Workload models and performance evaluation of cloud storage services

Workload Models and Performance Evaluation of Cloud Storage Services

Glauber D. Gonçalves[a,*], Idilio Drago[b], Alex B. Vieira[c], Ana Paula Couto da Silva[a], Jussara M. Almeida[a], Marco Mellia[b]

[a] *Universidade Federal de Minas Gerais, Brazil*
[b] *Politecnico di Torino, Italy*
[c] *Universidade Federal de Juiz de Fora, Brazil*

Abstract

Cloud storage systems are currently very popular with many companies offering services, 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 systems, the underlying processes that generate workload to the system have not been deeply studied.

This paper 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. We parameterize our model using passive measurements gathered from 4 different networks. 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 actual data from operational networks. 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. CloudGen is a valuable tool for administrators and developers interested in engineering and deploying cloud storage services.

Keywords: Cloud Storage, Models, Measurements

*Corresponding author
Email addresses: ggoncalves@dcc.ufmg.br (Glauber D. Gonçalves), idilio.drago@polito.it (Idilio Drago), alex.borges@ufjf.edu.br (Alex B. Vieira), ana.coutosilva@dcc.ufmg.br; (Ana Paula Couto da Silva), jussara@dcc.ufmg.br (Jussara M. Almeida), marco.mellia@polito.it (Marco Mellia)

1. Introduction

Cloud storage is a data-intensive Internet service that synchronizes files with the cloud and among different end user devices, such as PCs, tablets and smartphones. It offers the means for customers to easily backup data and to perform collaborative work, with files being automatically uploaded and synchronized. Cloud storage is already one of the most popular Internet services, generating a significant amount of traffic [1]. Well-established players such as Dropbox, as well as giants like Google and Microsoft, face a fierce competition for customers. Currently, Dropbox leads the market with more than 400 million users by 2015,[1] and Google and Microsoft show a fast growth [2], thanks to solutions that are more and more integrated into Windows or Android Operating Systems.

Both established and new players could greatly benefit from a deep understanding of the workload patterns that cloud storage services have to face in order to develop cost-effective solutions. However, several aspects make the analysis of cloud storage workloads a challenge. As the stored content is private and synchronization protocols are mostly proprietary, the knowledge of how these systems work is limited outside companies running the services. It is thus very hard to obtain large scale data that could drive workload analyses, given also the widespread use of encryption for both data and control messages [3].

Indeed, we are aware of only a few recent efforts to analyze the characteristics of cloud storage services [1], focusing either on architectural design aspects [4, 5], quality of experience [6, 7], or benchmark-driven performance studies [3, 8, 9]. Yet, a characterization of the underlying client processes that generate workload to the system, notably the data transfer patterns that emerge from such processes, is only partly addressed by our preliminary efforts presented in [10, 11]. Such knowledge is key to analyze the impact of these services on network bandwidth requirements as well as to the design of future services.

Towards filling this gap, this paper performs a thorough investigation of the workload experienced by Dropbox, the currently most popular cloud storage service. We propose a hierarchical model that describes client behavior in successive Dropbox sessions. Within each session, our model captures file system modifications and content sharing among users and devices, as well as client interactions with Dropbox servers while storing and retrieving files. We parameterize the model based on the analysis of Dropbox traffic traces collected in four different networks, namely two university networks – one in South America and one in Europe, and two European Internet Service Provider (ISP) networks. We offer these traces to the community, to foster future studies.

Next, we use the model to drive the development of a new synthetic workload generator, called *CloudGen*. Our validation experiments, comparing real and synthetic workloads, show that CloudGen is able to reproduce the behavior of Dropbox customers. CloudGen is a valuable tool that allows network administrators and cloud storage developers to analyze the traffic volume of cloud

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

storage as well as to evaluate future system optimization and developments in various synthetic, but still realistic, workload scenarios.

Given the steady increase in Dropbox user base [12], we illustrate the applicability of CloudGen by evaluating how network traffic would change in edge networks, such as ISP and campus networks, when customer population grows and when users share more content via cloud storage. Our results suggest that a hypothetical 4-fold increase in both user population and content sharing could lead to 30 times more cloud storage traffic, eventually challenging the capacity of such systems.

In sum, our main contributions are the following:

- We propose a hierarchical model of the Dropbox client behavior and parameterize 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 develop and validate a synthetic workload generator (CloudGen) that reproduces user sessions, traffic volume and data sharing patterns. We offer CloudGen as free software to the community.[2]
- We illustrate the applicability of CloudGen in a case study, where the impact of larger user populations and more content sharing are explored.

The paper is organized as follows. Section 2 presents the background on Dropbox. Our data collection methodology is presented in Section 3. Section 4 describes our client behavior model and characterizes its components, whereas our workload generator is introduced in Section 5. We illustrate CloudGen applicability in Section 6, exploring possible future scenarios for cloud storage usage. Section 7 reviews related work, while Section 8 summarizes our contributions and lists future work. Finally, additional details of the characterization of our model components are provided in Appendix A.

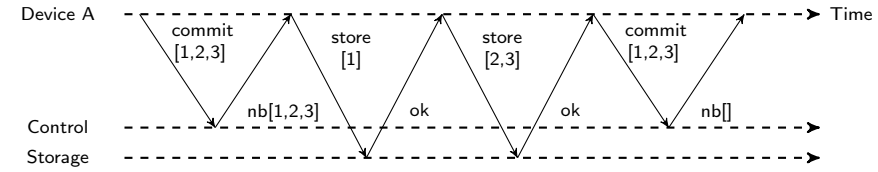
2. Overview of Dropbox

2.1. Architecture

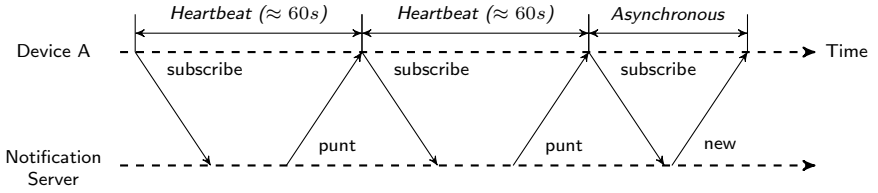
Dropbox relies on thousands of servers that are split into two major groups: *control* and *storage* servers. Control servers are responsible for: (i) user authentication; (ii) management of the Dropbox file system metadata; and (iii) management of device sessions. Storage servers are in charge of the actual storage of data, and are outsourced to the Amazon cloud by the time of writing.

