constant in each scenario. We present results for CloudGen parameterized with Campus-1 and PoP-1 distributions only, since other datasets result in similar conclusions. We perform 16 experiment rounds for each configuration, recording upload and download volumes. We compute the metrics for a one-week long interval, after discarding CloudGen warm-up period and weekends. Finally, for the sake of simplicity, we assume a static and closed population of devices in these experiments: all uploaded content is propagated in the local network and produced only by devices within this network.



(a) Campus-1 (b) PoP-1

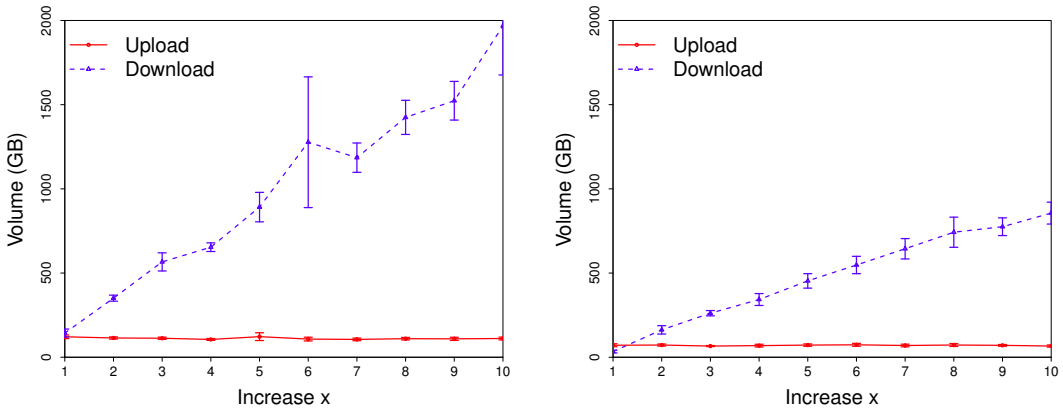
Figure 5.13: Mean upload and download volumes in larger population scenarios.

5.5.2 Traffic Analysis

Figure 5.13 presents mean upload and download volumes as number of devices is varied (larger population scenario). Error bars mark 95% confidence intervals.

We can see that upload and download volumes grow roughly linearly with the number of devices in the network – e.g., note how there is around twice as much upload when doubling the device population in both Campus-1 and PoP-1. Interesting, the download volume is greater than the upload one in Campus-1, whereas the opposite pattern is observed in PoP-1. Moreover, download volume grows slower than the upload volume with PoP-1 parameterization. This happens because a higher number of devices per namespace is observed in Campus-1 than in PoP-1 (Section 5.3.2). Thus, uploads in the campus more likely trigger several downloads. Overall, we conclude that the network traffic of larger device populations can be trivially estimated, provided that sharing behavior of users is known and remains constant as the population increases.

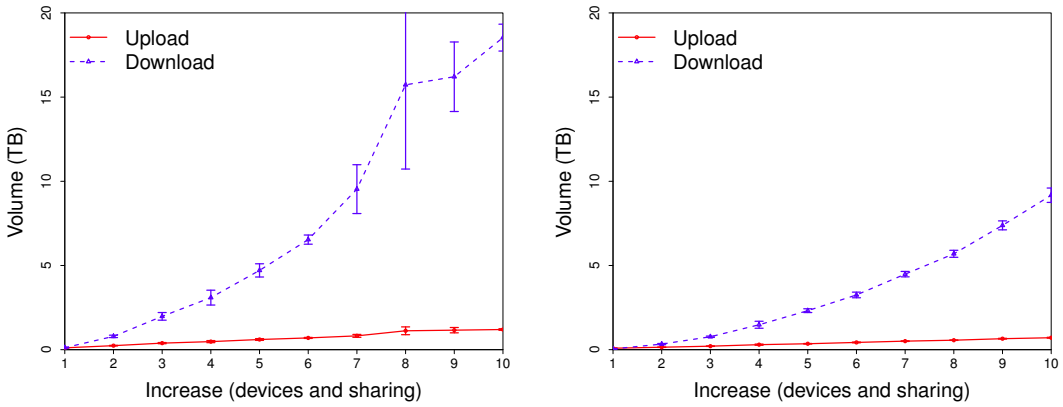
Figure 5.14 presents mean traffic volumes for the larger sharing scenario – i.e., when we increase only the mean number of devices per namespace. We can see that download volume is strongly affected by varying number of devices per namespace. Note the download volume more than doubles when doubling the mean number of devices per namespace for both Campus-1 and PoP-1. For instance, mean download volumes are 0.1, 0.3, 0.6 and 1.4 TB in Campus-1 when the respective mean number of devices per namespace is 1, 2, 4 and 8 times the values observed in actual traces. Similar trend can be noted for PoP-1. The upload volume, on the other hand, is not affected by the number of devices per namespace. Recall this result refers to the total network volume, whereas larger sharing might affect data volume generated per user



(a) Campus-1 (b) PoP-1

Figure 5.14: Mean upload and download volumes in larger sharing scenarios.

in the network. We investigate this issue in Section 6.3 by measuring the correlation between the number of namespaces per user (i.e., a social feature) and the volume of synchronization (upload and download) per user, i.e., user's activity.



(a) Campus-1 (b) PoP-1

Figure 5.15: Mean upload and download volumes in larger population and sharing scenarios.

Finally, Figure 5.15 presents the larger population and sharing scenario, in which we vary both the number of devices and the mean number of devices per namespace simultaneously. For this figure, we increase both quantities at the same rate. Upload volume follows the trend observed in Figure 5.13 as expected. More interesting, note how this combination of more devices with more sharing results in a quadratic-like growth in download traffic. For instance, when both the number of devices and the mean number of devices per namespace are 1, 2, 4 and 8 times the values observed in

actual traces, the mean download volumes are 0.1, 0.7, 3.0 and 15.7 TB in Campus-1 and 0.05, 0.3, 1.5 and 5.6 TB in PoP-1, respectively. That is, in a hypothetical scenario where there are four times more devices than what is observed nowadays, and where those devices share 4 times more namespaces, edge networks would experience up to a 30-fold increase in the download traffic related to cloud storage.

This simple study shows the effectiveness of CloudGen for evaluating future cloud storage usage scenarios.

5.6 Summary

In this chapter, we presented the results we obtained so far with respect to the RG1 of this dissertation, which regards modeling user behavior and workload patterns of cloud storage services. We can summarize our main contributions in this chapter as follows:

- We proposed a hierarchical model of the Dropbox client behavior and parameterized it using passive measurements gathered from 4 different networks. To the best of our knowledge, we are the first to model the client behavior and data sharing in Dropbox.
- We developed and validated a synthetic workload generator (CloudGen) that reproduces user sessions, traffic volume and data sharing patterns. We offer CloudGen as free software to the community[5].
- We illustrated the applicability of CloudGen in a case study, where the impact of larger user populations and more content sharing are explored.

In the next chapter, we leverage users' behavior we have learned through our hierarchical model to analyze the impact of content sharing in cloud storage services.

[5]CloudGen is available at `http://cloudgen.sourceforge.net`.

Chapter 6

The Impact of Content Sharing on Cloud Storage Services

In this chapter, we address our second research goal (RG2), which regards the effects of content sharing and its derived social features on cloud storage services cost and their user activity.

Cloud storage services offer a popular environment for users to store and share content. Yet, content sharing leads to multiple downloads of a single content for synchronizing devices. These downloads contribute to the workload imposed on the servers and may lead to data transference waste, when users sharing the same content are close to each other (e.g., within a campus network), thus generating *avoidable* downloads from the cloud. Intuitively, solutions to offload servers, such as the deployment of caches nearby end users, could be applicable to cloud storage services. However, most services do not implement any distributed synchronization architectures as we discuss in Chapter 3. Moreover, content sharing might increase the service usage, which is an important aspect for service providers interested in attracting users and increasing their profits, e.g., making users to store more data in the service.

Given the aforementioned issues, we intend to answer the following questions in this chapter:

1. To which extent content sharing in cloud storage would lead to avoidable downloads from the cloud (and thus data transference waste) to current networks?

2. Would the introduction of provider's caches (such as CDN nodes) reduce the volume of avoidable downloads in networks in a cost-effective way?

3. To which extent content sharing functionalities contribute to keep users active in the service?

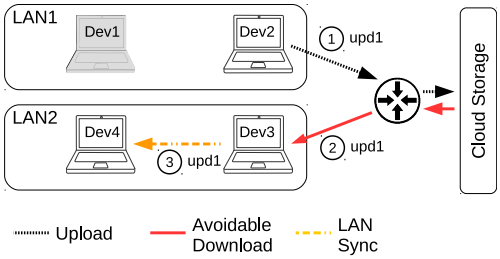
We address the first question in Section 6.1. Although Dropbox employs mechanisms to reduce network traffic, such as `LAN Sync`, we have observed in Chapter 5 that downloads account for higher traffic than uploads in the monitored networks. To investigate this issue, we characterize content sharing among Dropbox users and quantify the fraction of download traffic which are *avoidable*, i.e., related to content replicas within each monitored networks, and therefore, data transference waste.

We address the second question in Section 6.2. Some studies have proposed cache-based architectures to support content sharing in specific environments such as enterprises [Vrable et al., 2012; Bessani et al., 2014] and universities [Liu et al., 2013]. However, these studies do not consider avoidable downloads given by content sharing in a large cloud storage service as Dropbox and whether local caches could reduce such traffic cost. We propose a modification to the synchronization architecture of a typical cloud storage service[Drago et al., 2012; Gracia-Tinedo et al., 2015], introducing caches managed by the service provider to temporarily hold user updates. We evaluate this architecture in various setups, considering both typical scenarios based on our datasets and simulations with larger user populations and/or content sharing.

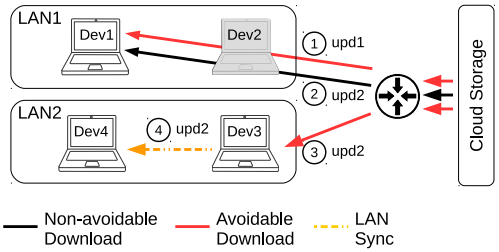
Finally, we address our last question in Section 6.3. Despite the costs associated with content sharing in cloud storage service, some studies have provided evidence that functionalities related to content sharing (e.g., collaborative work, multiple devices synchronization) are important requirements for service adoption by users [Marshall et al., 2012; Palviainen and Rezaei, 2015]. However, these studies do not provide metrics to quantify the relationship of such functionalities with user activity. Towards filling this gap, we evaluate the impact of social features, i.e., features that represent the interaction among users derived from content sharing in Dropbox, on user activity in different networks.

6.1 Avoidable Downloads

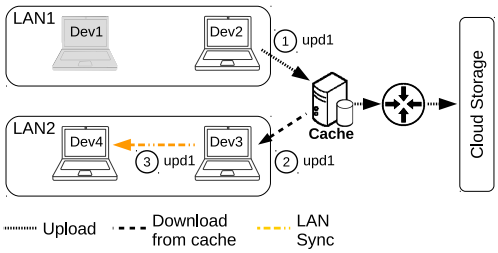
In this section, we address the first question we posted in the beginning of this chapter, namely: "to which extent content sharing in cloud storage would lead to avoidable downloads from the cloud (and thus data transference waste) to current networks?". We do so by analyzing datasets described in Chapter 4 to better understand the effects and dimensions of downloads that are *avoidable*.



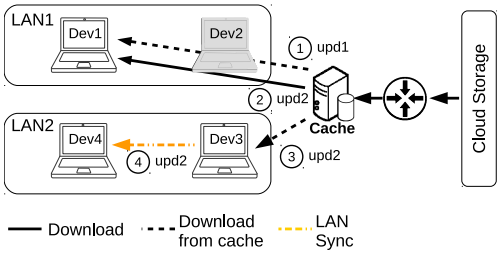
(a) Update generated inside the network (Dev1 is off-line)



(b) Update generated outside the network (Dev2 is off-line, Dev 1 is back on-line and retrieves upd1 and upd2 from the cloud)



(c) Update generated inside the network (Dev1 is off-line)



(d) Update generated outside the network (Dev2 is off-line, Dev 1 is back on-line and retrieves upd1 and upd2 from the cache)

Figure 6.1: Content synchronization in Dropbox current architecture with common cases of avoidable downloads from the cloud to synchronize devices (a–b). Our proposal for an alternative synchronization architecture (c–d).

6.1.1 Identifying Avoidable Downloads

In order to define *avoidable* downloads, we first briefly review Dropbox synchronization protocol, detailed in Chapter 2 Section 2.3. Specifically, we use Figure 6.1 to illustrate its main components. An update represents any modification in a Dropbox namespace that yields a new version of this namespace (i.e., an increment of the namespace JID). The occurrence of any update triggers synchronization of data in devices with the cloud storage. Figures 6.1 (a) and 6.1 (b) summarizes the current synchronization architecture of Dropbox with `LAN Sync`. Devices in different LANs are kept synchronized retrieving updates either from the cloud (solid black or red arrows) or from local peers using `LAN Sync` (orange arrows). In practice, however, typical campus and ISP networks are subdivided into multiple LANs – i.e., Dropbox broadcast messages reach a limited number of devices. Moreover, devices sharing namespaces must be on-line *simultaneously* to allow the `LAN Sync` protocol to work effectively.

Table 6.1: Uploads and downloads in Dropbox. Note the shared and avoidable volumes.

Dataset	Shared Namespaces	Retained Volume[1] (GB)			Shared Namespaces Volume[2] (GB)			Months
		Upload	Download	Total	Upload	Download	Avoidable	
Campus-1	12,529 (36%)	1,304 (77%)	2,282 (60%)	3,586 (65%)	605 (46%)	1,425 (62%)	411 (18%)	3
Campus-2	3,254 (33%)	326 (53%)	443 (44%)	769 (47%)	129 (40%)	253 (57%)	74 (17%)	3
PoP-1	7,514 (26%)	2,604 (67%)	3,909 (57%)	6,513 (61%)	1,482 (57%)	2,601 (67%)	761 (19%)	7
PoP-2	2,925 (24%)	1,850 (77%)	2,558 (68%)	4,408 (72%)	1,077 (58%)	1,801 (70%)	637 (25%)	11
Total	25,952 (30%)	6,084 (71%)	**9,192 (60%)**	**15,276 (64%)**	3,293 (54%)	6,080 (66%)	**1,883 (20%)**	–

[1] Percentages refer to the dataset before discarding updates. Differences are compatible with the prevalence of NATs in the networks.
[2] Percentages refer to the sample in which we can estimate update sizes – i.e., data in columns 3, 4 e 5.

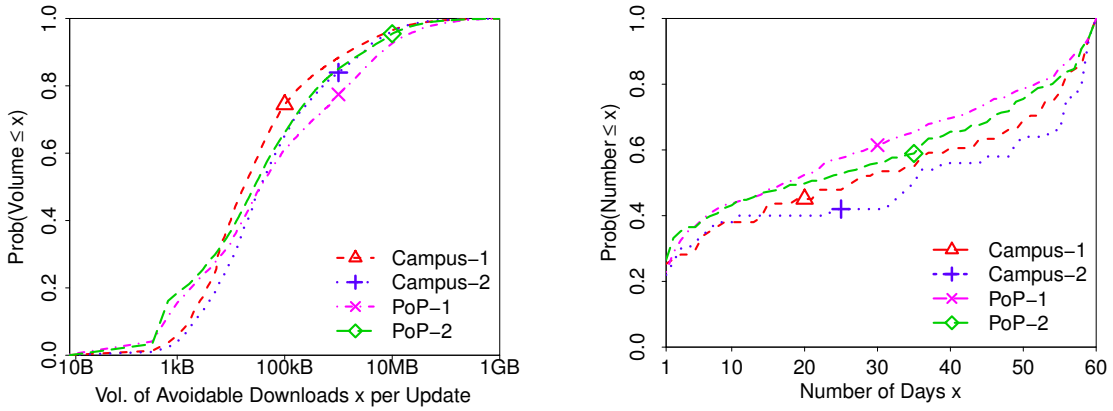
Note in Figure 6.1 that some updates (solid red arrows) need to be retrieved from the cloud to synchronize devices, even if the same updates have already been observed in the network – e.g., when `LAN Sync` is not effective. We call those cases *avoidable downloads*. Avoidable downloads occur either because the update has been previously uploaded by a device in the network, or because multiple devices download a single update generated elsewhere.

Specifically, we track all updates of namespaces and mark downloads as avoidable when: (i) a namespace is updated to a specific JID in a device; and (ii) at least one other device registering the namespace has been seen with the same JID. Note that our methodology does not take into account synchronizations performed using `LAN Sync`, since such traffic does not reach our probes. We thus identify avoidable downloads *after* `LAN Sync` actuates.

We list in Table 6.1 results regarding our identification of avoidable downloads as well as some statistics of the datasets. This table includes: (i) the number of namespaces shared by at least two devices in the monitored networks, i.e., shared namespaces; (ii) summary of traffic volumes retained for the analysis (see Table 4.2 for detailed description); (iii) traffic volumes associated with shared namespaces, and *avoidable downloads*; (iv) time period (in months) covered by each dataset.

