University of Crete
Computer Science Department


# Rise of the Planet of the Apps: A Systematic Study of the Mobile App Ecosystem

Thanasis Petsas

Master's Thesis

November 2012
Heraklion, Greece

University of Crete
Computer Science Department

# Rise of the Planet of the Apps: A Systematic Study of the Mobile App Ecosystem

Thesis submitted by

Thanasis Petsas

in partial fulfilment of the requirements for the
Master of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Thanasis Petsas

Committee approvals: _____
Evangelos P. Markatos
Professor, Thesis Supervisor

_____
Sotiris Ioannidis
Principal Researcher at FORTH-ICS

_____
Maria Papadopouli
Assistant Professor

Departmental approval: _____
Angelos Bilas
Professor, Chairman of Graduate Studies

Heraklion, November 2012

# Abstract

Mobile application stores have recently gained significant popularity due to the evolution of the smartphone applications' (apps) market and the large increase of smartphone users. Apart from official marketplaces such as Google's *Android Market* or Apple's *App Store*, there is a plethora of alternative app stores with a large number of both applications and users.

In this thesis, we perform a systematic study on four third-party Android marketplaces, in order to improve our understanding on several aspects of this rapidly evolving ecosystem. More specifically, we study (i) how apps are being produced and common strategies observed among app developers, (ii) the app popularity pattern, (iii) user download patterns and how can be affected by recommendation systems, as well as (iv) how app pricing affects apps popularity and the developers' income. Furthermore, we compare our findings with similar studies on other fields, such as the world wide web (WWW), peer-to-peer file sharing (P2P) systems, and user-generated content (UGC) sites.

The data of our analysis are gathered by systematically crawling four popular Android appstores in a daily basis, for several months. Our results indicate that mobile marketplaces are comprised of a small number of very popular applications that absorb the majority of downloads, confirming the existence of the *Pareto Principle*. Moreover, the distribution of app popularity follows a Zipf-like behavior with some deviations, very similar with the behavior observed in *P2P* and *UGC* workloads. We verify that these deviations from the Zipf distribution are caused in part due to the *"fetch-at-most-once"* user behavior, that other studies have already demonstrated, as well as by the existence of another phenomenon we call the *clustering effect*. According to the clustering effect, apps are grouped into clusters, which may be a result of recommendation systems, user communities or other grouping forces, and users tend to download apps from the same clusters with higher probability. We verify our app clustering hypothesis using a new metric called "user temporal affinity" to app categories, and we measure it using a dataset with user comments, which implies user downloads. The results show that indeed users have a strong affinity to app categories. Then, we propose a novel model based on the clustering effect and fetch-at-most-once property, and we evaluate our model with a simulation-based study comparing with the observed app downloads. We find that our model approximates very well the actual distribution of app downloads.

Supervisor: Professor Evangelos Markatos

# Περίληψη

Τα appstores έχουν αποκτήσει μεγάλη δημοτικότητα πρόσφατα λόγω της ανάπτυξης που παρατηρείται στην αγορά εφαρμογών για έξυπνα τηλέφωνα (smartphones) και λόγω της ραγδαίας αύξησης των χρηστών που χρησιμοποιούν αυτές τις συσκευές. Εκτός από τα επίσημα appstores, όπως το Android Market της Google και το App Store της Apple, υπάρχει μια πληθώρα από εναλλακτικά appstores με έναν μεγάλο αριθμό από εφαρμογές και χρήστες.

Σε αυτή την εργασία, πραγματοποιούμε μια συστηματική μελέτη σε τέσσερα εναλλακτικά appstores με εφαρμογές για Android, με σκοπό να κατανοήσουμε καλύτερα τις διάφορες πτυχές αυτού του ταχέως εξελισσόμενου «οικοσυστήματος». Πιο συγκεκριμένα, μελετάμε (1) πώς παράγονται οι διάφορες εφαρμογές, και ποιες είναι οι κοινές στρατηγικές που παρατηρούνται ανάμεσα στους προγραμματιστές αυτών των εφαρμογών (app developers), (2) την δημοτικότητα αυτών των εφαρμογών, (3) τα πρότυπα χρήσης των appstores και πως αυτά επηρεάζονται από τα συστήματα συστάσεων (recommendation systems), καθώς επίσης (4) πως το κόστος των εφαρμογών επηρεάζει τη δημοτικότητά τους και τα έσοδα των developers. Επιπλέον, συγκρίνουμε τα ευρήματα μας με παρόμοιες μελέτες σε άλλους τομείς, όπως το Παγκόσμιο Ιστό (WWW), τα Ομότιμα Συστήματα (P2P) διαμοιρασμού αρχείων και τα user-generated content (UGC) συστήματα.