Users can link several 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 clients in this work, since they are responsible for

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



(a) Control and Storage Protocols



(b) Notification Protocol

Figure 1: Basic operation of Dropbox client protocols.

most workload in cloud storage [1], although our model and workload generator can be extended to other usage scenarios as well.

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 every participant in the sharing as a *shared folder*.

Internally, Dropbox treats the initial folders selected at configuration time and all shared folders indistinctly. They are all called *namespaces*, and each 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 attached to a monotonic *journal ID (JID)*, representing the namespace latest version [13].

2.2. Client 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 in steady state.

When a Dropbox device identifies new local changes (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 in 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 [3].

As soon as chunks/bundles are ready for transmission, the device behaves as depicted in Figure 1a. 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 1a. 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 it already has, a mechanism known as client-side deduplication [14]. If any hash is unknown to servers, the device executes several storage operations directly with the storage servers (see *store* requests).[3] 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 changes performed in namespaces. For instance, when a partner in a shared folder performs changes, notifications 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 1b. 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. Every message sent to 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 s if no changes in 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 changes 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 in Figure 1a.

By observing traffic to notification servers, it is possible to identify when each namespace is changed. We explore these messages to trace the evolution of namespaces of Dropbox clients and reconstruct client behavior.

3. Datasets and Methodology

3.1. Datasets

We rely on Tstat [15] to perform passive measurements and collect data related to Dropbox from different networks. 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

[3] Dropbox also deploys a protocol for synchronizing devices in Local Area Networks, called `LAN Sync`. Devices in a LAN can use it to exchange files without retrieving content from the cloud. Since such traffic does not travel outside the LAN, we ignore the effects of LAN Sync.

Table 1: Overview of our datasets.

Name	Devices	Upload (TB)	Download (TB)	Namespaces	Changes[1]	Period
Campus-1	12,587	1.7	3.8	10,609	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	27,428	8.6	15.3	61,418	6,161,800	–

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

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 [16].

We adopt a methodology similar to the one in [1] 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. For example, 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 changes and heartbeat messages.

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 (see Section 2.2). For each event, we gather: (i) the timestamp of the event; (ii) the device ID related to the event; (iii) the user ID related to the event; (iv) each namespace ID registered in the device; and (v) the latest journal ID of each namespace.[4]

We have collected traffic in 4 different networks, including University campuses and ISP networks. Table 1 summarizes our datasets, presenting the number of Dropbox devices, the Dropbox traffic volume, the number of unique namespaces, the number of times the journal ID of a namespace has changed, and the data capture period.

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, whereas Campus-1 includes also WiFi networks.

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.

[4] Our data does not include any information that could reveal user identities, or information about the content users store in Dropbox. IP addresses are anonymized directly in the probes, whereas all Dropbox IDs are numeric keys that offer no hints about users' identities.

In total, we have observed more than 20 TB of data exchanges with Dropbox servers, 27 k unique Dropbox devices and 61 k unique namespaces during approximately 1 year of data collections. 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, aiming to foster independent validations of our work and new researches – e.g., to extend our model to other types of terminals and clients. We invite interested readers to contact the authors to access the datasets.

3.2. Data Preparation

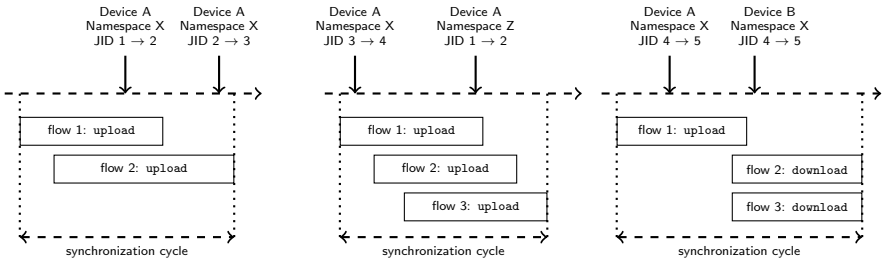
Our data capture resulted in two distinct and independent types of datasets for each analyzed network: (i) flow information that includes the data volume exchanged by clients with Dropbox storage servers; (ii) heartbeat messages with metadata related to the Dropbox namespaces. However, these two data sources have no common keys to provide an association between notifications of changes in namespaces and storage flows (i.e., data volume). Even more, we observe no one-to-one correspondence between notifications and flows. Indeed, files of different transactions might be exchanged in a single TCP flow, whereas a single transaction might be split across several TCP flows [3].

Our goal is to model and characterize the behavior of devices connected to Dropbox. As we will show later, our model requires knowledge about (i) how frequently namespaces are changed, and (ii) how much data is uploaded and/or downloaded to/from Dropbox servers. To extract this information, we develop a method to estimate the uploaded/downloaded volume that corresponds to each notification of changes in namespaces. 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 devices in Dropbox sessions, 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 uploading and/or downloading content. Considering the Dropbox protocols, notifications of changes 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 s, 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 [3]. Therefore, we label flows simply by



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

Figure 2: Examples of the most common cases processed by our method to label notifications and to estimate the transferred volumes (NS stands for namespace).

inspecting their downstream and upstream volumes. Then, we inspect metadata in notifications (e.g., device and namespace IDs) to decide how to distribute upload and download volumes among notifications.