6.1.2 Characterizing Avoidable Downloads and Namespaces Lifespan

Table 6.1 shows a high volume of downloads in all datasets. We observe that 58–64% of the total retained traffic corresponds to downloads (this is 44–68% of the datasets total downloads), and 57–70% of such download traffic belongs to shared namespaces. This raises a question on how much traffic is associated with content replication. Not all downloads are avoidable because devices may retrieve content uploaded in other networks, e.g., when a user has remote devices, or namespaces shared with users located somewhere else. We see however that the percentage of avoidable downloads is quite



(a) CDF of avoidable downloads per update (log scale on x-axis)

(b) CDF of namespace lifespans

Figure 6.2: Volume of avoidable downloads per namespace update and namespaces lifespan.

significant. Overall, we find that up to 25% of the Dropbox download traffic happens to synchronize content that has been observed in the same network.

Figure 6.2a presents the cumulative distribution function (CDF) of total download volume that is avoidable per update. Note that most updates that cause such downloads generate little traffic. Yet, a non-negligible portion of that updates results in large volumes. For example, while 25–39% of the updates generates at least 100 kB of replication, 3–7% of the updates surpass 10 MB, reaching up to 5–38 GB of replication[1]. The largest updates volumes correspond to PoPs networks, possibly, due to their dataset long period as well as large volume of multimedia content users synchronize with the cloud and among their devices in homes [Bocchi et al., 2015b]. It is worth noting, the number of avoidable downloads per update is also small: 71–84% of updates shown in that figure have only one avoidable download, with some updates presenting up to 8–14 avoidable downloads. Ultimately, we have observed that the amount of avoidable downloads shown in Table 6.1 consists in the majority of small updates volumes.

Table 6.1 also shows that the percentage of shared namespaces in campuses ($\geq$ 33%) is higher than in PoPs ($\leq$ 26%). Such difference is not surprising and has been associated with the use of Dropbox for collaborative work in campuses, and the synchronization of different devices of a user at home, as we have characterized in Section 5. Moreover, we showed in that chapter users restrict their work on few namespaces in short time intervals (e.g., a week). Here, we further analyze this

[1] Recall Chapter 5, update volume present a heavy tail distribution, thus we have limited the volumes shown in Figure 6.2a up to 1 GB to improve visualization.

behavior, as users' access to namespaces concentrated within a short time (i.e., strong temporal locality) might favor the deployment of providers' caches. To that end, we analyze the occurrence of updates in namespaces over time.

We observe that users' interest on namespaces tends to last for a short time: after an initial period, the number of accesses (i.e., updates or downloads) in a namespace becomes negligible. We illustrate this behavior in Figure 6.2b by showing CDFs of the number of days between the first and the last access in the first 60 days of life of namespaces. We refer to this time period as the namespace *lifespan*. Moreover, we focus on namespaces created during our captures and consider only namespaces seen on-line for at least 60 days[2] .

Notice how all accesses occur in the day the namespace first appears for 22–27% of the namespaces. The median lifespan of a namespace is around 1 month. There is, however, a non-negligible number of namespaces with long lifespans: e.g., around 20% of the namespaces in PoP-1 still present activity 50 days after the first access. Yet, overall, we observe that namespaces tend to have short lifespans – i.e., their accesses occur with a strong temporal locality.

In sum, our analysis confirms that downloads dominate Dropbox traffic in edge networks, such as campuses and PoPs, and a significant part of the downloads is avoidable, contributing to increasing costs of the provider. Table 6.1 shows significant percentages of avoidable downloads in four networks. Moreover, namespaces have limited lifespan. Updates on those namespaces are typically small and present strong temporal locality. These results motivate us to investigate next whether the introduction of providers' caches could reduce the volume of avoidable downloads in a cost-effective way.

6.2 A New Synchronization Architecture

We now turn to the second question we posed in this chapter: "would the introduction of providers' caches reduce the volume of avoidable downloads in networks in a cost-effective way?". To address this question, we propose a new synchronization architecture for cloud storage services with focus on content sharing as Dropbox. This architecture consists of introducing network caches to temporally hold user updates in the network. We aim at enabling device synchronization *without the need to retrieve content from the cloud* in scenarios where the `LAN Sync` protocol is not effective.

[2] Given this filter, we consider all namespaces observed in our datasets (not only shared ones) in order to analyze a representative number of namespaces.

We first discuss the main components of our new architecture and then present an evaluation of its cost-effectiveness.

6.2.1 A Caching Architecture for Cloud Storage

Figure 6.1 (c and d) illustrates our proposal into play in the same cases shown in Figure 6.1 (a and b). A *storage cache* is installed to cover several networks and possibly thousands of customers – e.g., multiple LANs in a large campus, complete ISP networks or multiple PoPs. As a consequence, devices can potentially find updates without retrieving content from the cloud.

The synchronization of devices using the storage caches would work as follows. The discovery of caches is orchestrated by the cloud storage provider's protocols when devices log in to the system. Provider's servers inform clients about the closest cache in the network. Devices always send updates in namespaces to the closest cache (see Figure 6.1c step 1). The cache stores updates locally, also forwarding them to the cloud. As soon as devices receive notifications about updates, they contact the closest cache. If the pending updates exist in the cache (i.e., a cache hit), the cache delivers the content directly to devices. Otherwise (i.e., a cache miss), the cache retrieves the content from the cloud and forwards it to the requesting devices (see Figure 6.1d steps 2 and 3). After any request from a client, the cache executes internal replacement policies to guarantee that the most likely useful content is available locally. Any caching replacement policy could be employed for that matter (see [Rabinovich and Spatschek, 2002] for references), and an evaluation of the best caching policy is outside the present scope.

We envision the caches being deployed and controlled by cloud storage providers directly, such as CDN nodes. Moreover, we do not assume any particular deployment topology. The location of caches in the network would be a choice of operators, and the only requirement is that clients must have routes to reach the caches. The extensions needed in the provider's protocols to diverge traffic to caches, as well as other aspects related to the implementation of the architecture, are also out of our scope. We here aim at simply assessing the extent to which the new architecture can bring benefits to the service provider by reducing the bills at the outsourced data store component, i.e., removing workload from this component. In the following, we discuss the methodology we adopted to evaluate whether the caching approach is cost-effective for storage providers considering the cache costs and resulting data transference savings. Equally important questions, such as the privacy risks and the management overhead of deploying caches in several locations are left for future work.

6.2.2 Evaluation Methodology

We evaluate the proposed architecture using both our datasets and synthetic traces. We first use real traces to study whether storage caches are cost-effective in typical campus and ISP networks. Then, we rely on synthetic traces to extrapolate measurements and understand how costs and benefits vary when large populations are covered, or more sharing is seen in the network.

For both the real and synthetic traces, we simulate a network as in Figure 6.1, where a provider's cache is deployed to cover the devices. It intercepts updates, potentially serving content without involving the cloud. For simplicity, we assume all content is uploaded from the simulated network as shown in Figures 6.1 a and c. In this case, external devices produce no workload to be downloaded for the synthetic traces, whereas only the content generated inside the network are cached for the real traces.

We rely on *CloudGen*, our synthetic workload generator for cloud storage services introduced in Chapter 5, for creating synthetic traces. We follow the methodology presented in that chapter (see Section 5.5), where a number of devices acts independently, performing updates that trigger several downloads. Next, we create scenarios where a larger devices population adopt the service and/or device populations share more content through shared namespaces. Thus, we generate four types of synthetic traces:

1. *Typical Workload*: We parameterize CloudGen with the number of devices observed in our datasets to illustrate how the synthetic workload compares to real traces;

2. *High Population*: We simulate larger device populations by tripling the number of devices – i.e., roughly equivalent to have a Dropbox client in every IP address observed in Campus-1. Both upload and download volumes grow linearly with the number of devices;

3. *High Sharing*: We triple the mean number of devices sharing each namespace. The scenario mimics environments where users share lots of namespaces, e.g., hypothetical companies adopting cloud storage as network file systems. Larger numbers of devices per namespace increase only download traffic;

4. *High Population and Sharing*: We combine the previous two scenarios. Uploads grow linearly with the number of devices, while downloads grow as much as 20 times when compared to the typical scenario.

We calculate savings considering different cache sizes for both real and synthetic traces. We use the first month in each dataset of real traces to warm-up the cache, and assess savings in subsequent months. For synthetic scenarios, we generate two-month long intervals (warm-up and evaluation) in 16 experiment rounds, and report mean values with 95% confidence intervals.

In all experiments we use the simple and yet widely-used *Least Recently Used* (LRU) cache replacement policy: cache insertions and evictions are triggered by uploads; downloads manipulate the order of items (i.e., updates) in the LRU cache, but do not change cache contents. Finally, by tracking updates and the behavior of the cache, we identify whether an update should be retrieved from the cache or from cloud.

6.2.3 Performance of the Cache-Based Architecture

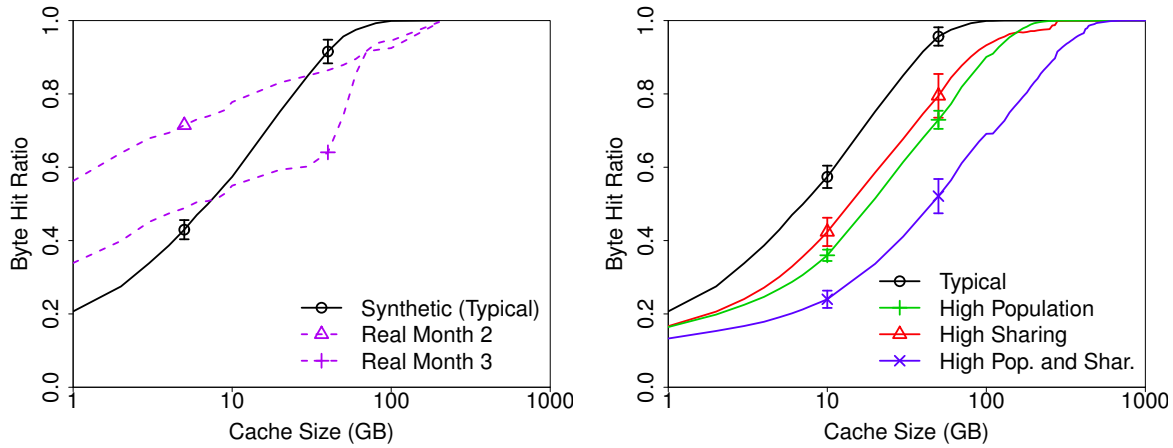
Now that we have detailed the architecture, we analyze its performance.

6.2.3.1 Data Transference Savings

We now discuss the data transference savings obtained with our cache-based architecture focusing on Campus-1 real trace, as well as on several synthetic traces generated using CloudGen (parameterized according to that dataset). Recall the cache in this architecture is managed by the cloud storage provider, which benefits of the data transference savings, even though the network, where the cache is hosted, also benefit of the application traffic reduction. We show results only for Campus-1 trace. Yet, general conclusions in the following hold for all traces.

We evaluate data transference savings using the *byte hit ratio* metric, which is computed as the volume of *downloads* served by the cache over the total volume of downloads reaching the clients. Figure 6.3a reports the byte hit ratios of two months extracted from the real trace (dashed purple curves) as well as a synthetic trace in the *typical scenario* (continuous black curve).

We note that differences between the two months in real traces are owing to normal variations in the workload: "month 3" presents higher volume in shared namespaces than "month 2", which reduces the byte hit ratio, as we will discuss next. We also note that, despite some divergences, the synthetic traces capture reasonably well the overall trend in real data. Such divergences are due to multiple factors. First, the synthetic trace is built by configuring CloudGen with input parameters extracted from the complete Campus-1 trace, as opposed to a sub-trace corresponding to a particular month. Thus, it is expected that it captures an overall trend observed during



(a) Real and Synthetic

(b) Synthetic Scenarios

Figure 6.3: Performance of the cache-based architecture (Byte Hit Ratio) in various scenarios.

the whole three-month period, and not the particular behavior in shorter periods. Moreover, some simplifying assumptions made in the design of CloudGen contribute to the divergences, particularly for caches lower than 10 GB as shown in Figure 6.3. For instance, CloudGen assumes that devices behave independently, which might not be always the case. Multiple devices sharing namespaces may be on-line simultaneously, favoring the caching approach.

Yet, despite such divergences, we still see overall common trends. For example, for both real and synthetic workloads, even the smallest cache we evaluate is able to remove a large fraction of avoidable downloads. In fact, a cache of 1 GB would achieve from 21% to 56% of byte hit ratio. Moreover, in all three curves, the byte hit ratio surpasses 90% with a 70 GB cache.

Having compared real and synthetic traces in the *typical scenario*, we focus on the latter to evaluate the proposed architecture in other scenarios. Results are in Figure 6.3b. We note that, for any given cache size, the byte hit ratio is higher in the typical scenario than in any of the other three scenarios. This is because workloads are heavier in other scenarios, resulting in more cache misses. When the population size is increased, the larger upload volumes force the cache to remove old updates more often. In the high sharing scenario, the larger number of devices downloading content (i.e., sharing namespaces) increases the probability that a device will fail to find a content in the cache, because the content has already faced eviction. Cache savings decrease even more in the high population and sharing scenario, as both factors are present.

Nevertheless, despite such variations, our proposed architecture provides significant data transference savings in all scenarios. For example, for a 10 GB cache, the byte hit ratio varies from 24% to 57%.

Overall, we conclude that even modest caches, as we consider the evaluated ones, can remove a relevant volume of avoidable downloads, in particular, when deployed in the typical network. As larger networks and/or high sharing scenarios are considered, the cache size to achieve similar savings needs to be adjusted.

6.2.3.2 Cost-Benefit Tradeoff

The tradeoffs of deploying our architecture involve the cache costs and savings achieved by removing downloads from the cloud. In other words, an effective cache should prevent avoidable downloads at a minimum cost for the cloud storage provider.

We evaluate costs and benefits of the architecture by computing the *relative cost savings* (rcs) for a time interval t:

$$rcs_t = \frac{cost\_nocache_t - cost\_cache_{t,c}}{cost\_nocache_t}, \tag{6.1}$$

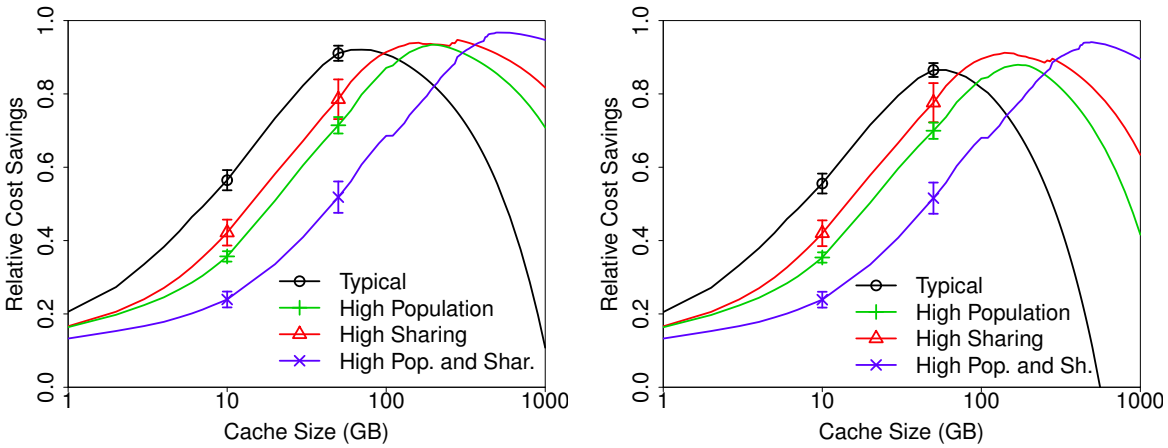
where $cost\_nocache_t$ is the cost to serve all avoidable downloads in the time period t, while $cost\_cache_{t,c}$ corresponds to the cost associated with deploying a cache of size c bytes during time period t.

In short, rcs_t *captures the fraction of the costs to serve avoidable downloads that the architecture can recover.* The architecture is effective when $rcs_t > 0$, i.e., the savings obtained by removing avoidable downloads at least pay off the costs associated with maintaining the cache. Note that rcs_t has an upper bound ($rcs_t = 1$) that indicates a scenario where all avoidable downloads are removed at negligible costs.

For the sake of simplicity, we express the *cost_nocache* and the *cost_cache* as functions of the number of bytes transmitted over the network and stored in the cache:

$$\begin{aligned} cost\_nocache_t &= d_t * \beta \\ cost\_cache_{t,c} &= m_{t,c} * \beta + c * \alpha, \end{aligned} \tag{6.2}$$

where d_t is the total number of bytes associated with avoidable downloads observed in the period t, $m_{t,c}$ is the number of bytes associated with avoidable downloads that a cache of size c bytes misses during time t, and α and β represent storage and data transference prices per byte, respectively. We here assume that the cost of downloading from the local cache is negligible. Our premise is that the traffic generated by users applications within the network does not imply costs for network administrators or



(a) Storage prices are half of data transference prices ($\alpha/\beta = 1/2$)

(b) Storage prices are equal to data transference prices ($\alpha/\beta = 1$)

Figure 6.4: Performance of the cache-based architecture (Relative Cost Savings) in various scenarios based on synthetic traces.

applications providers (e.g., Dropbox). In fact, network administrators would benefit of data transference reduction related to Dropbox download traffic from the cloud to the network.