Τα δεδομένα που χρησιμοποιούμε στην ανάλυσή μας συλλέχθηκαν κάνοντας συστηματικά crawl τέσσερα δημοφιλή Android appstores σε καθημερινή βάση για μερικούς μήνες. Τα αποτελέσματα μας υποδεικνύουν ότι τα appstores αποτελούνται από έναν μικρό αριθμό με πολύ δημοφιλή εφαρμογές που απορροφούν το μεγαλύτερο μέρος των downloads, επιβεβαιώνοντας την ύπαρξη της Αρχής του Pareto. Επίσης, βλέπουμε ότι η δημοτικότητα των εφαρμογών ακολουθεί κατανομή Zipf με κάποιες αποκλίσεις, πολύ παρόμοια με τη συμπεριφορά που παρατηρήθηκε σε P2P και UGC συστήματα. Επιβεβαιώνουμε ότι αυτές οι αποκλίσεις από την κατανομή Zipf, προκαλούνται εν μέρει από την συμπεριφορά των χρηστών που κατεβάζουν κάθε συγκεκριμένη εφαρμογή το πολύ μία φορά ("fetch-at-most-once"), που έχει αποδειχθεί ήδη από άλλες μελέτες, καθώς και από την ύπαρξη ενός άλλου φαινομένου, το οποίο ονομάζουμε «φαινόμενο κατηγοριοποίησης». Σύμφωνα με το φαινόμενο της κατηγοριοποίησης, οι εφαρμογές είναι χωρισμένες σε κατηγορίες, οι οποίες μπορεί να είναι αποτέλεσμα των συστημάτων συστάσεων, των διαφορετικών ενδιαφερόντων που έχουν οι χρήστες ή από άλλους τρόπους κατηγοριοποίησης των εφαρμογών, και οι χρήστες τείνουν να κατεβάζουν εφαρμογές από τις ίδιες ομάδες με μεγάλη πιθανότητα. Επαληθεύουμε την υπόθεσή μας περί κατηγοριοποίησης των εφαρ-

μογών χρησιμοποιώντας μια νέα μετρική που αποκαλούμε «χρονική συγγένεια του χρήστη» στις κατηγορίες των εφαρμογών, και κάνουμε μετρήσεις σε ένα σύνολο δεδομένων από σχόλια χρηστών, τα οποία υποδηλώνουν downloads εφαρμογών. Τα αποτελέσματα δείχνουν ότι πράγματι οι χρήστες έχουν μια ισχυρή «συγγένεια» προς τις κατηγορίες των εφαρμογών. Στη συνέχεια, προτείνουμε ένα μοντέλο που βασίζεται στο φαινόμενο της κατηγοριοποίησης και στην fetch-at-most-once ιδιότητα. Αξιολογούμε το μοντέλο μας με μία μελέτη βασισμένη σε προσομοίωση και συγκρίνουμε τα αποτελέσματα με τα πραγματικά δεδομένα που παρατηρήσαμε στα downloads των διαφόρων εφαρμογών. Βρίσκουμε ότι το μοντέλο μας προσεγγίζει πολύ καλά την πραγματική κατανομή των downloads των εφαρμογών.

Επόπτης: Καθηγητής Ευάγγελος Μαρκατος

The research outlined in this thesis has been conducted in the Distributed Computing Systems Laboratory (DCS), which is hosted in the Institute of Computer Science (ICS) of **Foundation for Research and Technology - Hellas (FORTH)**, N. Plastira 100 Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece.

# Acknowledgments

First of all, I would like to express my deepest gratitude to my supervisor Prof. Evangelos Markatos, for giving me the guidelines through my whole graduate studies. I am really pleased to cooperate with people of his mental and ethical values.

Special thanks to Spiros Antonatos, the person that helped me to take my first steps in the world of research and giving me the opportunity to work in very interesting projects.

I would like to thank Antonis Papadogiannakis for his invaluable help in this work and all the joyful moments we shared together gnuplotting. I would also like to thank Thomas Karagiannis (Microsoft Research) and Michalis Polychronakis (Columbia University), whose constructive comments and suggestions were of vital importance for the progress of this thesis. Daniel Song (Columbia University) implemented the prototypes of the Chinese Android Market crawlers, therefore I greatly appreciate his support.

I am also grateful to Dr. Sotiris Ioannidis for his advice and support.

Many thanks to all the rest (former and present) members of the DCS (Distributed Computing Systems) Lab at FORTH-ICS Demetris Antoniades, Iasonas Polakis, Elias Athanasopoulos, Giorgos Vasiliadis, Alexandros Kapravelos, Andreas Sfakianakis, Giorgos Kontaxis, Eleni Gessiou, Spiros Ligouras, Nikos Tsikoudis, Lazaros Koromilas, Panagiotis Papadopoulos, Antonis Papaioannou, Giorgos Chinis, Harris Papadakis, Christos Papachristos, Antonis Krithinakis, Manolis Stamatogiannakis and Melitini Christodoulaki as well as the non-DCS members Giorgos Saloustros and Panagiotis Garefalakis that contributed for a pleasant and productive environment all these years in the lab.