Figure 2 provides the most common cases we notice while assigning volumes to notifications. Figure 2a presents a scenario where a Device A performs several uploads 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 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 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 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 changes 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 associate 77% (60%) of the total volume of upload (download) with notifications in Campus-1, and 54% (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 NATs in each network and the packet loss rate in the respective probes.

4. Modeling the Client Behavior

4.1. Assumptions

We target two aspects to build a model that captures 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 devices, as well as the dynamics behind modifications 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., modifications 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 in Section 2. 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.

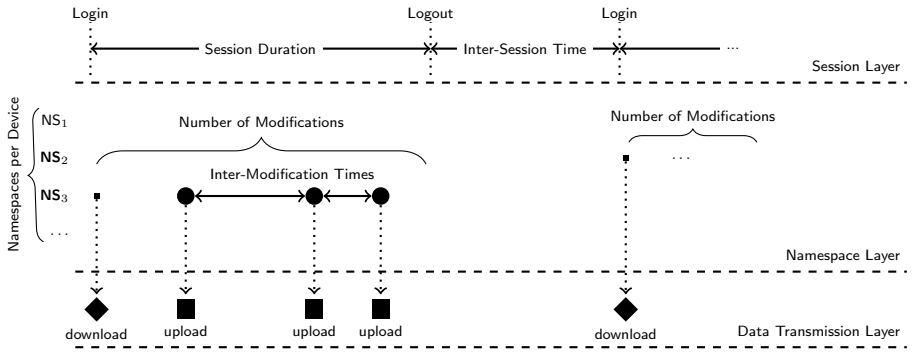


Figure 3: Model of the Dropbox client behavior.

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 in [17].

4.2. 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.

4.2.1. Modeling Device Behavior

Figure 3 depicts how our model represents the working of a single Dropbox device. It captures three fundamental aspects to 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 Dropbox 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

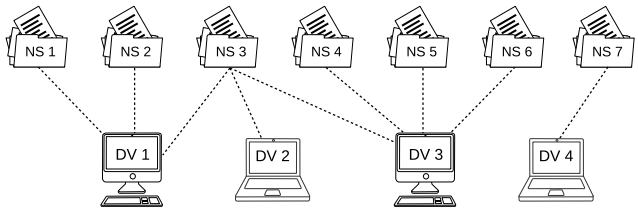


Figure 4: Example of the content propagation network (DV stands for device).

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 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.

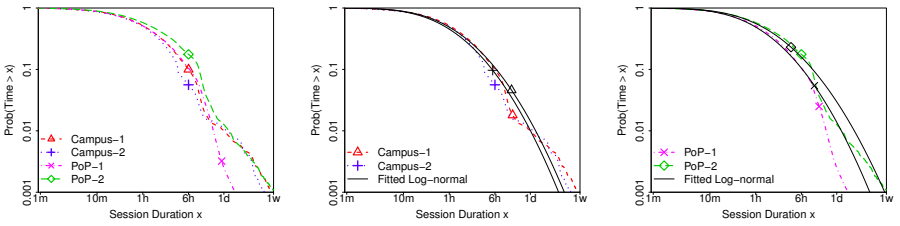
Finally, at the *data transmission layer*, we characterize the size of the workloads locally produced at the device, which are sent to servers. The *upload volume* is the single component in this layer.

4.2.2. Modeling the Content Propagation Network

Modifications in namespaces (i.e., uploads) trigger 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 4, 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 3, 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.



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

Figure 5: 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.

4.3. Characterization

We characterize each component presenting the statistical distribution that best fits our measurements. We focus only on the working days in our data, removing all activity observed during weekends or holidays, since we do not model such seasonality.

We summarize results next, while our methodology to obtain the best-fitted distributions and their respective parameters are described in Appendix A.

4.3.1. Session Layer

Session Duration:

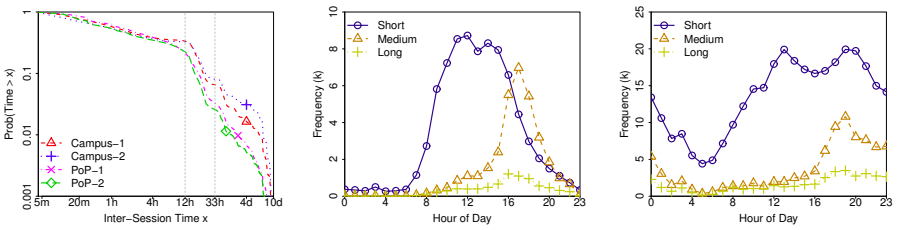
Figure 5 describes sessions duration by plotting the empirical Complementary Cumulative Distribution Functions (CCDFs) for each dataset. Figure 5a contrasts the datasets, whereas Figures 5b and 5c show separate plots for campuses and PoPs, including lines for the best-fitted distributions.

Focusing on Figure 5a, 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 [18].

Inter-Session Time:

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



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

Figure 6: 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.

to the routine of users in working days in different environments, which includes short breaks during the day and long breaks overnight or across days. Moreover, we clearly notice the existence of classes of users, as previously suggested in [1]. Only a small portion of devices (20–40%) connects daily to Dropbox, while the remaining ones are seen only occasionally.