We take as reference for α and β the prices for storage and data transference offered by Amazon S3/E2C[3], which already include operational costs. Based on that, we evaluate rcs_t on two distinct cost setups, defined by the ratio α/β: (i) the price per byte of storage is a half of data transference (i.e., $\alpha/\beta = 1/2$) – e.g., because magnetic storage is used; (ii) both prices are the same (i.e., $\alpha/\beta = 1$) – e.g., SSD storage is taken. We use Amazon S3/E2C as reference because many storage providers rely on it to build their services. Other references (e.g., market prices for SSD disks and mobile data plans) provide more optimistic cost-benefit tradeoffs.

Figures 6.4a and 6.4b show the rcs_t for the four scenarios and the two price ratios considering t equal to 1 month. Note that rcs_t increases to a maximum and then decreases. The inflection occurs when the best tradeoff between costs and benefits is achieved. Comparing to Figure 6.3b, we see that the decrease in rcs_t happens despite higher byte hit ratios. Therefore, caches larger than a certain threshold add costs to the system, without providing significant savings.

Figure 6.4a ($\alpha/\beta = 1/2$) shows that rcs_t reaches 92–93% for typical and high population scenarios with cache sizes of 70 GB and 200 GB, respectively. Scenarios

[3] https://aws.amazon.com/ec2/pricing

with higher sharing achieve slightly better rcs_t. The maximum rcs_t reaches 95% for a 280 GB cache in the high sharing scenario, going up to 97% in the high population and sharing scenario with a 500 GB cache. Better tradeoffs with higher sharing are explained by the large volume of downloads: even if the byte hit ratio is lower than in typical scenarios for certain cache sizes (see Figure 6.3b at 100 GB caches), the volume saved with avoidable downloads pays back the investment.

It is worth noting the effects of the ratio between storage and transference prices: the higher the storage price (compared to transference), the lower the advantage of deploying the architecture. Comparing Figures 6.4a and 6.4b, we see that the maximum rcs_t slightly decreases for $\alpha/\beta = 1$, e.g., it is 7% lower than for $\alpha/\beta = 1/2$ in the typical scenario. Moreover, the costs of overestimated caches become equivalent to the data transference savings faster when storage costs are high.

In sum, our proposal cost-effectively achieves high data transference savings in typical scenario, where 92% of the costs relative to avoidable downloads can be recovered. Storage providers thus have an incentive to deploy the caches, since savings by far compensate the costs. Benefits are higher with high-sharing (rcs_t reaches up to 95–97%), which hints to networks scenarios where cloud storage providers should start deploying the architecture.

6.3 The Importance of Content Sharing

In this section, we tackle the third question we posed and analyze to which extent content sharing contributes to keep users active in the service. To that end, we first define metrics to quantify user activity and interaction. We then investigate the relationship between these metrics.

6.3.1 Quantifying User Activity and Interaction

First, we define a metric for capturing the activity level of a user. Various technical features we have included in our hierarchical user behavior model, such as user sessions, content updates within sessions, volume of data transmitted (upload/download), may be used to represent user activity. Yet, we choose a different metric that captures the workload the user generates and imposes on the service. More specifically, we estimate the level of activity of a user u as the total volume of data synchronized by the user with Dropbox storage services during a time period t. This is given by the sum of

upload and download volumes associated with all devices of the same user u during window t.

Next, we define social features which represent the interaction among users when sharing content. We evaluate three social features:

1. number of namespaces per user: it refers to the number of distinct namespaces a user synchronizes any content during t. Recall Chapter 2, namespaces are the user initial folder, which is unique and not shared, and the user shared folders. The higher is the number of namespaces of user during t, the higher is her interaction with other users (i.e., collaborative work).

2. number of contacts per user: it refers to all distinct users associated with the shared folders a user synchronizes any content during t. This feature represents explicitly the user interaction with other users by sharing content.

3. number of devices per user: it refers to all distinct devices associated with the shared folders a user synchronizes any content during t, i.e., devices of the user and devices of her contacts. This feature also represents the file synchronization functionality which allows a user to synchronize content (private or shared) across different devices. Even though this is not a *social* feature per se, we here consider it as such because it may indirectly affect how a user collaborates with others.

We adopt three steps to quantify the relationship between social features and user activity. First, we estimate the activity level of a target user u and her social features for each time window t (we consider monthly time windows). Thus, we generate a number of samples consisting of user and period pairs (u, t) and associated social feature and activity level values of u during t. Note that the same user may appear in multiple samples in our analyses. Each occurrence of a user u is handled independently. Second, we group (u, t) samples based on the values of the social features of user u during period t. Specifically, we define different social feature *levels*, and group the (u, t) samples based on them. Each level represents the integer value of the given social feature. For example, we group together all samples for which the number of contacts is equal to 2 and associate with it the level of number of contacts 2. For each of the three social features, we thus define levels 1, 2, 3 ... n, grouping all samples for which the feature value is greater than or equal to n together at the same (maximum) level. This is performed as it is likely that there are only a few samples with such values. In the last step, we compute the correlation between social feature levels and their respective user activity level.

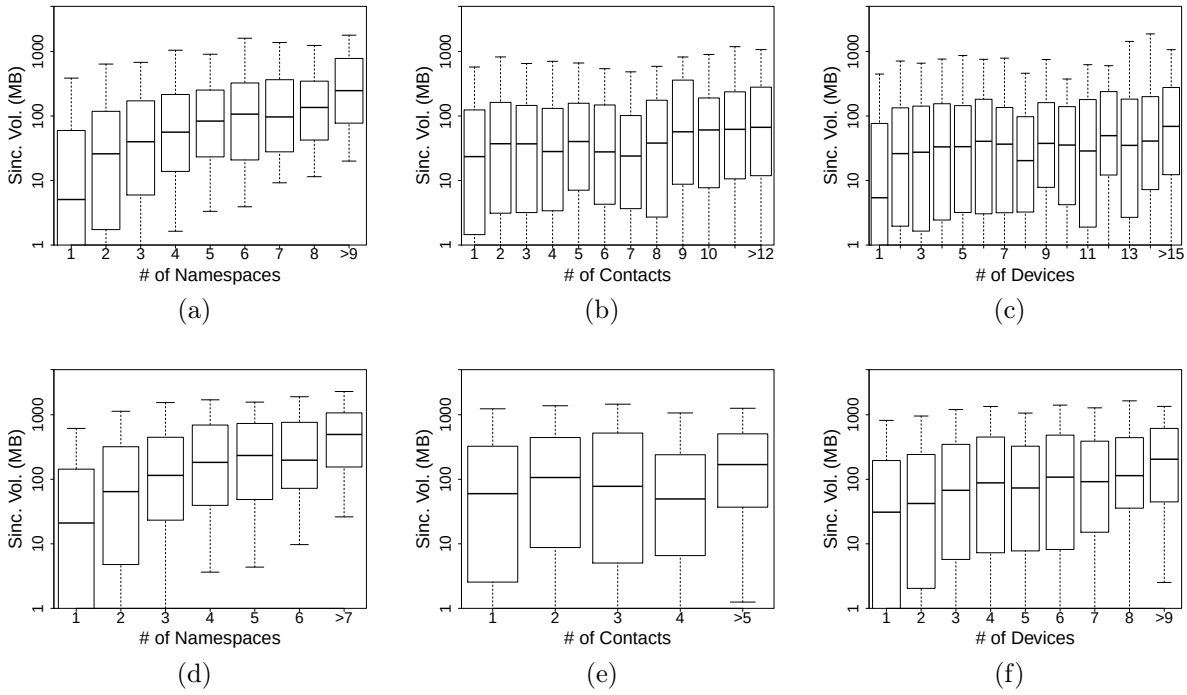


Figure 6.5: Relationship between social features levels and the activity of users per month in Campus-1 (1st. row) and PoP-1 (2nd. row). Number of namespaces presents best relationship with user activity.

6.3.2 Correlation Results

We compute the correlation between each social feature and user activity following the steps described in the previous section and using our datasets described in Chapter 4. Here, we focus only on real traces given our interest in assessing the importance of social features in current scenarios. Moreover, we focus only on users from whom some activity was observed in at least five distinct days during a month in each network, in order to decrease the influence of users who arrived recently or are only visiting the monitored networks.

Figure 6.5 presents boxplots summarizing the distributions of user activity, for $(user, period)$ samples grouped in each level of each social feature. Top plots are relative to Campus-1, while bottom plots depict results for PoP-1. Other networks are omitted since they show similar patterns. Each boxplot shows the following distribution percentiles: the central rectangle spans the first to the third quartiles, the segment inside is the median (second quartile), whereas whiskers above and below the rectangle represent the 9^{th} and 91^{th} percentiles. We use different number of social feature levels,

Table 6.2: Pearson correlation coefficient (Linear dependence) and Spearman correlation coefficient (Non-linear dependence) between social features levels and median user activity level.

	Linear Dependence (Pearson)				Non-linear Dependence (Spearman)			
	Campus-1	Campus-2	PoP-1	PoP-2	Campus-1	Campus-2	PoP-1	PoP-2
Namespaces	0.91	0.74	0.89	0.80	0.98	0.93	0.96	0.73
Contacts	0.80	0.60	0.53	-0.44	0.77	0.62	0.30	-0.50
Devices	0.70	0.67	0.88	0.09	0.71	0.75	0.96	0.29

as shown in x axis of figures, in order to guarantee the level n is represented by at least 1% of distinct users u observed in each network.

In general, this figure shows social feature levels in the campus network is higher than in the PoP as one can observe by the difference between x axes of figures in 1st and 2nd rows. This result is in agreement with our previous observations that content sharing is more prominent in the academic scenario, which concentrate more users focused on collaborative work. Users in residences (PoP networks) also have a relevant amount of collaborative work, as shown in Figure 6.5 (d), even though contacts and devices associated with their shared namespaces might be in different networks, which possibly prevent us from capturing higher social feature levels in Figures 6.5 (e) and (f).

Concerning the relationship between social features and user activity, one can observe in Figure 6.5 that the distribution of activity level exhibits high variability for each social feature level (note the larger span between the 9^{th} and 91^{th} percentiles) and also a clear trend towards smaller values (note the concentration of volumes between the 9^{th} percentile and 3^{rd} quartile, i.e., 66% of data). Therefore, we consider the median as the most adequate value to represent the user activity level per social feature level. Despite the trend towards smaller values of level of user activity, Figures 6.5 (a) and (d) seem to indicate that the higher the number of namespaces the user synchronize content per month is, the higher is the median user activity level. Such trend is not clearly visible for other social features, i.e., number of contacts or devices. We also observe in Figures (a) and (d) that users who synchronize more than five namespaces per month tend to have more homogeneous behavior, i.e., activity level distribution more concentrated around the median. Moreover, these users tend to synchronize high data volumes, e.g., median above 100MB and close to 1GB in PoP networks. Overall, this result suggests that there might be a non-negligible correlation between user activity level and her amount of namespaces (i.e., collaborative work). Again, these observations are not clear for the number of contacts and the number of devices.

Table 6.2 shows the results for each of the computed correlations using the Pearson correlation coefficient, which measures a linear dependence between two variables, and

the Spearman correlation coefficient, which measures a non-linear dependence between two variables [Jain, 1991]. Both coefficients vary between -1 and 1 inclusive, where -1 is the maximum negative (or indirect) correlation, 0 is no correlation, and 1 is the maximum positive (direct) correlation. Note that, in accordance with our above discussion, the number of namespaces is the feature with the strongest (positive) correlation with the median user activity in most networks. Moreover, it is worth noting that such correlation is better expressed by a non-linear correlation, as the Spearman coefficient (0.73-0.98 in the four networks) is higher than Pearson coefficient for the majority of networks.

We also computed the *p-value* to obtain the significance of the correlations shown in Table 6.2 by using the t-test and assuming no correlation as the null hypothesis. We found that all correlations above 0.6 are significant, i.e., the p-value is less than 0.05% then rejecting the null-hypothesis. For the sake of clarity, we omitted those p-values in the table. Note that the feature number of namespaces reached significant correlation in all networks, i.e., Pearson coefficient 0.74-0.91 and Spearman coefficient 0.73-0.98, and it is then the best representative for the user social features in our datasets. Non-significant positive and negative correlations for the features contacts and devices might indicate this sharing activity of users are not captured by our datasets in PoPs environments, as we discussed previously.

In sum, the correlation results we discuss in this section indicate the importance of content sharing functionality for cloud storage services, as an increase in social feature level (in particular, number of namespaces per user) increases the median user activity (i.e., median volume of synchronization) per month. Thus, content sharing contributes to keep users active in the service, and it also may yield increased profits to the service provider as users are likely to store more content in the cloud while interacting or sharing content with other users.

6.4 Summary

In this chapter, we presented the results we obtained so far with respect to the RG2 of this dissertation, which regards the effect of content sharing on cloud storage services. We proposed to answer three questions in RG2: (1) to which extent content sharing in cloud storage leads to avoidable downloads from the cloud (and thus data transference waste) in current networks? (2) would the introduction of network caches reduce the number of avoidable downloads in a cost-effective way? (3) to which

extent content sharing functionalities contribute to keep users active in the service? We performed a thorough investigation of these issues relying on network datasets described in Chapter 4 and simulations with CloudGen parameterized according to those datasets. In sum, our main contributions in this chapter were the following:

- We assessed the impact of content sharing on cloud storage traffic by measuring avoidable downloads in different networks.
- We proposed an alternative and cost-effective architecture for data synchronization in cloud storage services with focus on content sharing as Dropbox.
- We analyzed and quantified the relationship between content sharing and the user activity level in the service by means of social features.

In the next chapter, we tackle our last research goal (RG3), presenting our modeling and analysis effort about costs-benefits tradeoffs in cloud storage services.

Chapter 7

Cost-Benefit Tradeoffs of Content Sharing in Cloud Storage

In this chapter, we finally address our third research goal (RG3), which regards modeling the tradeoffs between the provider and end users interests in cloud storage services.

We have shown in the previous chapter that content sharing has become a valuable feature for users of cloud storage services. By promoting user interactions driven by social ties or facilitating content organization across multiple devices, content sharing may contribute to increase user satisfaction with the service. However, this functionality poses extra costs for the service provider. The synchronization of files to multiple devices associated with the sharing generates extra downloads from the cloud servers to such devices, which ultimately incur in more resources (notably data transference) required from the provider infrastructure.

This issue raises questions about the costs and benefits of content sharing in cloud storage for both end users and the service provider. Most providers adopt pricing models that allow free content sharing and storage usage with limited space, charging for extra space. The idea behind such model is that the service can attract more (paying and free) users while remaining profitable, as the payments for extra space compensate the overall costs. However, such model clearly pressures the provider to reduce costs in order to maintain profitability, since the fraction of users who pay for the service is often small. On the other hand, the policies adopted to reduce costs must not hurt user satisfaction, at the penalty of reducing service attractiveness and losing the paying users.

Indeed, although some providers have survived even with very small fraction of paying users (e.g., 4% in Dropbox [Rogowsky, 2013]), others have left the market (e.g., UbuntuOne, Wualla), possibly due to high operational costs [Lam, 2015; Silber, 2015]. Content sharing may exacerbate the issue. Ultimately, this dynamic[1] generates complex tradeoffs between costs and benefits for both users and providers. Identifying strategies that solve such tradeoffs by balancing the satisfaction of both parties is of utmost importance for providers to remain competitive in the market.

We have addressed costs of content sharing in the previous chapter, where we have shown that caches in networks, like CDN nodes, are a very efficient approach to reduce transference costs of a cloud storage provider. However, another alternative architecture, which count on users collaboration to reduce costs, i.e., Peer-to-Peer (P2P), also could be investigated in this context. Additionally, the cloud storage provider, which adopts P2P architecture, can offer extra benefits to those users, who collaborate to reduce the service cost.

In this chapter, we investigate cost-benefit tradeoffs of content sharing in cloud storage services for the provider and users jointly. We start from the following broad question: *How to model costs and benefits of cloud storage services so to help providers in assessing the effectiveness of alternative policies that aim at increasing both profitability and user satisfaction?* Various characteristics of cloud storage should be taken into account when tackling this question. Some examples are the majority of free users in the service, the attractiveness of content sharing for users (and the corresponding costs for provider), and the urge to keep users satisfied in such a competitive market. The inter-dependencies among these characteristics make the investigation quite challenging.

Given the aforementioned question, we make two contributions in this chapter:

• We propose a general model for the costs and benefits of cloud storage considering both users and providers. To that end, we propose *utility functions* that represent the benefits minus the costs of the service for each party. The greater the utilities are, the more satisfied providers and users become. To keep our scope limited, we focus mostly on objective aspects of cloud storage (e.g., storage and data transference costs), ignoring more subjective aspects (e.g., social impact of collaborative work). As such, our proposal is appealing for capturing in a simple (but representative) model key components of cloud storage.

[1] The dynamic comes from two conflicting goals: improve (free/paying) users satisfaction (e.g., offering some free service functionalities), but reducing operating costs to increase the service profitability.

• We investigate two alternative policies for the current cloud storage sharing architecture, which count on user collaboration to reduce provider's costs via P2P architecture. We evaluate the effectiveness of both policies by applying our proposed model to assess user and provider utilities in various scenarios.