A big shout out to my friends Elias (Endov Lane) Panagiotopoulos, Manolis Stylianakakis, Aris Tzermias, Leonidas Groneberg, Giorgos Grigoreas, Giorgos Sykiotakis, Giannis Theoharis, Elias Kouroudis, Nikos Dritsos, Manolis Kounalas and many others that I do not mention by name, for their constant encouragement and support.

I would like to thank, from the very depths of my heart, my family, my brother Manolis, my sister Dimitra, my grandmother Dimitra, my father Giorgos and especially my mother Anna for their support, endless encouragement and patience throughout these years.

Finally, I never would have made it through without Anna Maria Wateroil. I thank her for all the love and understanding (kouu!).

A big thank to NIGHTSTALKER, Spiritual Beggars, Kyuss, Queens of the Stone Age, Down, Clutch, Baroness, Pelican, Explosions in the Sky, God is an Astronaut, If These Trees Could Talk, Giannis Aggelakas and Thanasis Papakonstantinou, whose music kept me company during the long nights I spent studying and working.

Logo of the title is designed by Lazaros Koromilas.

*Στους γονείς μου, Γιώργο και Άννα.*

*To my parents, George and Anna.*

# Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 The Emerging Growth of App Ecosystem

Mobile applications have started to become extremely popular as the adoption of smartphones and tablet computers appears to be faster than that of any other consumer technology in history. A recent survey [18] conducted by the mobile analytics platform *Flurry*[1] validates the above proposal by stating a comparison of adoption rates between smart devices and other recent technologies. The survey shows that smart devices is being adopted 10× faster than that of 80s PC revolution, 2× faster than that of Internet Boom and 3× faster than that of recent social network adoption. In the same survey, it is mentioned that Flurry counted over than 640 million active *iOS* and *Android* devices in July 2012. This swift espousal of mobile devices is also shown in several other articles, through graphical representations [3,27]. According to *IDC*[2], Android and iOS possess the 82% of total smartphone sales in the first quarter of 2012, where there was a 44% year-on-year increase of smartphone sales from 2011 to 2012 [15] and an increase of 85% from 2011 to 2012 [16]. Android appears to dominate the smartphone market share [9,19] with 1.3 million activations per day, as the Google CEO Eric Schmidt recently reported [7]. The ever-growing popularity of smartphones has attracted the interest of developers who try to increase their profits by developing applications. The main platforms that host and distribute these applications to the end-users are the mobile *application stores* or *marketplaces*. These infrastructures that constitute a new component in the Web

[1] `http://www.flurry.com/`
[2] `http://www.idc.com`

world recently have seen a rapid growth. For instance, the official Android Market (lately rebranded as *Google Play*), reached the 600000 apps milestone by the end of June 2012 [52]. In Figure 1.1, there is a more detailed graph that shows the growth of Android Market in terms of number of apps for a period of 3 years. We can see that in May 2011, the number of available apps in the Android Market was 200000, while one year later, in May 2012, the number has increased to 500000, that is an 150% annual growth rate. This implies that the number of apps has doubled in less than one year. Moreover, the average monthly growth rate is about 12.5% (25000 new apps per day) and the average daily growth rate is almost 0.4% (882 new apps per day), indicating a very rapid growth that appears to be a more general phenomenon in the world of mobile marketplaces. Along with official appstores

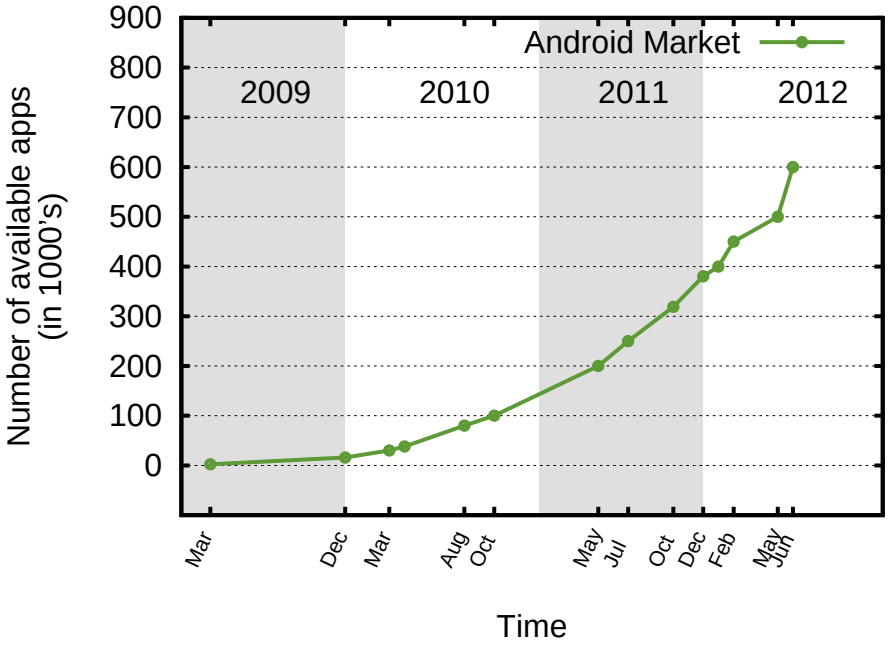



FIGURE 1.1: The growth of Android Market in terms of number of apps. Data sources: Wikipedia [52].

of mobile platform vendors (*e.g.* Google, Apple), there is a large number of other alternative third-party marketplaces that have become very popular as well, due to the rapid adoption of smartphones and other benefits that may offer to developers. For instance, SlideMe Android marketplace [25] offers a higher percentage of revenue to application developer than Google does in the official Android Market [4].