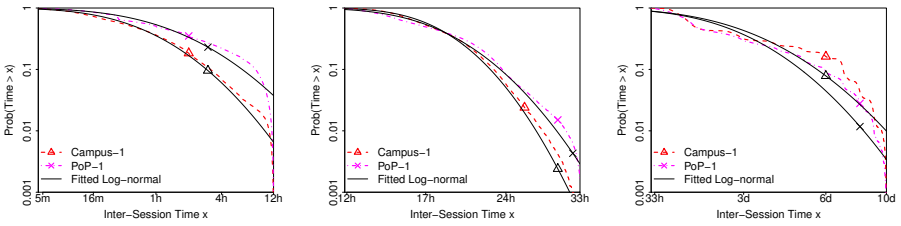
These patterns are illustrated in Figure 6a, which depicts CCDFs of inter-session times in the four datasets (note the logarithmic scales). We see a majority of *short inter-sessions* (67–76%) that are shorter than 12 hours, compatible with intraday breaks. A significant percentage of *medium inter-sessions* (20–28%) lasts between 12 and 33 hours. Those define primarily the overnight off-line intervals of devices that connect to Dropbox every day (i.e., frequent 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 occasional devices.[5]

Interestingly, the intraday routine in each environment determines when inter-session ranges appear in practice. Figures 6b and 6c 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 (frequent) or some days later (occasional).

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 generating workloads in Section 5, considering the existence of frequent and occasional devices and the probability of each range throughout the day.

Figure 7 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.

[5]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 7: Inter-session times per range and best-fitted Log-normal distributions.

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 occur in practice.

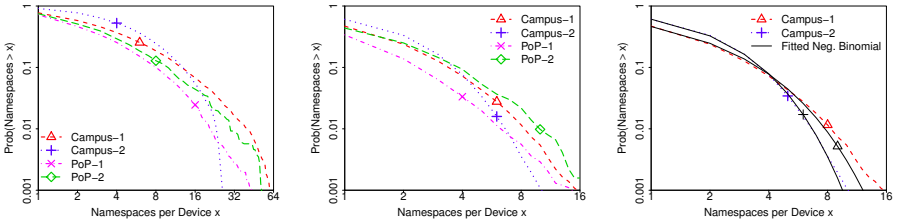
4.3.2. Namespace Layer

Namespaces per Device:

Figure 8a shows the CCDF of the number of namespaces per device. Results are in-line with findings in [1]. 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 8b, 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 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.

Since we are interested in the workload in short time intervals (i.e., weeks), we refrain from modeling 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 8b. Despite distinct patterns between campus and residential networks, Figure 8c shows that Negative Binomial distributions are the best fit in campus datasets. PoP datasets are omitted for the sake of brevity, but they provide the same conclusion.



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

Figure 8: 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.

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 restrict 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.

Number of Modifications:

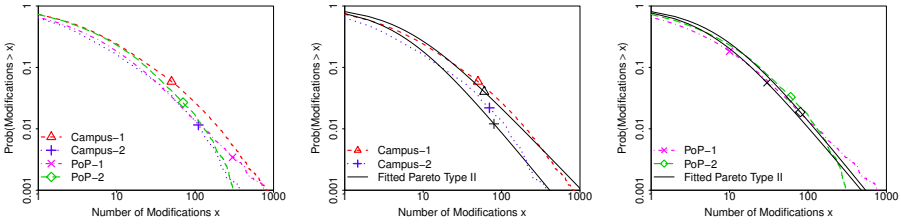
We characterized the 7–15% of sessions that have at least one modification in namespaces. Figure 9 shows CCDFs of the number of modifications per session.

In general, users make few modifications in namespaces per session. Figure 9a 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 9a, 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 9b and 9c that Pareto Type II distributions fit our data satisfactorily, despite differences in parameters per dataset.

Inter-Modification Time:

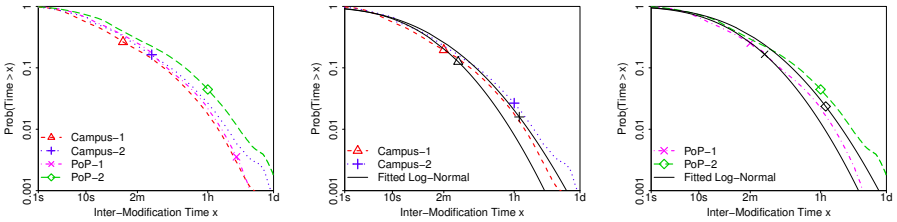
Figure 10 shows CCDFs for inter-modification times, which are computed considering sessions with at least two modifications.

Figure 10a shows that short time intervals are frequent between modifications. For instance, at least 60% of the inter-modification times are shorter than



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

Figure 9: Number of modifications. Pareto Type II distributions are the best candidate.



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

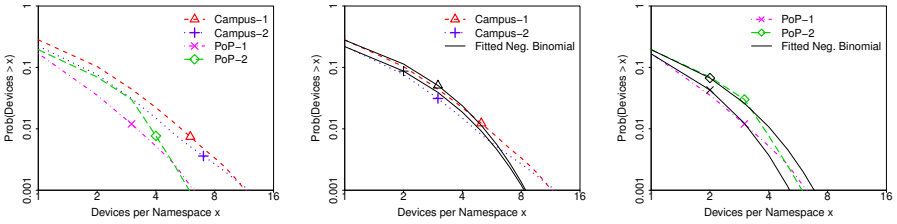
Figure 10: Inter-modification time. Log-normal distributions are the best candidate.

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 10b and 10c, 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. 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 number of devices per namespace.

Figure 11a 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 –



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

Figure 11: Devices per namespace (logarithmic scales). Most namespaces are accessed by a single device in the network. Negative Binomial distributions best fit our data.

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. Moreover, the probability of observing a namespace with a very high number of devices in campuses is slightly greater than in residential networks (see Figure 11a). This is somehow expected, since we have already shown that devices at campuses have more namespaces than devices at home (recall Figure 8a).