7.1 General Cost-Benefit Model

We now introduce our model for costs and benefits of cloud storage, covering both the service provider and users. We aim at developing a simple but reasonably representative model that captures, to a good extent, the main components of a cloud storage service that drive satisfaction of both parties. Thus, we focus on aspects related to resource consumption, notably storage and data transference. Other aspects, such as the speed of upload or download and the social value of collaborative work, are left out for limiting the present scope.

Our model is based on utility functions that capture, in an abstract level, the provider profit and the user surplus (i.e., satisfaction) with the service. As in other studies [Xu and Li, 2013; Lin and Tzeng, 2014; Shen and Li, 2015], these functions express the difference between the benefits and the costs of the service for each party. Table 7.1 lists the main notation used in the models presented here and in Section 7.2. We note that costs and benefits (and corresponding utilities) may vary over time. We then assume all model variables are expressed for the time window w (e.g., a week or a month). Resources consumed before/after w as well any feedback effect due to decisions made during w (e.g., user response to benefits received) are left out of the present analysis. For simplicity, we omit the time window from the notation, and present the model for any arbitrary time window.

We consider a monopolistic service provider, which charges a fixed monthly or annual price for a certain storage capacity per user (paying users). This provider also offers free storage with a small capacity to any user who registers in the service (free user). Under this pricing model, the utility of the service provider, U_s, can be defined as:

$$U_s = \mathcal{R} - (\alpha * \mathcal{S} + \beta * \mathcal{T}), \tag{7.1}$$

where the provider's benefit is represented by the revenue ($\mathcal{R}$) that comes from paying users and secondary sources. For example, it has been estimated that around 4% of the Dropbox users pay for the service [Rogowsky, 2013], and the company has been

Table 7.1: Summary of the model notation.

Variable	Description
U_s	Service provider utility
U_i	User i utility
$\mathcal{R}$	Provider's revenue including users' payments and other sources
V_i	Valuation of user i per byte – e.g., bytes stored in the cloud, or bytes offloaded from the provider by user i
α	Price of one byte of storage for the provider
β	Price of one byte transferred from/to the cloud for the provider
P_i	Price the user i pays for service
X_i	Storage capacity (in bytes) available to user i in the cloud
u	Users benefit function
$\mathcal{S}$	Total bytes stored in the cloud by all users
$\mathcal{T}$	Total bytes transferred from/to the cloud by all users
$\mathcal{O}$ ($\mathcal{O}_i$)	Total volume (in bytes) offloaded from provider (by user i)
κ	Units of bonus offered by the provider to users per offloaded byte
$\mathcal{C}_i$	Penalty factor imposed on user i per offloaded byte
$\mathcal{B}_i$	Maximum upload capacity (in bytes) user i is willing to offer for offloading the provider
$\mathcal{D}_i$	Maximum storage capacity (in bytes) user i is willing to offer for offloading the provider

supported by funding groups [Crunchbase Inc., 2017].[2] The cost of the provider is estimated in terms of total amount of data stored in the infrastructure ($\mathcal{S}$) and the total amount of data transferred to/from the cloud ($\mathcal{T}$). Both measures are given in bytes, and we define the parameters α and β to represent, respectively, the price per byte of storage per w and the price per byte transferred in the network (i.e., upload/download).

The utility of user i, U_i, is instead given by:

$$U_i = V_i * u(X_i) - P_i, \tag{7.2}$$

where the user's benefit is represented by the function (u) of the user cloud storage capacity (X_i), and a utility level (V_i) that corresponds to the user's valuation for each byte she can store in the cloud. The user's cost is given by the price she pays for the

[2] We assume such funding as a secondary revenue source, which allows the provider to cover free users' operating costs, thus increasing its user base.

service (P_i).[3] We assume u as a simple linear function of X_i in this work, even though more complex functions to represent users benefit (e.g., an alpha fair function [Xu and Li, 2013]) could also be adopted. In turn, the user valuation can vary according to P_i. For example, paying users ($P_i > 0$) may have a valuation greater than free users.

It is worth noting the conflicting interests represented by these equations. From the provider's point of view, in order to increase utility, it has to (i) grow the revenues by attracting more users, which can increase the service popularity (possibly leading to new investments) or even the fraction of paying users, or (ii) reduce the aforementioned costs. The latter is particularly important as more users lead to more resources required from the infrastructure (larger $\mathcal{S}$ and $\mathcal{T}$). From the user's point of view, instead, the utility can be increased by either uploading more data to the cloud or by looking for lower service prices. Free users ($P_i = 0$), who usually are the majority in cloud storage services, cannot increase utility beyond the small free storage capacity offered by provider. The challenging task is to reach the best tradeoff between users' and the provider's utilities.

One example of such tradeoff arises when content sharing is taken into account. Offering this functionality is a way of making the service more attractive to users. However, it leads to additional transfer costs (i.e., downloads from the cloud). Defining strategies to reduce such costs is then of utmost importance, as further discussed in Section 7.2.

7.2 New Content Sharing Policies

We now build upon the cost-benefit model presented in the previous section and propose content sharing policies that can be advantageous for both the provider and users. The goal of these policies is to enable device synchronization *without the need to retrieve content from the cloud.*[4] We start by presenting our general approach to design such policies and how our general model can be used to estimate the improvements achieved by any such policy (Section 7.2.1). We then introduce the design principles of two specific policies (Sections 7.2.2 and 7.2.3). Finally we build upon our general model to derive the impact that each policy causes on provider's and users' utilities (Section 7.2.4).

[3]Note that we ignore the costs of sending the personal content to the cloud, which is the onus applicable to every user registering to the service.

[4]In the case of Dropbox, the policies should enable synchronization in scenarios where `LAN Sync` is not effective, e.g., users in different LANs.

7.2.1 Estimating Utility Improvements

The provider can apply different strategies in order to jointly increase its utility and users' utility. We evaluate alternative service policies that trigger users to contribute with part of their idle resources (e.g., upstream data transference and/or storage), receiving as a compensation a bonus proportional to the costs offloaded from the provider. In this way, the provider can improve its utility by reducing its operational costs, whereas users' utility improvements come from earned bonus.

Specifically, we consider the user bonus as a free extra storage space in the cloud, which the provider attaches to the user account as she offloads the service traffic. Offering bonuses to users has already been explored by some cloud storage providers. For example, Dropbox offers free storage in campaigns for incentivizing users to invite friends to the service.[5]

Given a policy that offers bonuses to users, the new provider's utility function is defined as:

$$U_s^{new} = U_s + \mathcal{P}_s, \tag{7.3}$$

where utility gain $\mathcal{P}_s$ added to the previous provider utility U_s comes from the reduction of the provider's cost achieved with the policy. $\mathcal{P}_s$ includes costs related to resources that are offloaded by users as well as costs related to the new bonuses offered to the participating users.

The new user utility, in turn, is given by:

$$U_i^{new} = U_i + \mathcal{P}_i, \tag{7.4}$$

where $\mathcal{P}_i$ represents the net benefit earned by user i from her participation in the policy, which includes the bonuses received as well as the new costs related to local resources (storage and network) used for offloading the provider.

The exact definitions of $\mathcal{P}_s$ and $\mathcal{P}_i$ depends on the specific policy adopted. Given these definitions, we are able to estimate the impact of a particular policy in terms of changes in the utilities of the provider and of user i, I_s and I_i, respectively. Such impacts are computed as the relative difference between the new and previous utilities:

$$I_s = \frac{U_s^{new} - U_s}{U_s} = \frac{\mathcal{P}_s}{U_s}; \qquad I_i = \frac{U_i^{new} - U_i}{U_i} = \frac{\mathcal{P}_i}{U_i}. \tag{7.5}$$

[5] A user who invites someone to the service receives 500 MB of extra space in the cloud when the invited person installs the Dropbox client.

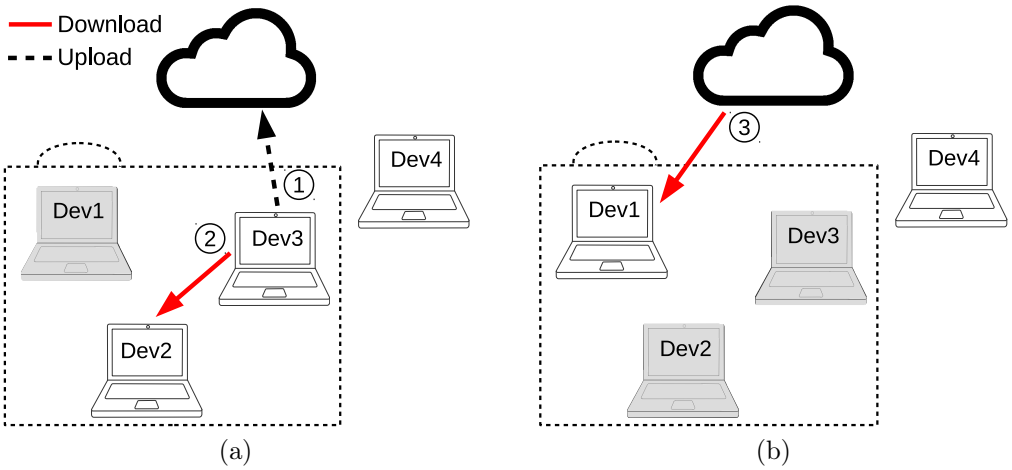


Figure 7.1: Common cases of downloads in cloud storage with Contact Sharing Policy: Dev 1, 2 and 3 are associated with the same shared folder, (a) Dev 3 sends new updates to the cloud and serves Dev 2 (Dev 1 is offline), (b) Dev 1 is back online and retrieves new updates from the cloud (Dev 2 and 3 are offline).

Note that I_s and I_i can become negative if the policies lead to negative impact on the utilities.

Having discussed how we estimate the impact of new content sharing policies on both utilities, we now turn to two specific policies, referred to as Contact Sharing Policy (CSP) and Anonymous Sharing Policy (ASP), presenting their main design principles.

7.2.2 Contact Sharing Policy (CSP)

CSP allows users associated with a shared folder, here referred to as *contacts*, to serve each other the content generated in the folder, thus offloading the provider of such data transfers. Users then receive bonus as a free extra storage space in the cloud, which the provider attaches to the user account as she offloads the service traffic.

To accomplish this policy, the provider first establishes a set of eligible user devices for each given folder. Only eligible devices are allowed to serve the folder's content to others. In general, all devices that share the given folder are eligible to serve it. The provider then invites eligible users to participate in the policy, expecting that a percentage of them will accept the invitation and join the policy. Users that accept the offer inform the provider the maximum upload capacity in bytes ($\mathcal{B}_i$) they are willing to contribute in the time window.

Figures 7.1 (a and b) illustrate the main synchronization steps with the CSP policy depending on whether participating devices are online or not. A device always

sends content updates created locally[6] to the cloud (step 1). As in most cloud storage synchronization protocols, control servers keep track of all updates each device owns and notify other online devices about new updates. Indeed, most cloud storage services employ control servers for keeping track of online devices and of the updates in content metadata. For example, in Dropbox, clients and control servers exchange periodic "keep alive" messages, and the client informs servers of new updates whenever available (see Chapter 2 Section 2.3).

In order to implement CSP, control servers should include, in each update notification sent to a device d, the identifier of one other device that is currently online and can serve the update, if one is available. The control server selects this *source* device s among all online devices that currently have the same update. The selection policy may be random or any other approach, aiming at keeping the load balanced across participating devices. As soon a device d receives the update notification, it tries to establish a connection with the indicated device s in order to download the update. If the download succeeds (step 2), d informs the control server the identifier of s, so that the server can update the bonus to be given the user who owns s accordingly. If the download fails, or no online device to serve the update was available, d retrieves the content directly from the cloud (step 3). After retrieving the content, either from another device s (contact) or the cloud, device d becomes able to serve it.

7.2.3 Anonymous Sharing Policy (ASP)

ASP also allows users to serve each other content and receive bonus as a free extra storage space in the cloud as she offloads the service traffic. However, unlike CSP, the user devices eligible to serve content from a folder do not need to be associated with (i.e., share) the folder, or any other shared folder. In this case, the provider may invite user devices that are often online to participate in this policy, thus increasing the chances of offloading downloads. As for the previous policy, users accepting the offer inform the maximum upload capacity they are willing to contribute ($\mathcal{B}_i$). Moreover, since participants do not own the content they will offload from the provider, they must also inform the provider the maximum local storage they are willing to contribute ($\mathcal{D}_i$).

Figure 7.2 (a and b) illustrates the main synchronization steps with ASP into play in the same cases shown in Figure 7.1 (a and b). The policy works as follows. The devices associated with shared folders receive from the control server a list with a random subset of all participating devices. The list contains the devices that will be

[6]Updates represent modifications in shared folders: new files, metadata and the commands that manipulate files, e.g., to delete files or create sub-folders.

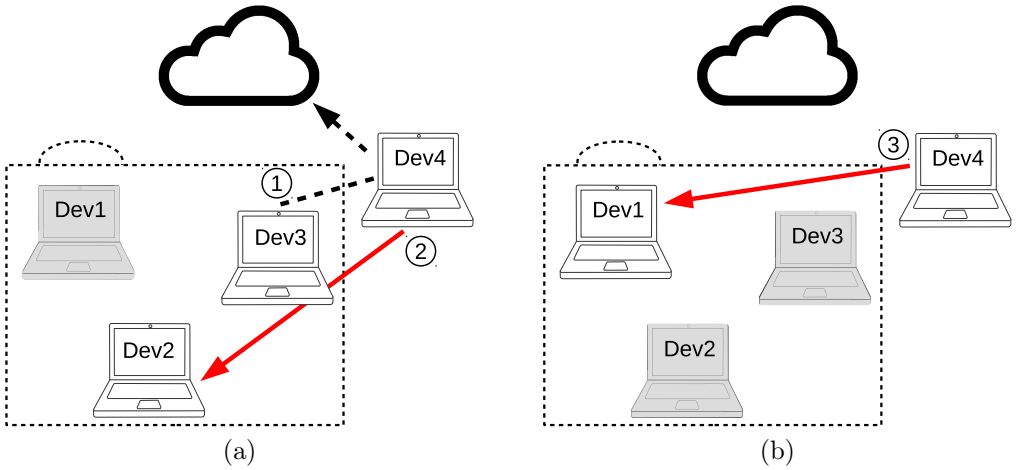


Figure 7.2: Common cases of downloads in cloud storage with Anonymous Sharing Policy: Dev 1, 2 and 3 are associated with the same shared folder, (a) Dev 3 sends new updates to the cloud through Dev 4 (often online but not associated with the folder) which serves Dev 2 (Dev 1 is offline), (b) Dev 1 is back online and retrieves new updates from the Dev 4 (Dev 2 and 3 are offline).

responsible for storing content updates. We refer to them as *anonymous* devices. The list of anonymous devices may change periodically so as to reflect the availability and promote load balancing of the participating devices.

A device g sends an update it generates to one of the anonymous devices, which is responsible for forwarding it to the cloud (step 1). The update is concluded only after g receives a confirmation from the control server. Otherwise, device g sends the update directly to the cloud. Control servers should keep track of all updates stored at each anonymous device. This does not represent a significant extra cost to the servers, as they already have to keep track of all content owned by the anonymous device. The rest of the policy is very similar to CSP. Control servers notify other devices about the online anonymous devices that can serve content updates by providing the identifier of one such *source* devices in each update notification. As soon as a device d receives a new update notification, it tries to establish a connection with the indicated source device s in order to download the update. If the download succeeds (steps 2), device d informs the control server the anonymous device s that served it. Otherwise, device d retrieves the content from the cloud. Note that, unlike in CSP, when a device comes back online and receives a notification of updates in one of its shared folders, it may be able to retrieve the content from an anonymous device even if none of its contacts are currently online (step 3).

Note also that the proposed design of ASP brings up two issues that are not found in CSP. First, users participating in ASP have an additional disk cost to store other users' data. Second, users are susceptible to privacy violations when forwarding their data to anonymous devices, since such devices may belong to unknown users (unlike in CSP, where data is exchanged only between contacts). The first issue has a direct impact on how user utilities are computed, as we will discuss later. Regarding privacy concerns, we argue that each update must be encrypted before being uploaded to an anonymous device (step 1), as in similar approaches adopted by some cloud storage services [Mager et al., 2012]. An investigation of the efficiency of alternative encryption mechanisms in this context is left for future work.

7.2.4 Service Cost Reduction and User Bonus

Building upon our model, notably Equations 7.3 and 7.4, we now derive the expressions for the cost reduction $\mathcal{P}_s$ experienced by the service provider and the net benefit of a user $\mathcal{P}_i$ for both CSP and ASP. As in Section 7.1, cost reductions and bonuses should be recomputed periodically, for pre-defined time windows w. We present the derived expressions again focusing on an arbitrary window w.