Despite the rapid growth that these online market structures appear to have, to the best of our knowledge, there are no other large-scale studies try to clarify the characteristics of the mobile app ecosystem. A system-

atic study aimed at understanding the characteristics and trends in the app marketplaces would be interesting for both application developers that use such infrastructures to distribute their software, as well as for researchers who wish to comprehend this new means of software distribution. In addition, an investigation into how mobile applications "produced", used and "consumed", as well as what are their popularity patterns over time, would be significant for designers of such kind of platforms, to understand any performance implications that may these standards pose and thus assist them in designing more efficient marketplaces in the future.

In this work, we perform a systematic study on four popular Android-based alternative marketplaces to understand the nature of the app ecosystem. To conduct our analysis, we develop and deploy a distributed non-intrusive crawling system that collects various information from four app stores in a daily basis. In particular, we explore how applications are being "produced" by developers and "consumed" by users, how their popularity changes over time and which are the main factors that may affect app popularity. Moreover, we examine how user patterns can be affected by user interests or recommendation systems and how they may affect the popularity of applications in a marketplace. Furthermore, we discuss how the pricing impacts the app popularity and developers' income. In addition, we compare our results with similar studies on other fields such as world wide web (WWW), peer-to-peer file sharing (P2P) systems, and user-generated video content (UGC).

## 1.2 Contributions

The highlights of this work can be summarised as follows:

- We demonstrate that, in general, the app marketplaces are dominated by a minimum number of popular apps that receive a very large number of download requests, while the majority of the applications are downloaded only a few times. In particular, we show that a more powerful form of *Pareto Principle* applies to all the monitored app markets.

- We show that in some monitored app store, there is a quite number of applications that have not received a single download during their life cycle. This phenomenon appears to occur due to poor design of the recommendation systems that these marketplaces have and due to the pricing of some portion of apps (*paid* apps).

- We analyze the popularity distributions of the apps in the monitored markets and we demonstrate that they exhibit a Zipf-like behaviour with truncated tails, which is differentiated from WWW traffic that follows a pure Zipf distribution. We argue that this deviation from

Zipf in part from *"fetch-at-most-once"* behavior that has been already observed in other areas as well (*e.g.*P2P, UCG), but also by the way that apps are grouped inside the market, trying to influence user preferences. We define the above *"grouping"* phenomenon as "clustering effect" and we believe that is a result of recommendation systems or other grouping forces (*i.e.*user-formed societies created by users' comments or ratings). We verify our app clustering hypothesis introducing a new metric called *"user temporal affinity"* to app categories, and we measure it using a dataset of user comments, which implies user downloads.

- We propose a novel model of appstore usage based on both clustering effect and fetch-at-most-once properties, and we evaluate our model with a simulation-based study comparing its results with the actual applications' downloads. We find that our model approximates very well the actual distribution of app downloads.

- We present a detailed study on the role of pricing in smartphone applications. We show how pricing affect the popularity of applications and we give insight on developers' income and their common strategies.

## 1.3 Thesis Outline

The remainder of this thesis is structured as follows. In Chapter 2 we provide information about our data collection, the crawling strategies we follow, the main challenges we are confronted with during the implementation of our crawling system, as well as the solutions we used to address these issues. Chapter 3 presents the first measurements on our data, where we attempt to understand how the marketplaces grow through time, in terms of number of available apps and total number of downloads. In Chapter 4, we study the popularity of mobile applications in terms of number of downloads and we present its main characteristics, along with various factors that may affect it. In Chapter 5, we present our *user temporal affinity* metric, where we explore whether users tend to stay within a single category, when they download apps, rather than switching to another one. Chapter 6 presents a novel model of appstore usage based on both fetch-at-most-once and clustering effect, as well as our results through simulations which validate our clustering effect hypothesis. In Chapter 8, we present similar studies to ours one, while in Chapter 9, we list some ideas for future work. Finally, Chapter 10 summarizes and concludes the thesis.

# 2
# Data Collection

This chapter introduces our dataset and describes the collection strategy we used to harvest our data. Moreover, we provide information about the challenges we encountered during the collecting process along with the solutions we used to overcome these problems. Finally, we discuss about the collected data used in our experiments. In general, we study the *smartphone app ecosystem* by looking at four popular alternative third-party Android marketplaces: SlideMe [25], 1Mobile [22], AppChina [24] and Anzhi [23].