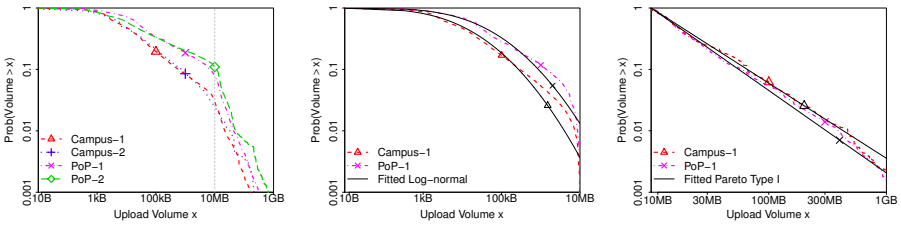
To understand these differences further, we analyze the number of devices per namespace considering which users own the devices – i.e., disregarding devices of the same user. We observe about 17% of namespaces are shared by multiple local users in campuses, whereas that percentage is 7–8% in PoPs. Thus, 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 11b and Figure 11c.

4.3.3. Data Transmission Layer

Finally, we study the volume of upload workloads. We characterize the variable using only the subset of notifications for which we have very high confidence about the upload volume – i.e., those notifications that occur while a single device is performing uploads to a single namespace (see Figure 2a in Section 3). We expect to select a random subset of notifications, thus providing a good estimation for the size of each workload.

Figure 12a shows CCDFs of upload sizes. Interestingly, uploads are larger in PoPs than in campuses. We conjecture that this happens because users at



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

Figure 12: 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.

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 interactive work.

The figure clearly shows knees in the distributions at around 10 MB 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 can be explained by two factors. First, the Dropbox client aggregates multiple small files in batches, committing transactions at arbitrary batch sizes to start content propagation while processing other files [1]. This strategy may bias workload sizes towards particular thresholds (e.g., 10 MB). Second, it has already been shown that file sizes in general are not modeled well by a single distribution model, with the body and the tail of the empirical distribution following distinct processes [19].

This suggests to break the measured data into two ranges, determining the best fit for each range independently. Figure 12b shows the empirical distributions and best fits for the measured volumes up to 10 MB, whereas Figure 12c 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 12b). On the other hand, uploads with volume greater than 10 MB are best fitted by the Pareto Type I distribution (see Figure 12c). Interestingly, these results are in agreement with the generative model for file sizes proposed in [19].

4.4. 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 [2]. 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 4.1), 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, a simple tuning of the propagation logic would adapt the model to other protocols. Readers can refer to [2, 3] for a comparison of cloud storage designs and for data about usage patterns in different types of terminals.

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, at a good extent, the generic functioning of the Dropbox client.

5. CloudGen – Cloud Storage Traffic Generator

5.1. Overview

We implement CloudGen based on our hierarchical model for the Dropbox client. CloudGen receives as input parameters for the set of distributions characterizing the model, as well as execution parameters, such as the synthetic workload duration and the target number of devices (d). The latter can be freely set 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, 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 servers in a cloud storage data-center face in different deployments.

CloudGen starts by setting up the content propagation network, i.e., the structure that represents how namespaces are shared among devices (see Figure 4). This process follows two steps. First, CloudGen 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). 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, CloudGen associates devices to namespaces. For that, CloudGen includes a method to generate random networks based on [20]: (i) CloudGen first

picks one namespace for each device to make sure all devices have at least one namespace; 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).

CloudGen then generates workloads following a classic discrete-event simulation approach. First, it creates a sequence of events to mark session logins and logouts of each device. The time of each event is determined by sampling the sessions duration and the inter-session time distributions, until the next login of all devices falls outside the synthetic trace duration.

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. However, CloudGen separates devices into two groups (frequent and occasional), and enforces only medium inter-sessions (frequent devices) or long inter-sessions (occasional devices) in night periods. As such, CloudGen mimics the behavior shown in Figure 6, which results in a sharp decrease in the number of new sessions at night, and multiple-day breaks for occasional 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 session, CloudGen decides whether new events are required by following our namespace selection strategy: 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. It does so by following a method found on [21] to match 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.

CloudGen assigns the upload volume to each modification and, eventually, schedules download events using the content propagation network and the information about login and logout times of peer devices. Modifications that happen in a shared namespaces are then propagated to all peer devices. CloudGen is available as free software at `http://cloudgen.sourceforge.net`.

5.2. 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 Dropbox.

Direct Metrics:

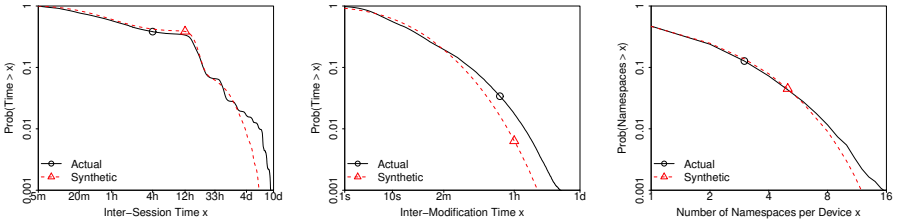
We recompute all components of our model characterized in Section 4.3, now using synthetic traces, to validate CloudGen implementation. For the sake of brevity, we show results only for Campus-1 in this first test, since other datasets lead to the same conclusions. We also omit plots for sessions duration, 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 13 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 13a. 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 split devices as frequent and occasional, as well as to select inter-session ranges, works well in practice.

Figure 13b 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 13c 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



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

Figure 13: Comparison of model components in actual and synthetic traces in Campus-1.

characterized for the variable. In fact, our method to create content propagation networks approximates well both distributions that characterize the Dropbox network, even when small networks are created (e.g., hundreds of devices).