For both policies, the service provider cost reduction is proportional to the amount of bytes served from client devices, as thus offloaded from the provider's cloud servers during the considered time window. That is:

$$\mathcal{P}_s = (\beta - \alpha * \kappa) * \mathcal{O}, \tag{7.6}$$

where α and β are the previously defined storage and transfer price units and κ is the bonus (i.e., number of storage units) the provider offers eligible users per unit of byte offloaded. In turn, $\mathcal{O}$ is the total amount of bytes offloaded from the server, given by the total number of bytes $\mathcal{O}_i$ served by all devices owned by user i (and thus offloaded from the provider's servers) so that $\mathcal{O} = \sum \mathcal{O}_i$. Note that, while the total amount of bytes offloaded $\mathcal{O}$ causes a reduction on transfer costs (β parameter), it also increases the costs related to storage (α parameter), as extra storage in the cloud servers is offered to participating users (κ bytes per offloaded byte).

We note that the bonus κ expires after a certain period of time (e.g., a time window w)[7]. The provider is able to reach cost reduction by setting $\kappa < \frac{\beta}{\alpha}$. In order to compute the bonus of a given user i, we assume that control servers keep track of the total volume of data served by each device of user i ($\mathcal{O}_i$) participating in the policy.

[7] Different options of bonus usage by users can be adopted, e.g., a single time window w or divided into equal parts by various windows.

The net benefit of user i for participating in the policy in turn is given by:

$$\mathcal{P}_i = V_i * \mathcal{O}_i * \kappa - V_i * \mathcal{O}_i * \mathcal{C}_i. \tag{7.7}$$

The utility change for user i is computed from two components. At the one hand, the volume the user has offloaded from the provider $\mathcal{O}_i$ is converted into a bonus using κ, and weighted by the user i's valuation for a byte V_i. On the other hand, from the computed bonus, we discount a penalty related to the user's resources used for offloading the servers. The penalty is also proportional to the bytes offloaded by the user (i.e., $\mathcal{O}_i$), considering a factor $\mathcal{C}_i$, which is the penalty per byte stored/transferred for the provider. Again, user i's valuation for a byte V_i is used as a weight for the penalty.[8]

The above expressions for $\mathcal{P}_s$ and $\mathcal{P}_i$ are the same for both CSP and ASP. The models of the two policies diverge when it comes to the penalty imposed on individual users. User devices participating in CSP only serve content they already share. Thus no extra storage, but rather only transfer capacity, is required from them. In ASP, in turn, user devices store content shared by others. Thus, extra storage is required as well. In both cases, we express the penalty factor imposed on user i, $\mathcal{C}_i$, as the fraction of the available resources that are actually used for offloading the provider.

In other words, given $\mathcal{B}_i$ and $\mathcal{D}_i$ the maximum upload and storage capacity (in total number of bytes) user i is willing to offer for serving others, the penalty factor imposed on a user participating in CSP is defined as $\mathcal{C}_i^{CSP} = \frac{\mathcal{O}_i}{\mathcal{B}_i}$. For ASP, in turn, the penalty factor is defined as $\mathcal{C}_i^{ASP} = \frac{\mathcal{O}_i}{\mathcal{B}_i} + \frac{\mathcal{O}_i}{\mathcal{D}_i}$. The closer the used resources get to the available capacity, the greater the penalty imposed on users.

7.3 Evaluation Methodology

In this section, we present the methodology to evaluate the efficiency of the two content sharing policies, CSP and ASP, using our model (Equations 7.5), delimiting specific scenarios where both, provider and users, experience improvements in their utilities. This methodology is organized into two parts: (i) *simulations* to obtain each policy dynamic (e.g., volumes of offload per user) over real cloud storage service traces, and (ii) *reference setup* to analyze model parameters as well as various policies adoption scenarios from simulations.

[8]Note that we could have used a distinct users' valuation for the penalty, in place of V_i. We opt for a single variable for simplicity.

7.3.1 Simulating the Content Sharing Policies

We perform a trace-driven simulation of CSP and ASP policies, using the Dropbox datasets described in Chapter 4. Specifically, for each user device in the traces, we track the exact time periods during which the device is connected to the service provider control servers (online) and which periods it is offline. We then select the set of eligible user devices that can participate in the policy. As presented in Section 7.2, for CSP, the set of eligible devices associated with each shared folder corresponds to all devices that synchronize any content in that particular folder. For ASP, this set corresponds the top $T\%$ devices with longest average online periods. In this case, we assume the users who own those devices may be interested in participating in ASP as they can earn more bonuses due to their availability to offload the provider. We vary user adoption of the policy by varying the fraction of eligible devices who accepts participating in the policy. In our evaluation, we set the number of eligible devices in ASP (T) equal to the total number of eligible devices for CSP, so as to be able to fairly compare both policies.

We then carry out trace-driven simulations of each policy by following each upload/download event observed in the input traces. As described in Section 7.2, an upload to a shared folder identifies a device that can serve future updates of that content (source device). Thus, whenever there is a download request from a destination device d, we first check whether there is any source device s that owns the update requested in the given download event *and* is currently online. If there is such s, the update is downloaded from source s to destination d. If there are multiple devices that can act as source, we select one randomly. Otherwise the download is served from the cloud, as it happens in the traces currently.

Note that, in our simulations, we can only track potential source devices located within the particular networks covered by the traces we own. As consequence, a source device only will be able to serve updates related to avoidable downloads (see Chapter 6). The generation of updates by devices outside the analyzed networks cannot be tracked, as they are not in the traces. Nevertheless, both CSP and ASP could be applied to offload more downloads from the cloud by tracking potential sources from other (neighboring) network domains. We leave an investigation of the efficiency of both policies in such scenario for the future as it requires traces with a broader view of content sharing in cloud storage.

7.3.2 Reference Setup

Our goal is to investigate scenarios where both the provider and end users are satisfied (utility improvements). To that end, we search for possible scenarios by varying the fraction of eligible user devices that accept to participate in the policy (*policy adoption*), and the units of bonus given to participants per byte offloaded (κ), which is the model key parameter to adjust utility improvements. The remaining parameters are kept fixed, using realistic cloud storage parameter values, as described next. Each simulation covers a one month time window w, and we compute results for all non-overlapping windows in our traces. In particular, the total number of bytes offloaded by each user ($\mathcal{O}_i$) is dictated by the dynamics of each policy applied on the input traces.

Regarding the provider configuration, we set parameters $\mathcal{S}$ and $\mathcal{T}$ based on our traces to represent realistic cost estimates for a cloud storage service in the analyzed networks. Specifically, we use the upload and download volumes estimated (see Chapter 6 Table 6.1) in each trace to define the total number of bytes transferred from/to the cloud ($\mathcal{T}$). Moreover, we add up all folder updates (uploads and non-avoidable downloads) observed in each trace to determine the total number of bytes stored in the cloud ($\mathcal{S}$).

All remaining parameters are set according to values currently adopted by main cloud infrastructure providers and cloud storage services. For example, we take as reference for α and β the prices for standard storage and data transference announced by Amazon S3 at the time of the writing[9]: $\alpha = \$0.03/GB/month$ and $\beta = 3 * \alpha$, i.e., transfer price is three times higher than storage within w. In addition, only traffic from cloud to devices (download) is charged by Amazon.

Given the lack of accurate estimates of the sources of revenue for the provider (parameter $\mathcal{R}$), we decide to express it in terms of the provider's total cost (storage and transfers). Specifically, we consider three scenarios for the provider:

- *Low cost*: total cost is 25% of its total revenue;
- *Moderate cost*: total cost is 50% of its total revenue;
- *High cost*: total cost is 75% of its total revenue.

Regarding user configuration, we assume that only free users adopt CSP and ASP, as they have more chances of reaching the space limit, and thus become interested in the storage bonuses offered by the policy. We then leave for future work analyses of scenarios where paying users also participate in policies and their improvements

[9] `https://aws.amazon.com/s3/pricing`

compared to free users. Given the focus on free users in this work, we set the user storage in the cloud based on current free storage capacity adopted by Dropbox ($X_i = 2GB$). Moreover, we assume the same valuation V_i for all users, i.e., a unique user class. We note that, under the free user class, our simulation results are not affected by the valuation parameter, since P_i=0.

Finally, we set the maximum upload capacity $\mathcal{B}_i$ a user is willing to offer in the time window w to 83 GB, which corresponds to the maximum volume a user can transfer in a month under a 256 kpbs connection. Similarly, the maximum storage capacity $\mathcal{D}_i$ is set to 500 GB. Both parameters are conservative, as they correspond to the basic setup offered by main ISPs and PC vendors nowadays.

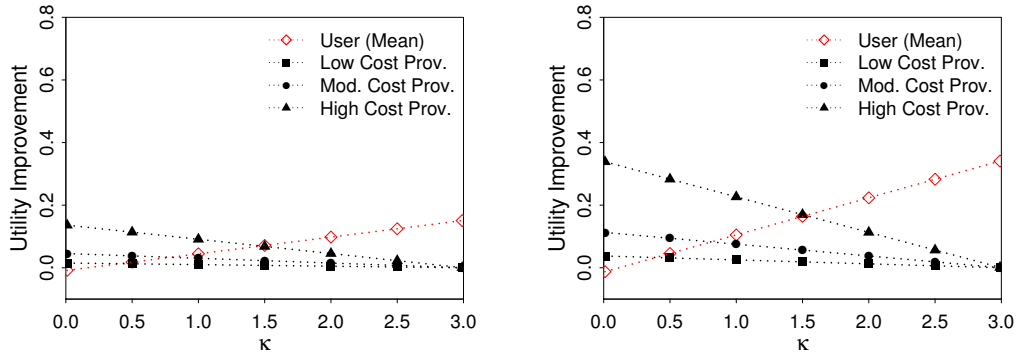
7.4 Results

In this section we present our most representative results. We analyze the proposed content sharing policies addressing three questions: (Q1) *Is it possible to reach a scenario where both the provider and users obtain utility improvements?* (Q2) *How does the fraction of user devices who adopt the policy impact its efficiency?* and (Q3) *What is the penalty imposed on users that adopt the policy?* We address each question by showing results corresponding to averages computed for all 1 month periods (i.e., w=1 month) in each trace.

7.4.1 Utility Tradeoffs

To tackle Q1, we analyze how the units of bonus given to each participant (parameter κ) affect utilities for the provider and users. As one might expect, user utility should increase with κ, whereas the provider utility should decrease with κ. One could then argue that different operation points exist (based on different κ values) where both provider and user experience utility improvements, and selecting the best value is a matter of weighting the satisfaction of both parties. Here we take no side and argue that a "good" operation point for κ is the value for which both provider and (an average) user experience the *same* improvements in their utilities. We refer to this point as the *target* κ. Thus, we search for a κ value where the utility improvements of the provider equal the average utility improvements of a user (average over all users who joined the policy), i.e., $I_s = \bar{I}_i$.

Figures 7.3-a and 7.3-b show the utility improvements for both provider and users (on average) as a function of κ for CSP and ASP policies, respectively, on one of our

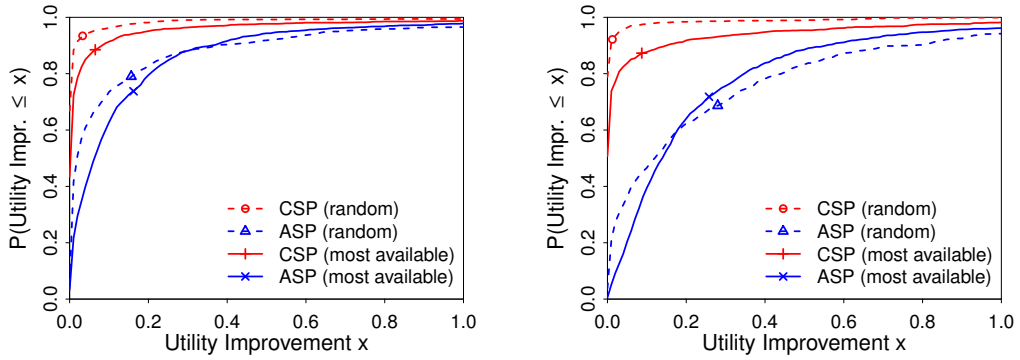


(a) CSP (Campus-1) (b) ASP (Campus-1)

Figure 7.3: Utility improvement for the provider and users as functions of κ with 10% policy adoption: best tradeoff occurs at the intersections of the provider and user utility improvement curves.

traces (Campus-1). Utility improvements for the three provider scenarios discussed in the previous section are shown in each figure. These results are computed assuming that the 10% eligible devices with the longest average online periods participate in the policy (policy adoption of 10%). Alternatively, adoption could be defined based on a random selection among the eligible devices. However, as we will discuss later, both policies are sensitive to churn in participating devices, i.e., devices often alternating between online and offline states [Stutzbach and Rejaie, 2006a]. Thus, we start with a scenario that favors the proposed policies. Results for the other traces are qualitatively similar, and thus are omitted for the sake of brevity.

As shown in the figures, for any scenario – low, moderate or high cost provider – and for any given value of κ, the utility improvements for both provider and user are higher for ASP than for CSP. This is due to the strategy used by ASP to select eligible devices among the most available ones, and not only those that share a folder as in CSP. Thus, even though the approach used to simulate policy adoption is the same, it turns out that the devices participating in ASP tend to be more available in the system, and thus, serve others (offloading the provider) more often. Note that the higher the costs of the provider, the more benefits the provider achieves from using either CSP or ASP, i.e., the higher the utility improvements experienced from applying either policy. For example, considering the high cost provider and ASP, the target κ is approximately 1.5, leading to an improvement of around 17% for both provider and users. In the case of a moderate cost provider, the target κ is 0.8, and the improvements of both parties are still relevant, falling around 8%, while in the low cost case, the improvements, although positive (up to 3%), may not be attractive to justify policy deployment.



(a) Campus-1 (b) PoP-2

Figure 7.4: User utility improvements at target κ for a high cost provider. The average utility improvements are shown as symbols in each curve.

We delve further into the efficiency of both ASP and CSP from the perspective of individual users by plotting the Cumulative Distribution Function (CDF) of the utility improvements for each individual user (and not only average improvements, as in Figure 7.3), fixing κ at the target value. Figure 7.4 shows the distributions for two of our traces, considering a high cost provider. Note that we here present two set of results: one assuming that devices that adopt the policy are selected among the most available ones ("most available" curves) and one assuming that they are selected randomly from the eligible devices ("random" curves). For reference, the average utility improvements are shown as symbols in each curve. Results are very similar for moderate and lower providers as well. Let's focus first on the "most available" adoption scenario. For ASP, around 27% of users experience a utility improvement above average, while for CSP this fraction falls in the 8-13% range. Thus, given the strategy used to select eligible devices, ASP leads to higher improvements for users not only on average but also for individual users. The same is true for the "random" adoption. However, both policies become less effective as some participating devices tend to be less available to serve others.

7.4.2 Impact of User Adoption

So far we have assumed a fixed policy adoption at 10%. We now turn to Q2 and analyze the impact the adoption has on policy efficiency. We use the Campus-1 and PoP-2 traces as representative of university and PoP traces, as those traces have the largest traffic volumes in terms of avoidable download. We also focus on the scenario

of a high cost provider, where cost reductions should be more valuable, but the same overall conclusions hold for the other scenarios and traces as well.

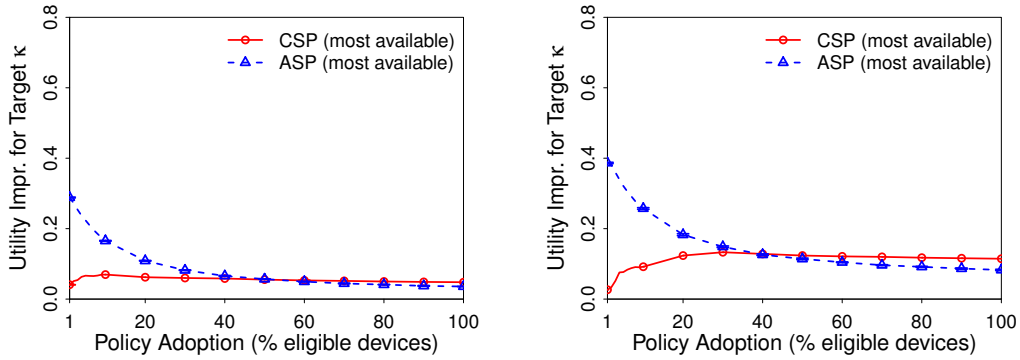
Specifically, we vary the fraction of eligible devices that effectively adopt the policy from 1% to 100% (i.e., all eligible devices), assuming the "most available" and "random" selection cases, and measure the utility improvements at the target κ value. For each given fraction, since source devices of policies are non-deterministic, we report average results for 15 independent simulation runs.

Figure 7.5 shows the utility improvements at target κ for a high cost provider as a function of adopting devices with the most available eligible devices selection. As shown in this figure, improvements are reached for all cases. The largest improvements are 29% in Campus-1 and 39% in PoP-2 with 1% adoption. However, note that the utility improvements experienced under ASP drop sharply as more devices adopt the policy. Given our approach to select adopting devices, as more devices join the policy, it becomes more likely that, at the time of a download event, the device that currently holds the requested update (and thus can serve it) is offline. In other words, the adoption by a larger number of devices with shorter average online period greatly hurts the efficiency of ASP.