## 2.1 The Monitored Appstores

In order to study the mobile app ecosystem we collected information from various app marketplaces. We were not willing to use the official Android Market (Google Play) for this purpose since the Google *Terms of Service* (ToS) do not allow users to access Google Play through any type of automated means (including use of scripts, crawlers *etc.*) without Google's consent [17]. Moreover, official Android Market, as well as other marketplaces (*e.g.*Amazon Appstore [5], AppBrain [13], AndroLib [12] *etc.*), does not provide precise details of the applications it is hosting such as the exact numbers of downloads or installations of an app, but instead ranges. We chose not to use such kind of appstores to collect our data since the results of our analysis would not be that accurate. The four popular third-party Android marketplaces we selected to monitor for our analysis are listed below along with some descriptive information:

- **SlideMe [25].** This is one of the oldest alternative Android marketplaces. It was founded in 2008, and contains over than twenty thousands of free and paid apps [1].

- **1Mobile [22].** One of the largest third-party Android appstores (the largest of our monitored ones) with an app population exceeding the 130000 apps.

- **AppChina [24].** A very popular alternative Android appstore in China with over than 60000 of apps.

- **Anzhi [23].** Yet another popular Chinese Android Market that often can be found pre-installed on *HTC*[1] smartphones in China [2].

All these marketplaces, apart from the website they maintain, which users can browse to find, download or buy applications, they also provide *application manager apps*. That is, smartphone applications with an Android client capable of managing the discovery and download of Android applications directly from an Android device.

## 2.2 Data Collection Strategy

In order to collect our data systematically we implemented several spiders in Python based on the Scrapy framework [20]. For each appstore we implemented a distinct spider program which is distributed over a set of interconnected host machines. We designed our spiders to be stealthy in terms of requests per time unit, so as not to be identified as abusive from the monitored marketplaces. For these reasons, each spider instance takes into account *The Robots Exclusion Protocol*, that is, the *robots.txt* file, provided by the website of the appstore which currently being crawled.

We followed the same crawling strategy for all the candidate appstores. First, we crawl the whole appstore (each page with app information) to collect the main dataset, which is stored in a database. The main dataset consists of all apps available in the appstore the first day of our crawling process (this is the first snapshot of the appstore). Then, the crawling process is divided into two independent parts. In the first part we crawl each (already known) app in the main dataset every single day, so as to get new statistics (*e.g.* the new number of the downloads) of these apps. In the second part, we collect information for all the latest added apps in the appstore since the last crawling process and store them to the database, expanding our main dataset. The collecting information includes various statistics of the app such as the *number of downloads*, *user rating*, *price*, *current version*, *category*, *hosted URL etc.* The URL of the app is used as a

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

unique identifier that crawler instances take into account to gather statistics of the apps included in the main dataset. To extract all these data from the web pages of each appstore we used the default Scrapy's mechanism *XPATH selectors*, which are able to "select" certain parts of an HTML file specified by XPATH expressions. *XPATH*[2] is a language used to navigate through elements and attributes in XML and HTML documents. Apart from statistics, the crawlers download the last updated APK files (the app itself) for each app. That is, if an app has changed since the last crawling process the new APK file will be downloaded, collecting in this way a series of all the versions for every app in the marketplace. We have automated our crawlers so as to collect information of all the apps in each appstore on a daily basis.

## 2.3 Challenges

Here we describe some of the challenges we faced during the crawling process as well as techniques we used to address these issues.

One of the challenges was to collect data from pages that contain Javascript generated content. For instance, the app web pages of 1Mobile appstore contain some fields (*e.g.* the *user rating* field) produced by a piece of Javascript code. Scrapy by default is unable to collect content from a Javascript-rendered page. To address this problem we used Selenium Remote Control (RC) [21], a browser automation tool, combined with a headless Firefox browser running in an X virtual frame buffer (Xvfb). Therefore, the code of our spiders was altered to proxy the HTTP requests through the Selenium server with the controlled headless browser. Thus, Scrapy was able to collect all the required information from the already browser-rendered HTML page.

A second challenge was the fact that some Chinese appstores (*i.e.* AppChina and Anzhi appstore) apply rate limiting to hosts located away from China. We tried to download a set of APK files of these appstores, from different regions using the PlanetLab [34] and we observed that all sites far from China (*e.g.* in EU, US) exhibited lower download rates than those that were close to it. In order to deal with this issue properly, we used several Chinese PlanetLab nodes as proxies to download the APK files to machines running the crawler instances, for both AppChina and Anzhi appstores. We used *Wget* via SSH SOCKS proxy tunnels to the Chinese PlanetLab nodes (based on the *tsocks*[3] tool).

Furthermore, we noticed that AppChina appstore uses IP address blacklisting in which a host with a specific IP address may be blocked from the appstore for a period of time, if it exceeds a maximum request's limit within

[2] `http://www.w3.org/TR/xpath/`

[3] `http://tsocks.sourceforge.net/`

a certain interval. To overcome this issue we used a big set of PlanetLab nodes (roughly around 100) running a lightweight HTTP proxy in Python. When a crawler instance is about to send an HTTP request to AppChina appstore, it picks randomly one of the PlanetLab proxies to make this request instead and receives the forwarded response.