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

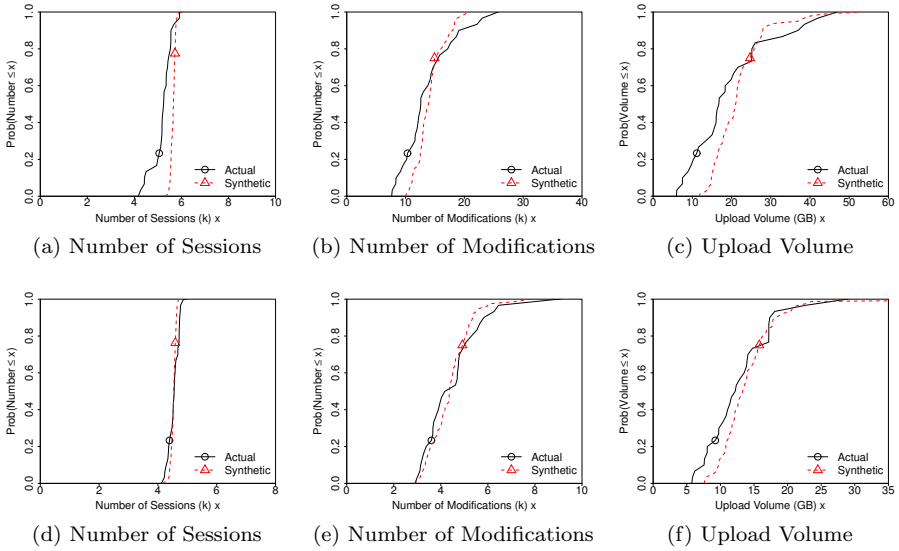
Indirect Metrics:

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 14 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 for brevity since they show similar patterns.

Figures 14a and 14d show that the number of sessions per day in synthetic traces is close to actual values. Focusing on Figure 14d, 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 14b and 14e 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%),



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

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

Figure 14: 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.

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 14b, once more, thanks to the heterogeneous users' routine in the environment throughout the week.

Figures 14c and 14f 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 administrators to easily provision resources for cloud storage services.

6. Future Scenarios for Cloud Storage Usage

6.1. 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:

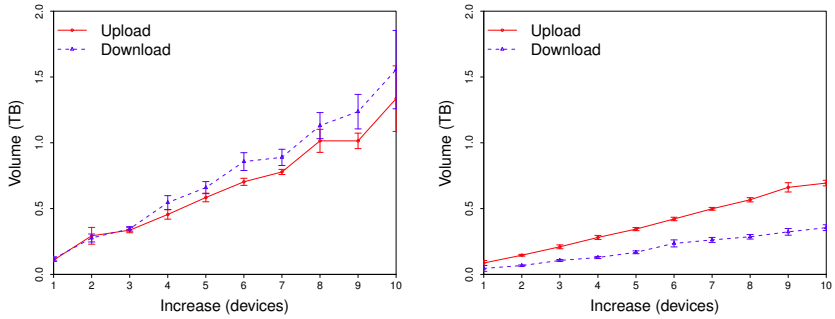
1. *Larger Population*: This scenario studies how Dropbox user population growth affects network traffic. We increase the number of devices, evaluating how the network traffic changes with up to 10-times more devices than what is observed nowadays in edge networks.
2. *Larger Sharing*: We simulate a 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: devices never leave the simulated environment, all uploaded content is propagated in the local network and external devices produce no workload.

6.2. Traffic Analysis

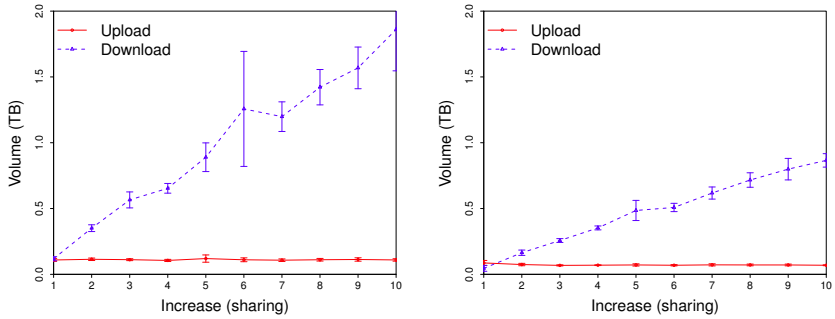
Figure 15 presents mean upload and download volumes as number of devices is varied (larger population scenario). Error bars mark 95% confidence intervals.

We can see that upload and download volumes grow roughly linearly with the number of devices in the network – e.g., note how there are 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 4.3.2). 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.



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

Figure 15: Mean upload and download volumes in larger population scenarios.

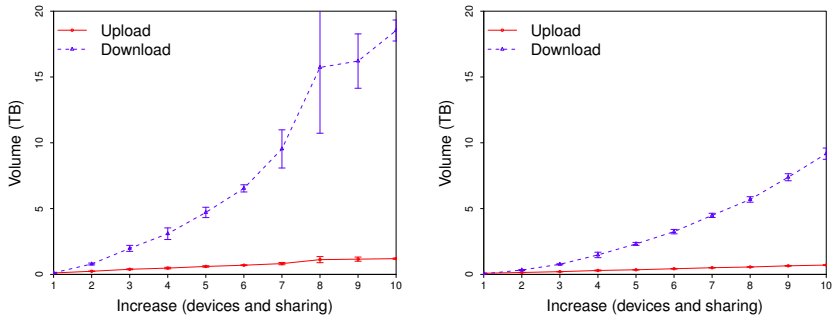


(a) Campus-1 (b) PoP-1

Figure 16: Mean upload and download volumes in larger sharing scenarios.

Figure 16 presents mean traffic volumes for the larger sharing scenario – i.e., when we increase only the mean number of devices per namespace. Naturally, varying the number of devices per namespace does not affect the upload. Download volume, on the other hand, is strongly affected by the parameter. We observe that the download volume more than double 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.

Finally, Figure 17 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 15 as expected. More interesting, note how this combination of more devices with more sharing



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

Figure 17: Mean upload and download volumes in larger population and sharing scenarios.

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.