For CSP, instead, the utility improvements actually increase with the fraction of adopting devices, dropping slightly or remaining roughly stable after a peak. For example, for the Campus-1 trace, the highest utility improvements – 7% – are obtained with 10% of adopting devices. For PoP-2, 30% adopting devices leads to the highest utility improvements (13%). This happens because when the fraction of adopting devices is too low, multiple devices associated with the same shared folder are rarely simultaneously online to serve each other. Thus, offload opportunities increase as more devices join CSP. Indeed, the total percentage of bytes offloaded from the provider ($\mathcal{O}/\mathcal{T}$ – not shown in the figures) varies from 3% to 15%[10] when the percentage of devices adopting CSP goes from 1% to 100%. However, as more devices participate in CSP, these offloads are distributed across a larger number of source device, and thus the average user utility improvement does not increase accordingly (or may actually slightly drop).

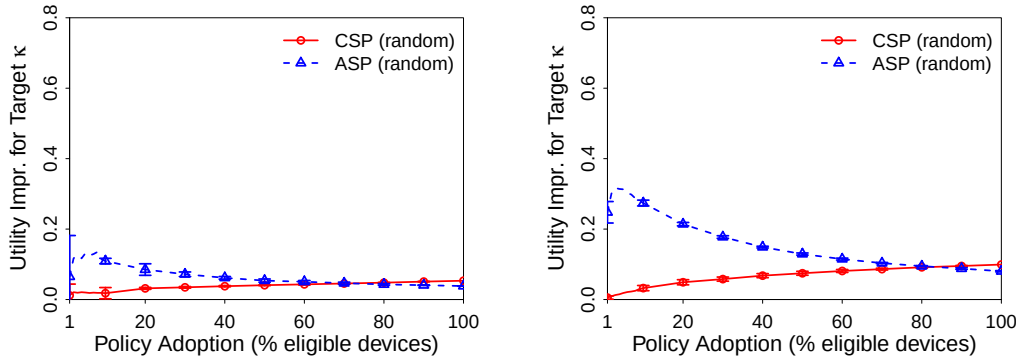
It is also worth noting that the utility improvements are higher in PoP-2 than in Campus-1. This happens because the volumes of downloads, in particular avoidable downloads, are larger in PoP-2 dataset as shown in Table 6.1 in Chapter 6. The larger the volume of content shared among user devices, the more the provider can reduce its

[10]This is 6% to 80% of the avoidable downloads shown in Table 6.1 in Chapter 6.



(a) Campus-1 (b) PoP-2

Figure 7.5: Utility improvements at target κ for a high cost provider as a function of adopting devices.

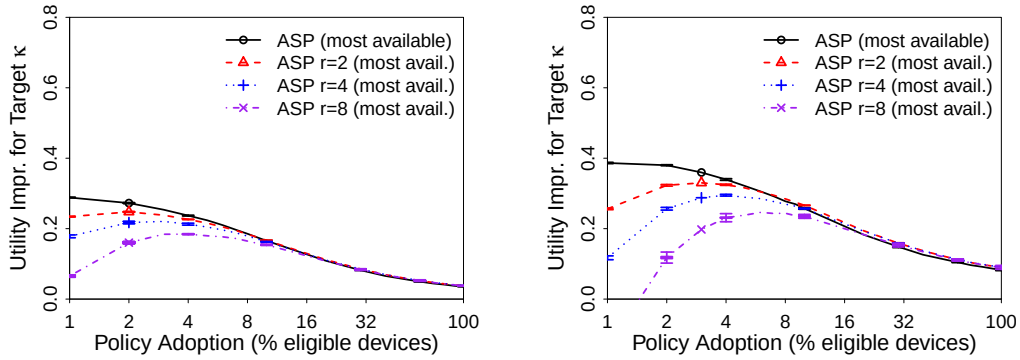


(a) Campus-1 (b) PoP-2

Figure 7.6: Policies with random eligible devices selection and utility improvements at target κ for a high cost provider as a function of adopting devices.

costs with either policy, while the bonuses offered to users as incentive for offloading also increase their satisfaction with the service.

Now, we analyze both policies with random selection of eligible devices. Figure 7.6 shows the utility improvements at target κ for a high cost provider as a function of adopting devices. We can observe, again, that improvements are reached for all cases, although lower than the most available selection case, shown in Figure 7.5. For example, by comparing random selection with most available one at the adoption with the largest improvement for the later, in PoP-2, ASP have reduced from 39% to 23% (1% adoption) and CSP from 13% to 6% (30% adoption). Similarly, in Campus-1, ASP have reduced from 29% to 13% (1% adoption) and CSP from 7% to 2% (10% adoption). Interestingly, for ASP, such reductions becomes negligible from 10% adoption. It means that ASP is



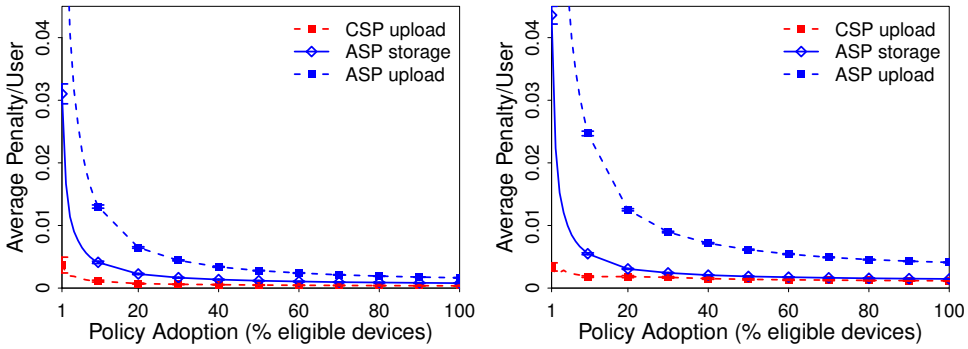
(a) Campus-1 (b) PoP-2

Figure 7.7: ASP without/with replications (r equal to 2, 4 and 8) and utility improvements at target κ for a high cost provider as a function of adopting devices. Note a base 2 log scale is used for x axis.

less dependent of the most frequent devices collaboration when adoption is not very low (above 10%). On the other hand, CSP, notably, suffers higher performance decrease with random selection and reaches results similar to the most available selection case only with 100% (all) eligible devices adoption.

Given that ASP reached the best utility improvements, we also investigate whether replication of the same update across multiple anonymous devices can increase even more the observed utility improvements for this policy. To accomplish this investigation, we set each anonymous device in our simulations to send the received update to other r different online anonymous devices, after sending the update to the cloud. Figure 7.7 shows utility improvements at the target κ as a function of adopting devices for ASP without replication (previous analyses) and ASP with replication, where r assumes values 2, 4 and 8. We note this figure shows only the most available eligible devices selection case, however, results for random selection are qualitatively similar and lead to the same conclusions.

As one can observe, ASP without replications leads to the highest utility improvements for a low adoption percentage (less than 10%), whereas improvements become roughly similar with higher adoption for the four ASP settings. This happens because replication increases devices resources usage (i.e., upload and storage capacities), while the total offload volume is about the same in low adoption scenarios. On the other hand, replication increases total offload in higher adoption scenarios and keeps utility improvements roughly similar to ASP without replication, as shown in Figure 7.7. For example, with 100% of adoption devices, the total percentage of bytes offloaded from the provider ($\mathcal{O}/\mathcal{T}$ – not shown in the figure) is 15% and 18% for



(a) Campus-1 (b) PoP-2

Figure 7.8: Average user penalty vs. policy adoption for a high cost provider.

non-replication and replication ($r = 8$) respectively in Campus-1 and 22% and 25% in PoP-2.[11] Ultimately, our results show that ASP without replication is the best choice to reach a good tradeoff between the provider and users' utilities improvements.

7.4.3 Penalty on Participants

Finally, we turn to Q3 and analyze the penalties each policy imposes on users. Figure 7.8 shows the average value of the penalty factor (parameter C_i) imposed on users participating in CSP and ASP as a function of the fraction of adopting devices with the most available eligible devices in Campus-1 and PoP-2 traces. For ASP, we separately show the penalty in terms of storage ($\frac{\mathcal{O}_i}{\mathcal{D}_i}$) and upload ($\frac{\mathcal{O}_i}{\mathcal{B}_i}$) resource consumption.

As one can observe in Figure 7.8, the penalties decrease as more devices join the policy. But, overall, the user resources consumed for offloading the provider tend to be very small, even with low adoption of the policy. This holds for both policies, but especially for CSP. Let's consider the Campus-1 trace. With only 1% of policy adoption, users participating in ASP use, on average, 12% and 3% of their available upload data transference and storage, respectively, for offloading the provider. For CSP, the fraction of available upload data transference used is only 0.3%. On the other hand, the PoP-2 trace, which, as mentioned, has the greatest opportunities for offloading, when policy adoption is 1%, users participating in ASP use, on average, 23% and 4% of their available upload data transference and storage, respectively. For CSP, the fraction of available upload data transference used is very low again, only 0.4%.

[11] This is 85% and 99% in Campus-1 and 87% and 99% in PoP-2 of the avoidable downloads shown in Table 6.1 in Chapter 6.

Thus, the proposed content sharing policies require low commitment of resources from users.

7.5 Summary

In this Chapter, we presented the results we obtained so far with respect to the RG3 of this dissertation, which regards modeling the tradeoffs between the provider and end users interests in cloud storage services. We addressed the following broad question in RG3: *How to model costs and benefits of cloud storage services so to help providers in assessing the effectiveness of alternative policies that aim at increasing both profitability and user satisfaction?* We can summarize our main contributions in order to answer this question as follows:

- We propose a general model for the costs and benefits of cloud storage considering both users and providers. Our model is based on *utility functions* that capture, in an abstract level, the provider profit and users satisfaction given by the benefits minus the costs of the service for each party.
- We investigate two alternative policies for the current cloud storage sharing architecture, which count on user collaboration to reduce provider costs. We evaluate the effectiveness of both policies by applying our proposed model to assess user and provider utilities in various realistic scenarios.

The next chapter concludes this dissertation and points out future research directions.

Chapter 8

Conclusions and Future Research Directions

In this chapter we summarize the main achievements of this dissertation. Moreover, we also present a discussion on future research directions. First, we show this dissertation achievements in Section 8.1 and break down this section in one subsection for each research goal containing a brief summary of the goal and the obtained results. Next, we present a broader discussion on future research directions and open research issues in Section 8.2.

8.1 Current Achievements

Recall from Chapter 1 that this dissertation aims at tackling three main research goals:

- RG1 - Modeling User Behavior and Workload Patterns
- RG2 - Analyzing the Effects of Social Features
- RG3 - Modeling the Tradeoffs Between User and Service Provider Interests

In the following we present obtained results for each research goal.

8.1.1 RG1 - Modeling User Behavior and Workload Patterns

In RG1, we focused on characterizing and modeling user behavior and associated workload in cloud storage services. To that end, we proposed a hierarchical user behavior model composed of three layers: user session, namespaces and data transmission. Unlike other services analyzed by previous user behavior modeling efforts [Menascé et al., 2003; Costa et al., 2004; Veloso et al., 2006; Borges et al., 2012], cloud storage services present a new challenge to user behavior modeling which consists in capturing the behavior of users during file synchronization and collaborative work. Towards capturing the effects of such functionalities, our model explicitly represents data modifications and sharing. It captures content modifications that lead to data exchanges in the network, ultimately reflecting how content is propagated in the cloud storage network of shared folders.

Our main contributions regarding modeling user behavior and workload patterns in cloud storage services can be summarized as follows:

- We proposed a user behavior hierarchical model based on the main users' actions in cloud storage and parameterized it using passive measurements gathered from a real service and networks traffic. Our model parameterization is robust. Indeed, we have characterized each model component from datasets collected in four different networks and we have found the same statistical distributions for each component in the four networks and close agreement among their parameters, as shown in Appendix A.

- We developed and validated a synthetic workload generator (CloudGen) which reproduces user sessions, traffic volume and data sharing patterns. We have offered CloudGen as free software to the community.[1]

- We illustrated the applicability of CloudGen in a case study, where the impact of large user populations and high content sharing is explored. Our experiments indicate that a hypothetical 4-fold increase in both user population and content sharing could lead to 30 times more network traffic.

These contributions were presented in two conference papers [Gonçalves et al., 2014b,a] and one journal paper [Gonçalves et al., 2016b].

[1] CloudGen is available at `http://cloudgen.sourceforge.net`.

8.1.2 RG2 - Analyzing the Effects of Content Sharing

In RG2, we focused on analyzing the effects of content sharing features on costs and user activity of cloud storage services. To that end, we showed, through measurements in the four networks, that a large fraction of downloads is associated with content shared among multiple user devices located within the same network. This is worrisome given our observations that downloads volume are higher than uploads one in cloud storage. In addition to this, the most important cloud infrastructure providers (e.g., Amazon, Google and Microsoft), which are used by most cloud services, charge the traffic from the cloud to devices (i.e., download). Cloud services providers thus have an incentive to reduce download traffic. Concerning this issue, we proposed and evaluated an alternative synchronization architecture that uses caches to offload storage servers from such downloads. Moreover, we showed the importance for cloud storage services to maintain content sharing functionality, as its related social features have a high correlation with user activity.

Our main contributions regarding the analysis of the effects of content sharing and its derived social features on cloud storage services can be summarized as follows:

- We assessed the impact of content sharing on the traffic of a popular cloud storage service (Dropbox) by quantifying avoidable downloads in different edge networks. We showed that 57–70 percent of downloads generated by Dropbox users are associated with content shared among multiple devices within these networks, and up to 25% of this traffic happens to synchronize content that was observed in the same network. Thus, such download traffic is *avoidable*.

- We proposed an alternative synchronization architecture for cloud storage services with focus on content sharing as Dropbox. We showed that this architecture is cost-effective to offload service from handling almost all avoidable downloads. For example, we found that local cache could cost-effectivelly offload servers around 92% of the costs for serving avoidable downloads, reaching 95–97% if we consider hypothetical scenarios with larger user populations and content sharing.

- We analyzed and quantified the relationship between content sharing and user activity levels in Dropbox by means of social features. We showed that the user activity (i.e., data volume synchronized with the cloud) is highly correlated with the amount of shared folders (namespaces) of the user: the measured Spearman correlation coefficient was 0.73-0.98 in the four networks, which indicates a highly

positive dependence between the two variables. In short, we have found the most active users are likely to share more content in Dropbox.

These contributions were presented in two conference papers [Gonçalves et al., 2015, 2016c] and one journal paper [Gonçalves et al., 2016a].

8.1.3 RG3 - Modeling the Tradeoffs Between User and Service Provider Interests

In RG3, we focused on analyzing the costs and benefits of cloud storage for the service providers and end users jointly. Toward this RG, we dealt with important issues of cloud storage services as the majority of free users in the service, the attractiveness of content sharing for users (and the corresponding costs for the provider), and the urge to keep users satisfied in such a competitive market. Ultimately, such issues clearly pressure providers to reduce costs in order to maintain profitability. On the other hand, policies adopted to reduce costs must not hurt user satisfaction, at the penalty of reducing service attractiveness. Identifying strategies that deal with such tradeoff by balancing the satisfaction of both parties is of utmost importance for providers to remain in the market. Nevertheless, most prior studies that have investigated this tradeoff analyzed costs and benefits from a single point of view, either focusing on end users [Naldi and Mastroeni, 2013; Shin et al., 2014] or service providers [Wang et al., 2012; Wu et al., 2013]. A prior effort that jointly analyzed user and provider perspectives [Lin and Tzeng, 2014] focused on the cost to replicate users' data into multiple disks and ignores the role of data transfer costs and content sharing, which are important concerns in cloud storage.

Towards filling this gap, our main contributions regarding the investigation on the costs and benefits of cloud storage services can be summarized as follows:

- We proposed a general model for the costs and benefits of cloud storage services considering both users and providers interests. This model is based on utility functions that represent the benefits minus the costs of the service for each party. The greater the utilities are, the more satisfied providers and users become. Our proposal appeals for capturing in a simple (but representative) model key components of cloud storage. Thus, we focused on the main aspects related to the service resource consumption, notably storage and bandwidth.

- We applied our model to evaluate two alternative policies for the current cloud storage sharing architecture. Our results show that the investigated policies are

advantageous for providers and users, leading to utility improvements of up to 39% for both parties with respect to current settings. Improvements come from offloading the provider infrastructure by counting on users to help transferring shared content via a Peer-to-Peer (P2P) architecture. This strategy becomes attractive to users as the provider gives bonus to those who collaborate. Best utility improvements are achieved with a limited number of collaborators (around 30% of the eligible users' devices). Gains are more relevant in scenarios where users share lots of contents, but those collaborating still need to contribute with only a small amount of resources – e.g., no more than 23% of the Internet bandwidth they are willing to offer to the provider.

These contributions were presented in two conference papers [Gonçalves et al., 2016; Gonçalves et al., 2017].

8.2 Future Work

In this section, we discuss future research directions we plan on pursuing. Those new research directions are organized again by the three research goals adopted in this dissertation.

8.2.1 Modeling user behavior and workload patterns

We intend to extend our user behavior model to include three aspects: (i) different users' profile, (ii) mobile devices and (iii) dynamic of content propagation network. With this extension, we envision to become our workload generator (CloudGen) able to generate workload for various cloud services in addition to storage. In particular, we intend to extend CloudGen to services that count on a cloud infrastructure to promote communication among people or collaborative work through content sharing. Current examples of such services are popular mobile applications as Whats App and Telegram[2], which offers means for users to share multimedia content in addition to communication via audio and video streaming. In the following, we describe the aforementioned aspects we intend to explore.