Finally, another one challenge was the fact that some of the apptores, particularly 1Mobile and AppChina, changed their web graphical interface during the measurement interval. This had as a result the XPATH rules, we had defined in Scrapy XPATH selectors, not to be effective according to the new changes. To address this problem we monitored the graphical interface structure of every appstore in our dataset on a daily basis and adjusted our XPATH rule set to any of these changes.

In Figure 2.1, there is an overview of the data collection strategy we followed in order to gather the information needed for our analysis.

## 2.4 Collected Data

The previously discussed crawling process resulted in the four datasets:

| appstore | set | first crawling date | last crawling date | period | number of apps | number of APKs (all versions) |
|---|---|---|---|---|---|---|
| *SlideMe* | free | 2012-03-01 | 2012-08-01 | 5 months & 3 days | 16,578 | 31,731 |
| | paid | 2012-03-01 | 2012-08-01 | 5 months & 3 days | 5,606 | - |
| | total | 2012-03-01 | 2012-08-01 | 5 months & 3 days | 22,184 | 31,731 |
| *1Mobile* | total (free) | 2012-05-15 | 2012-08-01 | 2 months & 18 days | 156,221 | 219,508 |
| *AppChina* | total (free) | 2012-03-30 | 2012-06-03 | 2 months | 55,357 | 81,969 |
| *Anzhi* | total (free) | 2012-06-04 | 2012-08-03 | 2 months | 60,197 | 89,441 |
| **Total** | - | - | - | - | **316,143** | **454,427** |

TABLE 2.1: Summary of the collected data.

- *SlideMe*: The database contains information of 22,184 number of apps that were available in SlideMeappstore from $1^{st}$ of March until $1^{st}$ of August 2012 (5 months). The 25.3% (5,606) of these apps were paid apps, while the rest 74.7% (16,578) were free apps. The total number of all collected versions of APK files is 31731.

- *1Mobile*: There is information of more than 150,000 apps of 1Mobilemarketplace in the database for the period between May $15^{th}$ and August $1^{st}$ 2012 (2 and a half months). This appstore is the largest, in terms of number of apps, in our dataset and contains only free apps. The total amount of collected APK files is 219,508.

- *AppChina*: The database contains information of 55,357 apps of this Chinese appstore for the period between $30^{th}$ March and $3rd$ June 2012 (2 months). The total APK files collected by this appstore is 81,969.

- *Anzhi*: We have information of 60,197 apps of Anzhi market stored in our database, from $4^{th}$ of April until $3^{rd}$ of August (2 months). The total number of all the collected APK files amounts to 89,441.

The total number of monitored apps is 316,143, while the total number of APK files is 454,427. For each application, apart from the harvested information, all the versions of APK files are also collected in the measurement interval. Table 2.1 summarizes all the collected data.