7. Related Work

This work studies the *underlying client processes* in cloud storage services. In particular, we extend our previous efforts to model Dropbox client behavior [10, 11]. Unlike [10], our new model explicitly represents data modifications and sharing. It captures content modifications that lead to data exchanges in the network, and show how content is propagated in the Dropbox network of shared folders and among the several devices of a user. We also build on top of the proposed model now, by developing and validating a new workload generator, namely CloudGen.

In [11] we introduced a preliminary implementation of CloudGen, which was able to simulate content propagation, given a number of shared folders. Here, we have the complete description of client behavior in CloudGen. CloudGen is also validated using real traces, and we demonstrate CloudGen functionality through an evaluation of the expected growth of cloud storage traffic in campus and ISP networks.

Next, we summarize other related work, discussing both efforts that address aspects of cloud storage services (Section 7.1) and efforts to model and evaluate workloads of Internet services in general (Section 7.2).

7.1. Cloud Storage Services

Drago et al. [1] are the first to present an in-depth characterization of typical usage and performance bottlenecks of Dropbox from passive measurements. We partly rely on their methodology to collect and process data about Dropbox. Unlike [1], and motivated by our model for the Dropbox client behavior, we characterize Dropbox workload taking into account users' sessions and content sharing. Moreover, we leverage the characterization to build a workload generator and to evaluate possible future scenarios for Dropbox usage.

Various studies focus on benchmarking cloud storage using active experiments. Hu et al. [22] study the backup performance of four services. Wang et al. [23] analyze Dropbox bottlenecks, while the public APIs of three providers are compared in [9]. Authors of [3] introduce a framework to benchmark cloud storage. Relying on an automated testing methodology, they compare design choices and implications by benchmarking 11 services. Li et al. [8] follow a similar methodology and compare cloud storage performance considering also different types of client terminals. None of them characterize or model the behavior of actual cloud storage users, as we do in this work.

Some works propose protocols and architectures to improve performance or to decrease costs of cloud storage services. In [24], a framework that collects semantic information from files (e.g., file type) is proposed to improve performance of the services. Zhang et al. [5] propose a mechanism to ensure consistency between the local file system and remote repositories. Authors of [4] propose QuickSync, a system that optimizes synchronization by combining novel techniques, such as network-aware chunking. Bo et al. [25] propose a data distribution approach to improve cloud storage availability that exploits workload characteristics of providers. Our work contributes to such efforts with a workload generator that can be used to evaluate new proposals in realistic scenarios.

Other works analyze cloud storage focusing on economics [26], security [14], quality of experience [6, 7] or privacy [27], and thus are orthogonal to our work.

7.2. Models and Workload Generators for Internet Services

To the best of our knowledge, we are the first to propose a model and to implement and validate a system to generate synthetic workloads for cloud storage services, such as Dropbox.

Several models and workload generators have been proposed to create web request streams [21, 28–30]. Two of them model users' behavior and its impacts on network traffic and, as such, are closer to our work. SURGE [21] is a workload generator that employs the concept of user equivalents to generate traffic for web servers. SWAT [30] considers user sessions and the dependence of content requests within sessions. We borrow general ideas from SURGE, such as to match distributions of sessions duration and number of modifications. Neither SURGE nor SWAT, however, include mechanisms to represent the dynamics of content generation and propagation as in cloud storage services. Hence, both approaches are insufficient to cover the particularities of such services.

Other workload generators have been proposed targeting media streaming services. Tools like GISMO [31] and MediSyn [32] generate synthetic workloads, including client sessions and arrival patterns. Abad et al. [33] propose a general synthetic generator to create object request streams, but without including aspects fundamental to cloud storage, such as workload volumes and sessions duration. Closer to our work, Veloso et al. [34] and Borges et al. [35] propose hierarchical models for the behavior of users in live streaming systems. Those models incorporate users' sessions, ON/OFF times that represent active and idle periods, peering behavior and data transfers. We adopt a similar hierarchical approach to model Dropbox, but focusing on a completely different type of service, for which both model components and parameterization differ.

Finally, our work contributes to efforts in studying cloud workloads in general. Some works characterize and model workloads of the cloud computing environment [36–38], such as to estimate CPU and memory utilization. Delimitrou et al. [39] propose a workload generator that recreates I/O loads in data centers, based on inter-arrival times of storage access seen in application traces. Similarly, tools like CloudSim [40] aim at simulating workloads in cloud computing. CloudGen can help such efforts in reproducing workloads of cloud applications, since it models the behavior of end-users and consequent network traffic for one of the largest cloud services currently deployed (i.e., Dropbox).

8. Conclusion and Future Work

This paper investigated the workload of cloud storage services. We proposed a model to describe the Dropbox workloads, and parameterized it using data from four different networks. We then used our model to implement CloudGen, the first workload generator that simulates both users' behavior and the network traffic of cloud storage. We validated our workload generator, and illustrated its applicability by exploring the traffic volume that can be expected in future usage scenarios. We offer CloudGen to the community as free software, as a valuable tool to simulate the behavior of cloud storage clients.

The analyses of Dropbox workloads from real data showed that our model for the Dropbox functioning is robust. Indeed, we found close agreement for all components of the model in four distinct datasets.

We envision several directions for future work. Regarding our workload generator, we plan to extend it to other cloud storage protocols and terminals. For instance, we plan to add components to control the level of replication of updates, modeling cases where only part of peer-devices receives them. We also intend to model seasonality to simulate long-term dynamics of workloads in cloud storage as well as to plug CloudGen in well-known network simulators, then testing the impacts of network conditions on workloads.

Moreover, we will investigate how individual user's profile influences the workload, and factor that into CloudGen, to allow the simulation of populations with particular characteristics. Finally, regarding applications of our model, we will leverage CloudGen to evaluate trade-offs (e.g., return on investments) of alternative architectures for cloud storage services.