First, the inclusion of different users' profile in our model will allow the simulation of populations with particular characteristics. In fact, we observed evidence of those

[2] Those are one of the most popular applications for communication among people at the time of this writing: `www.whatsapp.com,www.telegram.org`.

profiles given that some components of our model were fitted by mixed probability distributions (e.g., inter-sessions and volume of modifications). We assumed a single profile for users in our current model in order to propose a simple but still accurate model to generate workload for cloud storage services. We then intend to offer optional parameters to set multiple users profiles, which can improve accuracy and applicability of the model to different services workload.

Second, we intend to add characteristics of mobile devices to our current model. These devices have limited capacity for energy and storage, so as they do not download shared content automatically as it happens in PC applications. Thus, we have to include in our model cases where only part of devices receives updates automatically. Mobile devices might also impact on the content propagation network. We should expect a larger number of devices per namespace, given the sum of PCs and mobile devices of same users.

The lifespan of sharings may also become longer when taking into consideration mobile devices. Shared files not stored locally in these devices should be often requested, while users are interested in these files. In this context, epidemic models, usually applied to model lifespan of multimedia content on the Internet [Figueiredo et al., 2014; Ribeiro, 2014], will be investigated as alternative for our current approach based on the number of devices per namespaces (Figure 5.9) and short lifespan (Figure 6.2b). We intend to develop mobile (Android/iOS) applications and ask volunteers to install and collect data about their sharing activity via mobile devices.

Third, we intend to explore the content propagation network to predict future groups of collaboration among users, the volume of content they are going to share as well as the lifespan of the group. Such predictions are important as they may allow service providers to provision cloud resources in a cost-effective way. Moreover, they are important for the development of alternative content sharing policies, e.g., CSP/ASP (Section 7.2), in order to offload the provider cloud infrastructure and reduce costs.

8.2.2 The effects of content sharing on cloud services

The use of distributed architectures to support content sharing in cloud storage, as we proposed in this dissertation, is in line with *edge computing* [Wang et al., 2017; Mach and Becvar, 2017; Mao et al., 2017]. This research area aims at pushing part of computing, network control and storage from the centralized cloud to the network edges, e.g., base stations and access points, so as to enable computation-intensive and latency-critical services at the resource-limited devices, mainly mobile devices. In this

context, we intent to extend our proposal of distributed architecture by going toward two new directions that are considered research challenges for edge computing area.

The first research challenge concerns the distribution and management of resources. We showed the cost-effectiveness of a local cache node for cloud storage services in universities and ISP PoPs. However, there are several possibilities of those nodes placement to cover users in multiple networks spread over a certain region, country or globe, and, at same time, avoid an expensive growth of the provider's infrastructure. The challenge is then to find an optimal way where to physically place the nodes depending on expected users demands, while considering the provider relative cost-effectiveness, e.g., the relative cost savings (rcs), shown in Section 6.2.3, for several cache nodes. Can cloud storage caches take advantage of existing and well-developed researches on content distribution networks (CDNs) [Erman et al., 2009]? We thus intend to develop new models and simulations to investigate this issue.

The second research challenge regards user mobility. A recent study based on NEC cloud storage show new evidence of massive adoption of content sharing by users [Gracia-Tinedo et al., 2016]. Yet, we consider that further studies based on real systems traces are still necessary to consolidate our evidence that content sharing is a key feature for users. More than that, these studies will allow a deep understanding on *the mobility of users* who share content across different networks. This might be an important aspect on caching decision, as the frequent mobility of users will cause frequent handovers among cache nodes. Thus, techniques to manage mobility, as the ones surveyed by Wang et al. [2017], should be explored in the context of cloud storage services in order to allow users to access services caches seamlessly. To achieve such study on mobility of users, we intent to collect data from mobile cloud services, not specifically cloud storage, as mentioned in the previous section.

8.2.3 Tradeoffs between user and service provider benefits

Finally, we envision four directions for future work regarding our studies about tradeoffs between user and service provider benefits. First, let's consider content sharing policies. In this dissertation, we showed the contribution of anonymous devices (not participating in sharings) is relevant to offload data transferences from the provider's cloud infrastructure. The next investigation step, then, concerns the user privacy issue, i.e., how efficient are current encryption techniques to protect users data that are relayed through various anonymous devices to/from the cloud? We believe that experimental evaluation of existing security approaches (e.g., [Mager et al., 2012]) could address this issue. Note that, CPU usage, now, becomes an important variable to be

incorporated in the provider and users' cost component, given the introduction of encryption methods in content sharing policies.

Second, we intend to apply our cost-benefit model to analyze *alternative storage policies* in cloud storage services. These policies aim at satisfying users with different characteristics (i.e., profiles) as well as reducing the provider's operational cost with storage. Previous studies analyzed the tradeoffs between users and providers' satisfaction for the number of content replicates and discount for reduced replication [Lin and Tzeng, 2014] as well as the tradeoffs between the use of fast input/output solid state disks or low cost magnetic disks [Cheng et al., 2016]. However, a cost-benefit tradeoff not yet investigated concerns the frequency users access personal content in the cloud. We showed, in Section 6.1, users' folders (or namespaces) in Dropbox have high temporal locality and short lifespan. Providers may take advantage of this characteristic by storing data not often accessed by users in low cost/slow access storage ("data freezing"), whereas users should obtain a service price discount (bonus) by owing data with such characteristic[3]. Nevertheless, providers may require a specific time period to make such data available again at user request. Such unavailable data period may represent a penalty for users, thus reducing their earned bonus or overall utility. A relevant study we intend to accomplish is the proposal of a method to analyze the best tradeoff between users' and the provider's utilities, regarding the right time to start "freezing" users' content in the cloud.

The third future direction encompasses the inclusion of different users classes on current analyses derived from the proposed general cost-benefit model. We explored utility improvements reached by free users from content sharing policies that is attractive for this user's class (see Chapter 7 Section 7.3.2). However, different classes of paying users also can benefit from alternative service policies. The aforementioned storage policies, in particular, are based on a service price reduction as bonus. Thus, paying users may become interested in the bonuses offered by policies. To investigate this issue, we will include paying users classes on the model by estimating the parameters service valuation and service price for those classes.

Finally, the fourth future direction concerns the exploration of "what if scenarios" from synthetic traces generated by CloudGen. Our goal is to evaluate the performance of alternative architectures and service policies in scenarios that are different from the current one, observed in our datasets, but still realistic. The initial scenario we intend to investigate is the feedback effect of content sharing policies CSP and ASP on costs and benefits for both providers and users. Examples of such feedback are the

[3] Some infrastructure providers offer low cost storage services developed for infrequently accessed data as Amazon Glacier (`https://aws.amazon.com/glacier`).

growth of online period of devices and volumes of content shared by users in order to earn bonuses. We intend to analyze cost-benefit tradeoffs of such possible changing on users' behavior empirically from synthetic traces.

Flash crowd in cloud services [de Paula Junior et al., 2015] is another scenario we intend to analyze. Synthetic traces that simulate unexpected surge in users' access in the service will allow us to analyze the efficiency of the architecture based on local caches and policies to reduce the impact of such flash crowd events. Moreover, realistic synthetic traces will also allow us to achieve empirical studies of pricing for cloud resources, thus corroborating for existing theoretical studies [Xu and Li, 2013]. Example of issues we can analyze empirically, regarding the freemium business model, are (a) the percentage of paying users that is enough for the service to become profitable (i.e., revenue greater than operating costs), and (b) which price those users should pay for the service.

Bibliography

Abad, C. L., Yuan, M., Cai, C. X., Lu, Y., Roberts, N., and Campbell, R. H. (2013). Generating Request Streams on Big Data using Clustered Renewal Processes. *Performance Evaluation*, 70(10):704–719.

Akhil Gupta (2016). Scaling to exabytes and beyond. `https://blogs.dropbox.com/tech/2016/03/magic-pocket-infrastructure`.

Almeida, J. M., Krueger, J., Eager, D. L., and Vernon, M. K. (2001). Analysis of educational media server workloads. In *Proc. of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 21–30.

Amrehn, P., Vandenbroucke, K., Hossfeld, T., Moor, K. D., Hirth, M., Schatz, R., and Casas, P. (2013). Need for Speed? On Quality of Experience for Cloud-based File Storage Services. In *Proc. of the PQS*, pages 184–190.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

Barford, P. and Crovella, M. (1998). Generating Representative Web Workloads for Network and Server Performance Evaluation. *ACM SIGMETRICS Performance Evaluation Review*, 26(1):151–160.

Benevenuto, F., Rodrigues, T., Cha, M., and Almeida, V. (2012). Characterizing user navigation and interactions in online social networks. *Information Sciences*, 195:1 – 24.

Bermudez, I., Mellia, M., Munafò, M. M., Keralapura, R., and Nucci, A. (2012). DNS to the Rescue: Discerning Content and Services in a Tangled Web. In *Proc. of the ACM Internet Measurement Conference*, pages 413–426.

Bessani, A., Mendes, R., Oliveira, T., Neves, N., Correia, M., Pasin, M., and Verissimo, P. (2014). Scfs: a shared cloud-backed file system. In *USENIX Annual Technical Conference*, pages 169–180.

Bestavros, A. and Krieger, O. (2014). Toward an open cloud marketplace: Vision and first steps. *Internet Computing, IEEE*, 18(1).

Bocchi, E., Drago, I., and Mellia, M. (2015a). Personal Cloud Storage Benchmarks and Comparison. *IEEE Transactions on Cloud Computing*, PP(99):1–14.

Bocchi, E., Drago, I., and Mellia, M. (2015b). Personal Cloud Storage: Usage, Performance and Impact of Terminals. In *Proc. of the IEEE CloudNet*.

Borges, A., Gomes, P., Nacif, J., Mantini, R., de Almeida, J. M., and Campos, S. (2012). Characterizing SopCast Client Behavior. *Computer Communication*, 35(8):1004–1016.

Busari, M. and Williamson, C. (2002). ProWGen: A Synthetic Workload Generation Tool for Simulation Evaluation of Web Proxy Caches. *Computer Network*, 38(6):779–794.

Buyya, R., Yeo, C. S., and Venugopal, S. (2008). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications*, pages 5–13.

Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A. F. D., and Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software-Practice & Experience*, 41(1):23–50.

Calzarossa, M. C., Massari, L., and Tessera, D. (2016). Workload characterization: A survey revisited. *ACM Computing Surveys*, 48(3):48:1–48:43.

Casas, P., Fischer, H., Suette, S., and Schatz, R. (2013). A first look at quality of experience in personal cloud storage services. In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 733–737.

Casas, P. and Schatz, R. (2014). Quality of Experience in Cloud Services: Survey and Measurements. *Comput. Netw.*, 68(1):149–165.

Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S. (2009). Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networks*, 17(5):1357–1370.

Chaabouni, R., Sánchez-Artigas, M., and García-López, P. (2014). Reducing costs in the personal cloud: Is bittorrent a better bet? In *14th IEEE International Conference on Peer-to-Peer Computing*, pages 1–10. ISSN 2161-3559.

Chaabouni, R., Sánchez-Artigas, M., García-López, P., and Pàmies-Juàrez, L. (2016). The power of swarming in personal clouds under bandwidth budget. *Journal of Network and Computer Applications*, 65:48 – 71.

Chandrashekhara, S., Marcus, K., Subramanya, R. G., Karve, H. S., Dantu, K., and Ko, S. Y. (2015). Enabling automated, rich, and versatile data management for android apps with bluemountain. In *Proc. of the 7th USENIX Conference on Hot Topics in Storage and File Systems*, pages 2–2. USENIX Association.

Chen, F., Mesnier, M. P., and Hahn, S. (2014). Client-Aware Cloud Storage. In *Proc. of the IEEE Symposium on Mass Storage Systems and Technologies*.

Cheng, Y., Iqbal, M. S., Gupta, A., and Butt, A. R. (2016). Provider versus tenant pricing games for hybrid object stores in the cloud. *IEEE Internet Computing*, 20(3):28–35.

Chetty, M. and Buyya, R. (2002). Weaving computational grids: How analogous are they with electrical grids? *Computing in Science & Engineering*, 4(4):61–71.

Costa, C. P., Cunha, I. S., Borges, A., Ramos, C. V., Rocha, M. M., Almeida, J. M., and Ribeiro-Neto, B. (2004). Analyzing client interactivity in streaming media. In *Proc. of the 13th International Conference on World Wide Web*, pages 534–543.

Crunchbase Inc. (2017). Dropbox. `https://www.crunchbase.com/organization/dropbox/funding-rounds`.

Cui, Y., Lai, Z., Wang, X., Dai, N., and Miao, C. (2015). QuickSync: Improving Synchronization Efficiency for Mobile Cloud Storage Services. In *Proc. of the ACM International Conference on Mobile Computing and Networking*.

D'Agostino, R. B. and Stephens, M. A. (1986). *Goodness-of-Fit Techniques*. Marcel Dekker, New York, USA.

de Paula Junior, U., Drummond, L. M. A., de Oliveira, D., Frota, Y., and Barbosa, V. C. (2015). Handling flash-crowd events to improve the performance of web applications. In *Proc. of the ACM Symposium on Applied Computing*, pages 769–774.

Dee, M. (2015). Inside LAN Sync. `https://blogs.dropbox.com/tech/2015/10/inside-lan-sync`.

Delimitrou, C., Sankar, S., Vaid, K., and Kozyrakis, C. (2011). Accurate Modeling and Generation of Storage I/O for Datacenter Workloads. In *Proc. of the EXERT*.

Drago, I., Mellia, M., Munafò, M. M., Sperotto, A., Sadre, R., and Pras, A. (2012). Inside Dropbox: Understanding Personal Cloud Storage Services. In *Proc. of the 12th ACM Internet Measurement Conference*.

Duarte, R., Vieira, A. B., Cunha, I., and Almeida, J. M. (2015). Impact of provider failures on the traffic at a university campus. In *Proc. of IFIP Networking Conference*, pages 1–9.

Erman, J., Gerber, A., Hajiaghayi, M. T., Pei, D., and Spatscheck, O. (2009). Network-aware forward caching. In *Proc. of the 18th International Conference on World Wide Web*, pages 291–300.

Figueiredo, F., Almeida, J. M., Matsubara, Y., Ribeiro, B., and Faloutsos, C. (2014). Revisit Behavior in Social Media: The Phoenix-R Model and Discoveries. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Finamore, A., Mellia, M., Meo, M., Munafò, M. M., and Rossi, D. (2011). Experiences of Internet Traffic Monitoring with Tstat. *IEEE Network*, 25(3):8–14.

Goh, K.-I., Kahng, B., and Kim, D. (2001). Universal Behavior of Load Distribution in Scale-Free Networks. *Physical Review Letters*, 87(27):278701.

Gonçalves, G., Drago, I., da Silva, A. P. C., Vieira, A. B., and de Almeida, J. M. (2014a). Modeling the Dropbox Client Behavior. In *Proc. of the IEEE International Conference on Communications*, pages 1332–1337.

Gonçalves, G., Drago, I., da Silva, A. P. C., Vieira, A. B., and de Almeida, J. M. (2017). Cost-benefit tradeoffs of content sharing in personal cloud storage. In *Proc. of the IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*.

Gonçalves, G., Drago, I., Vieira, A., Silva, A., and Almeida, J. (2016a). The impact of content sharing on cloud storage bandwidth consumption. *IEEE Internet Computing, Special Issue Measuring the Internet July/August*. ISSN 1089-7801.

Gonçalves, G., Drago, I., Vieira, A., Silva, A., Almeida, J., and Mellia, M. (2016b). Workload models and performance evaluation of cloud storage services. *Elsevier Computer Networks, DOI: http://dx.doi.org/10.1016/j.comnet.2016.03.024*. ISSN 1389-1286.

Gonçalves, G., Drago, I., Vieira, A. B., da Silva, A. P. C., and de Almeida, J. M. (2014b). Characterizing and Modeling the Dropbox Workload. In *Proc. of the Brazilian Symposium on Networks and Distributed Systems (in Portuguese)*.

Gonçalves, G., Drago, I., Vieira, A. B., da Silva, A. P. C., and de Almeida, J. M. (2015). Analyzing the Impact of Dropbox Content Sharing on an Academic Network. In *Proc. of the Brazilian Symposium on Networks and Distributed Systems (in Portuguese)*.

Gonçalves, G., Vieira, A. B., da Silva, A. P. C., and de Almeida, J. M. (2016c). Trabalho Colaborativo em Serviços de Armazenamento na Nuvem: Uma Análise do Dropbox. In *Proc. of the Brazilian Symposium on Networks and Distributed Systems (in Portuguese)*.

Gonçalves, G. D., Drago, I., Borges, A. V., Couto, A. P., and de Almeida, J. M. (2016). Analysing costs and benefits of content sharing in cloud storage. In *Proc. of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks*, pages 43–45.

Gracia-Tinedo, R., García-López, P., Gómez, A., and Illana, A. (2016). Understanding data sharing in private personal clouds. In *Proc. of the IEEE International Conference on Cloud Computing*.