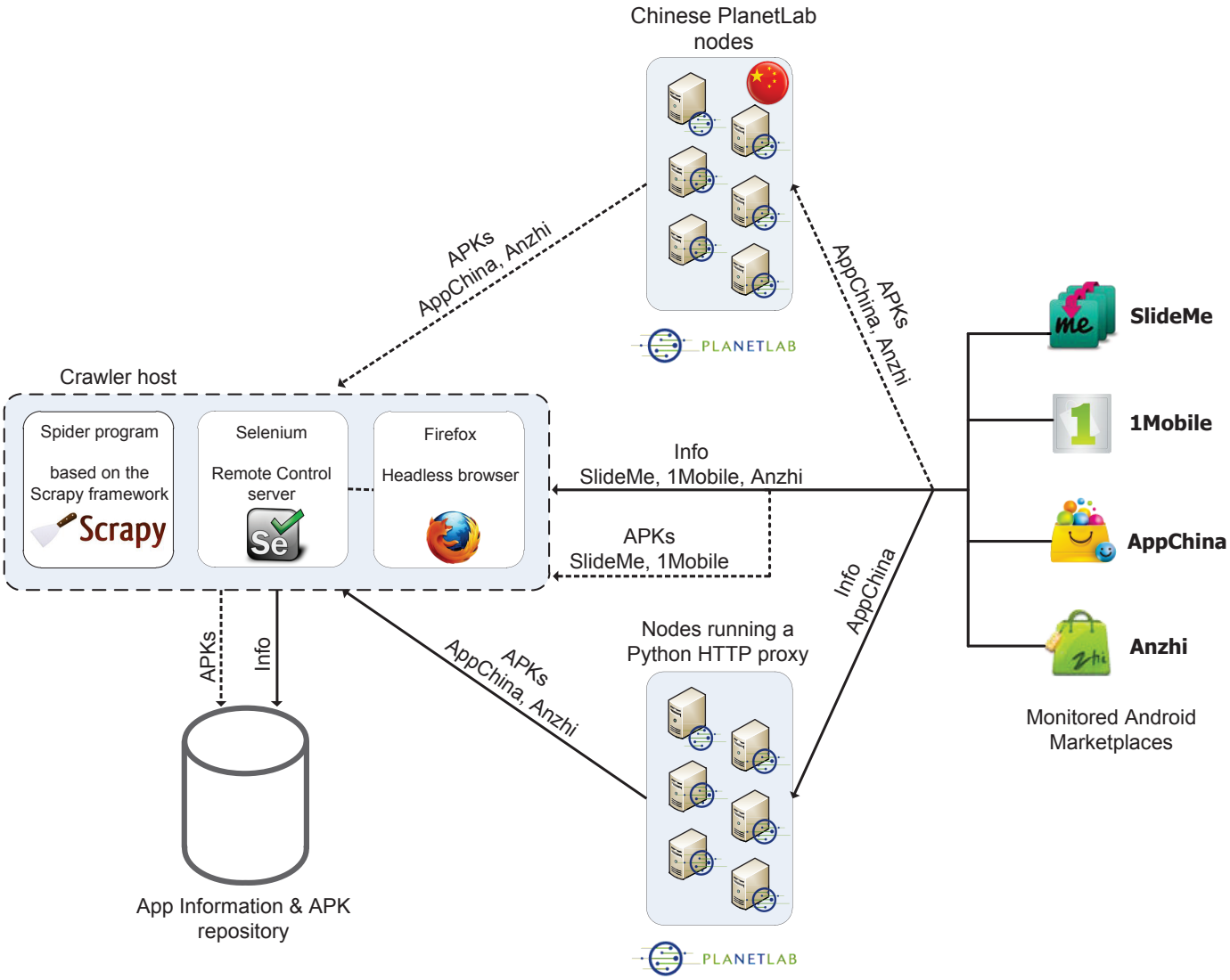



FIGURE 2.1: Overview of the data collection strategy.

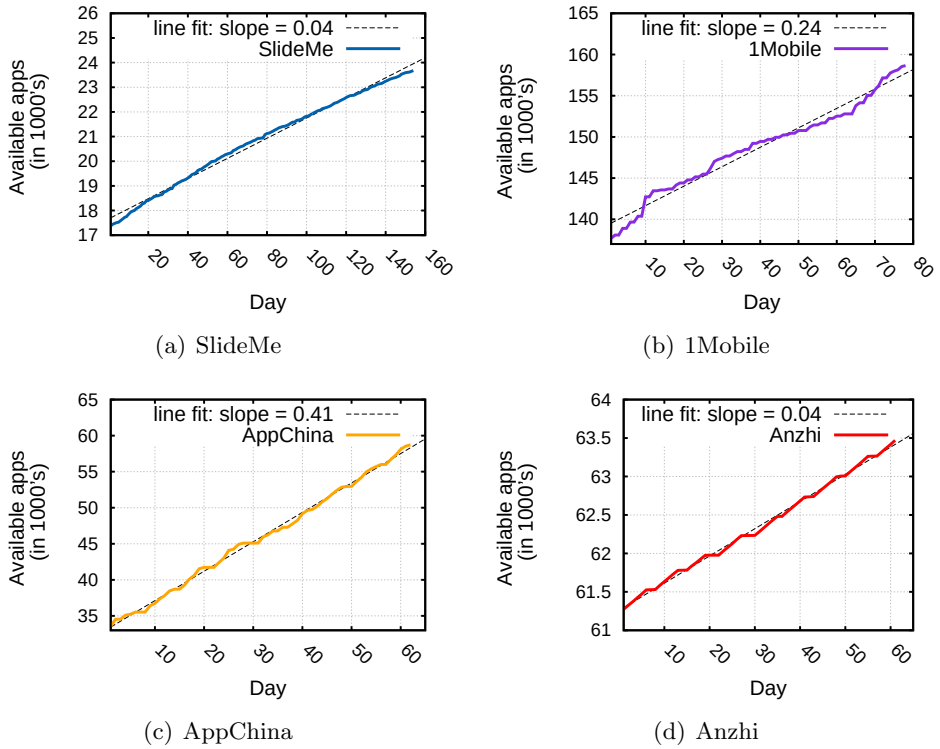
*The solid lines indicate information data flow (e.g. number of downloads, user ratings etc.). The dashed lines indicate APK files' flow.*

# 3
# The Rise

In this chapter we explore the size of the monitored marketplaces in terms of number of available apps and investigate how this number changes over time. Moreover, we further investigate how these apps are consumed by the users, that is, how frequently these apps are downloaded though time. Finally, we examine this downloads' growth among different applications, where we are interested to see whether there are apps whose number of downloads increase very rapidly through time or apps with no downloads at all.