9. Acknowledgments

This research is partially funded by the first author' individual grants from CNPq, by the FAPEMIG-PRONEX-MASWeb project – Models, Algorithms and Systems for the Web, process number APQ-01400-14, by the National Institute of Science and Technology for the Web (INWEB), CNPq and FAPEMIG, as well as by the European Union under the FP7 Grant N. 318627 (Integrated Project "mPlane").

References

[1] I. Drago, M. Mellia, M. M. Munafò, A. Sperotto, R. Sadre, A. Pras, Inside Dropbox: Understanding Personal Cloud Storage Services, in: Proc. of the ACM Internet Measurement Conference, 481–494, 2012.

[2] E. Bocchi, I. Drago, M. Mellia, Personal Cloud Storage: Usage, Performance and Impact of Terminals, in: Proc. of the IEEE CloudNet, 106–111, 2015.

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

[4] Y. Cui, Z. Lai, X. Wang, N. Dai, C. Miao, QuickSync: Improving Synchronization Efficiency for Mobile Cloud Storage Services, in: Proc. of the ACM International Conference on Mobile Computing and Networking, 592–603, 2015.

[5] Y. Zhang, C. Dragga, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, ViewBox: Integrating Local File Systems with Cloud Storage Services, in: Proc. of the USENIX Conf. on File and Storage Technologies, 119–132, 2014.

[6] P. Amrehn, K. Vandenbroucke, T. Hossfeld, K. D. Moor, M. Hirth, R. Schatz, P. Casas, Need for Speed? On Quality of Experience for Cloud-based File Storage Services, in: Proc. of the International Workshop on Perceptual Quality of Systems, 184–190, 2013.

[7] P. Casas, R. Schatz, Quality of Experience in Cloud Services: Survey and Measurements, Computer Networks 68 (1) (2014) 149–165.

[8] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai, Z.-L. Zhang, Towards Network-Level Efficiency for Cloud Storage Services, in: Proc. of the ACM Internet Measurement Conference, 115–128, 2014.

[9] R. Gracia-Tinedo, M. S. Artigas, A. Moreno-Martinez, C. Cotes, P. G. Lopez, Actively Measuring Personal Cloud Storage, in: Proc. of the IEEE International Conference on Cloud Computing, 301–308, 2013.

[10] G. Gonçalves, I. Drago, A. P. C. da Silva, A. B. Vieira, J. M. de Almeida, Modeling the Dropbox Client Behavior, in: Proc. of the IEEE International Conference on Communications, 1332–1337, 2014.

[11] G. Gonçalves, I. Drago, A. B. Vieira, A. P. C. da Silva, J. M. de Almeida, Analyzing the Impact of Dropbox Content Sharing on an Academic Network, in: Proc. of the Brazilian Symposium on Networks and Distributed Systems (in Portuguese), 100–109, 2015.

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

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

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

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

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

[17] M. Yuksel, B. Sikdar, K. S. Vastola, B. Szymanski, 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, 93–98, 2000.

[18] D. Stutzbach, R. Rejaie, Understanding Churn in Peer-to-Peer Networks, in: Proc. of the ACM Internet Measurement Conference, 189–202, 2006.

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

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

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

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

[23] H. Wang, R. Shea, F. Wang, J. Liu, On the Impact of Virtualization on Dropbox-Like Cloud File Storage/Synchronization Services, in: Proc. of the IEEE International Workshop on Quality of Service, 11:1–11:9, 2012.

[24] F. Chen, M. P. Mesnier, S. Hahn, Client-Aware Cloud Storage, in: Proc. of the IEEE Symposium on Mass Storage Systems and Technologies, 2014.

[25] H. J. Bo Mao, Suzhen Wu, Exploiting Workload Characteristics and Service Diversity to Improve the Availability of Cloud Storage Systems, IEEE Transactions on Parallel and Distributed Systems PP (99) (2015) 1–1.

[26] M. Naldi, L. Mastroeni, Cloud Storage Pricing: A Comparison of Current Practices, in: Proc. of the ACM International Workshop on HotTopiCS, 27–34, 2013.

[27] S. Liu, X. Huang, H. Fu, G. Yang, 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, 327–334, 2013.

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

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

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

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

[32] W. Tang, Y. Fu, L. Cherkasova, A. Vahdat, 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, 12–21, 2003.

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

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

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

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

[37] Q. Zhang, J. L. Hellerstein, R. Boutaba, Characterizing Task Usage Shapes in Google's Compute Clusters, in: Proc. of the Large Scale Distributed Systems and Middleware Workshop, 2011.

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

[39] C. Delimitrou, S. Sankar, K. Vaid, C. Kozyrakis, Accurate Modeling and Generation of Storage I/O for Datacenter Workloads, in: Proc. of the EXERT, 2011.

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

[41] W. N. Venables, B. D. Ripley, Modern Applied Statistics with S, Springer, New York, USA, 2002.

[42] R. B. D'Agostino, M. A. Stephens, Goodness-of-Fit Techniques, Marcel Dekker, New York, USA, 1986.

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

Appendix A. Best-Fitted Distributions

We characterize the parameters of our model by searching for the distributions that best-fit our measurements. For each component of the model, we use the Maximum-Likelihood Estimation (MLE) method [41] to estimate best-fitted curves (i.e., distribution parameters) from the measured data. We consider the following distributions as candidates for best fit: Normal, Log-Normal, Exponential, Cauchy, Gamma, Logistic, Beta, Uniform, Weibull, Pareto (continuous variables); and Poisson, Binomial, Negative Binomial, Geometric and Hypergeometric (discrete variables).

The final decision of what candidate distribution best fits our data is taken by comparing the Kolmogorov-Smirnov statistic [42] (for continuous distributions) and the least square errors (LSE) [43] (for discrete distributions) of the fitted curves. We then visually analyze the obtained best-fitted curves both at the body and at the tail of the distributions to support our decisions.

Table A.2 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.2: 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	$\alpha = 1.312$ $\beta = 10,000$