Gracia-Tinedo, R., Sanchez Artigas, M., Moreno-Martinez, A., Cotes, C., and Garcia Lopez, P. (2013). Actively measuring personal cloud storage. In *Proc. of the 6th International Conference on Cloud Computing*, pages 301–308.

Gracia-Tinedo, R., Tian, Y., Sampé, J., Harkous, H., Lenton, J., García-López, P., Sánchez-Artigas, M., and Vukolic, M. (2015). Dissecting ubuntuone: Autopsy of a global-scale personal cloud back-end. In *Proc. of the IMC*.

Graupner, S., Konig, R., Machiraju, V., Pruyne, J., Sahai, A., and Moorsel, A. (2003). Impact of virtualization on management systems. Technical report, Technical report, Hewlett-Packard Labs.

Hobfeld, T., Schatz, R., Varela, M., and Timmerer, C. (2012). Challenges of qoe management for cloud applications. *Communications Magazine, IEEE*, 50(4):28–36.

Hu, W., Yang, T., and Matthews, J. N. (2010). The Good, the Bad and the Ugly of Consumer Cloud Storage. *ACM SIGOPS Operating Systems Review*, 44(3):110–115.

Jain, R. K. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, New York, USA.

Jeners, N. and Prinz, W. (2014). Metrics for cooperative systems. In *Proc. of the 18th International Conference on Supporting Group Work*, pages 91–99.

Jin, S. and Bestavros, A. (2001). GISMO: A Generator of Internet Streaming Media Objects and Workloads. *SIGMETRICS Performance Evaluation Review*, 29(3):2–10.

Joe-Wong, C., Ha, S., and Chiang, M. (2015). Sponsoring mobile data: An economic analysis of the impact on users and content providers. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, pages 1499–1507.

Karunakaran, S. and Sundarraj, R. (2015). Bidding strategies for spot instances in cloud computing markets. *Internet Computing, IEEE*, 19(3).

Ken Florance (2016). How Netflix Works With ISPs Around the Globe to Deliver a Great Viewing Experience. `https://media.netflix.com/en/company-blog/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experien`

Koorapati, N. (2014). Streaming File Synchronization. `https://blogs.dropbox.com/tech/2014/07/streaming-file-synchronization/`.

Krishnamurthy, D., Rolia, J. A., and Majumdar, S. (2006). A Synthetic Workload Generation Technique for Stress Testing Session-Based Systems. *IEEE Transactions on Software Engineering*, 32(11):868–882.

Kumar, V. (2014). Making freemium work. *Harvard Business Review*, pages 701–703.

Lam, I. (2015). Farewell, wuala! pioneering secure storage shuts down, recommends tresorit. `https://tresorit.com/blog/encrypted-cloud-storage-wuala-is-shutting-down-and-recommends-tresorit`.

Lee, J. (2003). An end-user perspective on file-sharing systems. *Commun. ACM*, 46(2):49–53.

Li, Z., Jin, C., Xu, T., Wilson, C., Liu, Y., Cheng, L., Liu, Y., Dai, Y., and Zhang, Z.-L. (2014). Towards Network-Level Efficiency for Cloud Storage Services. In *Proc. of ACM Internet Measurement Conference*, pages 115–128.

Li, Z., Wilson, C., Jiang, Z., Liu, Y., Zhao, B. Y., Jin, C., Zhang, Z.-L., and Dai, Y. (2013). Efficient batched synchronization in dropbox-like cloud storage services. pages 307–327.

Lin, C. Y. and Tzeng, W. G. (2014). Game-theoretic strategy analysis for data reliability management in cloud storage systems. In *Software Security and Reliability (SERE), 2014 Eighth International Conference on*, pages 187–195.

Liu, S., Huang, X., Fu, H., and Yang, G. (2013). Understanding Data Characteristics and Access Patterns in a Cloud Storage System. In *Proc. of the IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*, pages 327–334.

Mach, P. and Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys Tutorials*, 19(3):1628–1656.

Mager, T., Biersack, E., and Michiardi, P. (2012). A measurement study of the wuala on-line storage service. In *12th International Conference on Peer-to-Peer Computing*, pages 237–248.

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, 19(4):2322–2358.

Marshall, C. and Tang, J. C. (2012). That syncing feeling: Early user experiences with the cloud. In *Proc. of the Designing Interactive Systems Conference*, pages 544–553.

Marshall, C. C., Wobber, T., Ramasubramanian, V., and Terry, D. B. (2012). Supporting research collaboration through bi-level file synchronization. In *Proc. of the 17th ACM International Conference on Supporting Group Work*, pages 165–174.

Massey, C., Lennig, T., and Whittaker, S. (2014). Cloudy forecast: An exploration of the factors underlying shared repository use. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2461–2470.

Menascé, D., Almeida, V., Riedi, R., Ribeiro, F., Fonseca, R., and Jr., W. M. (2003). A hierarchical and multiscale approach to analyze e-business workloads. *Performance Evaluation*, 54(1):33 – 57.

Mishra, A. K., Hellerstein, J. L., Cirne, W., and Das, C. R. (2010). Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters. *SIGMETRICS Performance Evaluation Review*, 37(4):34–41.

Mitzenmacher, M. (2004). Dynamic Models for File Sizes and Double Pareto Distributions. *Internet Mathematics*, 1(3):305–334.

Moreno, I. S., Garraghan, P., Townend, P., and Xu, J. (2014). Analysis, Modeling and Simulation of Workload Patterns in a Large-Scale Utility Cloud. *IEEE Transactions on Cloud Computing*, 2(2):208–221.

Mosberger, D. and Jin, T. (1998). Httperf — a Tool for Measuring Web Server Performance. *SIGMETRICS Performance Evaluation Review*, 26(3):31–37.

Mulazzani, M., Schrittwieser, S., Leithner, M., Huber, M., and Weippl, E. (2011). Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space. In *Proc. of the USENIX Security Symposium*.

Naldi, M. (2014). Balancing leasing and insurance costs to achieve total risk coverage in cloud storage multi-homing. *Economics of Grids, Clouds, Systems, and Services*, pages 146–158.

Naldi, M. and Mastroeni, L. (2013). Cloud Storage Pricing: A Comparison of Current Practices. In *Proc. of the ACM International Workshop on HotTopiCS*, pages 27–34.

Nebeling, M., Geel, M., Syrotkin, O., and Norrie, M. C. (2015). Mubox: Multi-user aware personal cloud storage. In *Proc. of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1855–1864.

NIST (2011). The nist definition of cloud computing. `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`.

Oliveira, D., Carvalho, P., and Lima, S. R. (2015). Towards cloud storage services characterization. In *Proc. of IEEE 18th International Conference on Computational Science and Engineering (CSE)*, pages 129–136.

Palviainen, J. and Rezaei, P. P. (2015). The next level of user experience of cloud storage services: Supporting collaboration with social features. In *Software Engineering Conference (ASWEC), 2015 24th Australasian*, pages 175–184. ISSN 1530-0803.

Partridge, C. and Allman, M. (2016). Ethical considerations in network measurement papers. *Commun. ACM*, 59(10):58–64.

Perkins, D., Agrawal, N., Aranya, A., Yu, C., Go, Y., Madhyastha, H. V., and Ungureanu, C. (2015). Simba: Tunable end-to-end data consistency for mobile apps.

In *Proc. of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 7:1–7:16.

Peterson, L. L. and Davie, B. S. (2012). *Computer networks: a Systems Approach*. Elsevier.

Pindyck, R. and Rubinfeld, D. (2013). *Microeconomics*. Pearson Education, Inc.

Rabinovich, M. and Spatschek, O. (2002). *Web Caching and Replication*. Addison-Wesley Longman Publishing, Boston, MA, USA. ISBN 0-201-61570-3.

Ribeiro, B. (2014). Modeling and predicting the growth and death of membership-based websites. In *Proc. of the 23rd international conference on World wide web*.

Rogowsky, M. (2013). Dropbox Is Doing Great, But Maybe Not As Great As We Believed. `http://onforb.es/1Kle8IE`.

Salmon, B., Schlosser, S. W., Cranor, L. F., and Ganger, G. R. (2009). Perspective: Semantic data management for the home. In *Proc. of the 7th USENIX Conference on File and Storage Technologies*, volume 9, pages 167–182.

Shami, N. S., Muller, M., and Millen, D. (2011). Browse and discover: Social file sharing in the enterprise. In *Proc. of the ACM 2011 Conference on Computer Supported Cooperative Work*, pages 295–304.

Shen, H. and Li, Z. (2015). New bandwidth sharing and pricing policies to achieve a win-win situation for cloud provider and tenants. *IEEE Transactions on Parallel and Distributed Systems*, PP(99):1–1.

Shen, Z., Luo, J., Zimmermann, R., and Vasilakos, A. (2011). Peer-to-peer media streaming: Insights and new developments. *Proc. of the IEEE*, 99(12):2089–2109.

Shin, J., Jo, M., Lee, J., and Lee, D. (2014). Strategic management of cloud computing services: Focusing on consumer adoption behavior. *IEEE Transactions on Engineering Management*, 61(3):419–427.

Silber, J. (2015). Shutting down ubuntu one file services. `http://blog.canonical.com/2014/04/02/shutting-down-ubuntu-one-file-services`.

Stutzbach, D. and Rejaie, R. (2006a). Understanding churn in peer-to-peer networks. In *Proc. of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202, Rio de Janeriro, Brazil.

Stutzbach, D. and Rejaie, R. (2006b). Understanding Churn in Peer-to-Peer Networks. In *Proc. of the ACM Internet Measurement Conference*, pages 189–202.

Tanenbaum, A. S. and Van Steen, M. (2007). *Distributed Systems*. Prentice-Hall.

Tang, W., Fu, Y., Cherkasova, L., and Vahdat, A. (2003). MediSyn: A Synthetic Streaming Media Service Workload Generator. In *Proc. of the Int. Workshop on Network and Op. Syst. Support for Digital Audio and Video*, pages 12–21.

Veloso, E., Almeida, V., Jr., W. M., Bestavros, A., and Jin, S. (2006). A Hierarchical Characterization of a Live Streaming Media Workload. *IEEE/ACM Transactions on Networking*, 14(1):133–146.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, USA.

Vieira, C., Bittencourt, L. F., and Madeira, E. R. M. (2013). Towards a paas architecture for resource allocation in iaas providers considering different charging models. In *International Conference on Grid Economics and Business Models*, pages 185–196. Springer.

Vieira, C., Bittencourt, L. F., and Madeira, E. R. M. (2014). Reducing costs in cloud application execution using redundancy-based scheduling. In *Proc. of the IEEE/ACM International Conference on Utility and Cloud Computing*, pages 117–126.

Voida, A., Olson, J. S., and Olson, G. M. (2013). Turbulence in the clouds: Challenges of cloud-based information work. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2273–2282.

Vrable, M., Savage, S., and Voelker, G. M. (2012). Bluesky: A cloud-backed file system for the enterprise. In *Proc. of the 10th USENIX conference on File and Storage Technologies*, pages 19–19.

Wang, H., Shea, R., Wang, F., and Liu, J. (2012). On the impact of virtualization on dropbox-like cloud file storage/synchronization services. In *Proc. of the 20th International Workshop on Quality of Service*, pages 11:1–11:9.

Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., and Wang, W. (2017). A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, 5:6757–6779.

Wang, W., Li, B., and Liang, B. (2014). Dominant resource fairness in cloud computing systems with heterogeneous servers. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 583–591.

Wickremasinghe, B., Calheiros, R. N., and Buyya, R. (2010). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *Proc. of 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 446–452.

Wu, Z., Butkiewicz, M., Perkins, D., Katz-Bassett, E., and Madhyastha, H. V. (2013). Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In *Proc. of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 292–308.

Xu, H. and Li, B. (2013). A study of pricing for cloud resources. *SIGMETRICS Perform. Eval. Rev.*, 40(4):3–12.

Yeo, H.-S., Phang, X.-S., Lee, H.-J., and Lim, H. (2014). Leveraging client-side storage techniques for enhanced use of multiple consumer cloud storage services on resource-constrained mobile devices. *Journal of Network and Computer Applications*, 43:142–156.

Yuksel, M., Sikdar, B., Vastola, K. S., and Szymanski, B. (2000). Workload Generation for NS Simulations of Wide Area Networks and the Internet. In *Proc. of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 93–98.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud Computing: State-of-the-Art and Research Challenges. *Journal of Internet Services and Applications*, 1:7–18.

Zhang, Q., Zhani, M., Jabri, M., and Boutaba, R. (2014a). Venice: Reliable virtual data center embedding in clouds. In *Proc. of INFOCOM*, pages 289–297.

Zhang, Y., Dragga, C., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H. (2014b). ViewBox: Integrating Local File Systems with Cloud Storage Services. In *Proc. of the USENIX Conf. on File and Storage Technologies*, pages 119–132.

Zink, M., Suh, K., Gu, Y., and Kurose, J. (2009). Characteristics of youtube network traffic at a campus network – measurements, models, and implications. *Computer Networks*, 53(4):501 – 514.

Appendix A

Best-Fitted Distributions

Table A.1 lists parameters of the components of our model. The PDF of the Log-normal distribution (parameters μ and σ) is $p_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(ln(x)-\mu)^2}{2\sigma^2}}$. The PDF of the Pareto Type I distribution (parameters α and β) is $p_X(x) = \frac{\alpha\beta^\alpha}{x^{\alpha+1}}$. The PDF of the Pareto Type II distribution (with two parameters α and κ) is $p_X(x) = \frac{\alpha\kappa^\alpha}{(x+\kappa)^{\alpha+1}}$. Finally, the PDF of the Negative Binomial distribution (parameters r and p, and Gamma function Γ) is $p_X(x) = \frac{\Gamma(x+r)}{\Gamma(r)x!} p^r(1-p)^x$ and we add 1 to generated random numbers, since the Namespaces per Device and Devices per Namespace components are always larger than zero.

Table A.1: Parameters of the components of our hierarchical model.

Component	Model	Parameters	
Session Duration (seconds)	Log-normal	Campus-1	$\mu = 8.222$ $\sigma = 1.395$
		Campus-2	$\mu = 8.484$ $\sigma = 1.680$
		PoP-1	$\mu = 8.209$ $\sigma = 1.464$
		PoP-2	$\mu = 8.492$ $\sigma = 1.545$
Inter-Session Time (seconds) Range 1	Log-normal	Campus-1	$\mu = 7.748$ $\sigma = 1.178$
		Campus-2	$\mu = 7.489$ $\sigma = 1.294$
		PoP-1	$\mu = 8.307$ $\sigma = 1.329$
		PoP-2	$\mu = 7.971$ $\sigma = 1.308$
Inter-Session Time (seconds) Range 2	Log-normal	Campus-1	$\mu = 11.057$ $\sigma = 0.189$
		Campus-2	$\mu = 11.033$ $\sigma = 0.189$
		PoP-1	$\mu = 11.045$ $\sigma = 0.232$
		PoP-2	$\mu = 10.984$ $\sigma = 0.202$
Inter-Session Time (seconds) Range 3	Log-normal	Campus-1	$\mu = 12.367$ $\sigma = 0.559$
		Campus-2	$\mu = 12.469$ $\sigma = 0.602$
		PoP-1	$\mu = 12.295$ $\sigma = 0.508$
		PoP-2	$\mu = 12.275$ $\sigma = 0.516$
Namespaces per Device	Negative Binomial	Campus-1	$r = 0.666$ $p = 0.391$
		Campus-2	$r = 1.056$ $p = 1.287$
		PoP-1	$r = 0.483$ $p = 0.634$
		PoP-2	$r = 0.470$ $p = 1.119$
Number of Modifications	Pareto Type II	Campus-1	$\alpha = 1.329$ $\kappa = 5.863$
		Campus-2	$\alpha = 1.336$ $\kappa = 6.079$
		PoP-1	$\alpha = 1.534$ $\kappa = 5.601$
		PoP-2	$\alpha = 1.591$ $\kappa = 7.264$
Inter-Modification Time (seconds)	Log-normal	Campus-1	$\mu = 3.016$ $\sigma = 2.149$
		Campus-2	$\mu = 3.367$ $\sigma = 2.400$
		PoP-1	$\mu = 3.486$ $\sigma = 2.177$
		PoP-2	$\mu = 3.748$ $\sigma = 2.286$
Devices per Namespace	Negative Binomial	Campus-1	$r = 0.429$ $p = 0.467$
		Campus-2	$r = 0.400$ $p = 0.352$
		PoP-1	$r = 0.425$ $p = 0.226$
		PoP-2	$r = 0.328$ $p = 0.306$
Upload Volume (kB) $\leq$ 10 MB	Log-normal	Campus-1	$\mu = 2.288$ $\sigma = 2.582$
		Campus-2	$\mu = 1.945$ $\sigma = 3.306$
		PoP-1	$\mu = 3.417$ $\sigma = 2.721$
		PoP-2	$\mu = 2.464$ $\sigma = 2.893$
Upload Volume (kB) $>$ 10 MB	Pareto Type I	Campus-1	$\alpha = 1.291$ $\beta = 10,000$
		Campus-2	$\alpha = 0.806$ $\beta = 10,000$
		PoP-1	$\alpha = 1.432$ $\beta = 10,000$
		PoP-2	$\alpha = 1.312$ $\beta = 10,000$