## 3.1 Number of apps

In our first experiment, we set out to explore how many apps are hosted in each marketplace. Figure 3.1 plots the number of apps hosted in each appstore as a function of time for our measurement interval. We can see that each appstore hosts tens of thousands of apps. For example, SlideMe hosts about 24 thousands of apps, 1Mobile hosts about 160 thousands of apps, AppChina hosts around 60 thousands of apps and finally Anzhi hosts almost 64 thousands of apps. Interestingly, we see that app developers add new apps on a daily basis. Indeed, as Figure 3.1 shows the total number of apps hosted increases linearly with time. Note, however that the rate of increase is different for each appstore. For instance, we see that AppChina rises much faster than the other marketplaces, achieving an increase of 75% in 62 days, reaching an increase rate of close to 1.2% per day, *greater than the average daily growth of the official Android Market (which is in average 0.4%, as we have already shown in Section 1.1).* This outcome can be verified




(a) SlideMe (b) 1Mobile

(c) AppChina (d) Anzhi

FIGURE 3.1: Number of available apps in the marketplaces as a function of time.

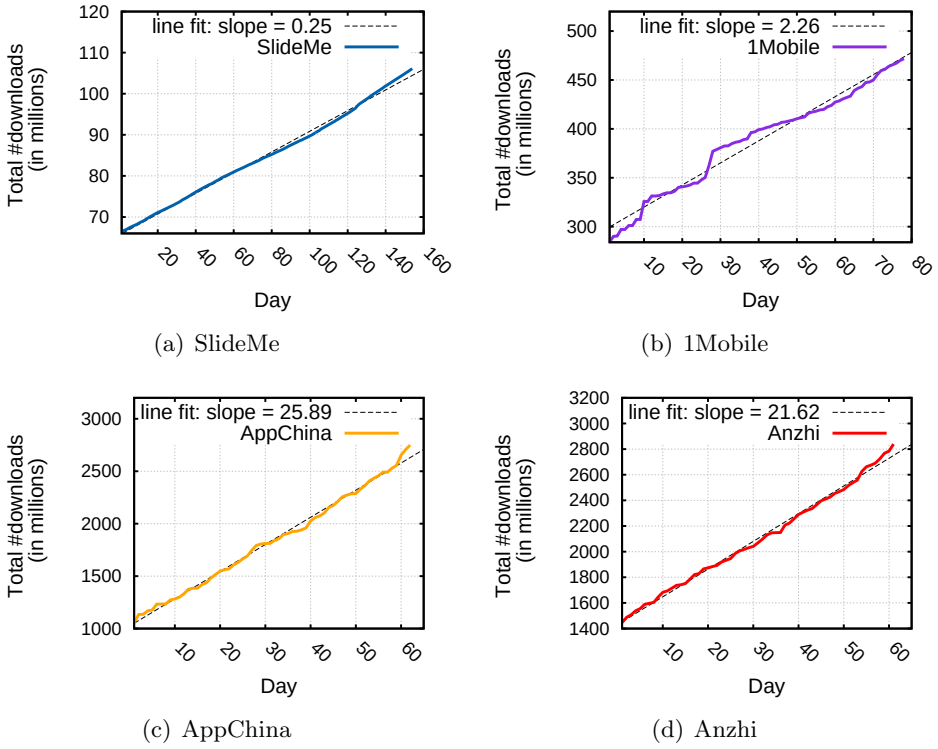
in part by Flurry, which claims that China is the fastest growing share of iOS and Android active devices with a 401 percent growth between July 2011 and July 2012 [14]. This remarkably vast share of mobile devices in China seems to attract the interest of app developers, who try to promote their software in this large market for profit. Another possible reason that could explain this extraordinary rise may be the fact that Android Market was blocked from China in 2011 [11]. This fact may have led many smartphone users to rely on other local marketplaces for downloading applications.

| | App Growth Rate | | |
|---|---|---|---|
| **Marketplace** | **Total** | **Monthly** | **Daily** |
| SlideMe | 36.2% (154 days) | 7.0% | 0.2% |
| 1Mobile | 15.2% (78 days) | 5.8% | 0.2% |
| AppChina | 74.9% (62 days) | 36.2% | 1.2% |
| Anzhi | 3.5% (61 days) | 1.7% | 0.05% |

TABLE 3.1: Marketplaces' growth rate in terms of number of available apps.

Table 3.1 presents the growth rate in terms of number of apps for the 4 monitored marketplaces. We can see that except from AppChina, the rest marketplaces note reasonable daily growth rate from 0.05% to 2%, that is about the half of the one observed in the official Android Market.

## 3.2 Total Downloads




(a) SlideMe (b) 1Mobile

(c) AppChina (d) Anzhi

FIGURE 3.2: Number of total downloads in the marketplaces as a function of time.

Figure 3.1 shows that the number of apps produced and hosted in appstores increases linearly with time. It would be interesting to see how frequently are these apps downloaded (*i.e.* consumed) by the users. Figure 3.2 shows the total number of downloads for all apps, hosted in each appstore, as a function of time. Interestingly, we see that the number of downloads increases rapidly with time, and in all appstores approaches a straight line. The most rapid increase in downloads can be found in AppChina, where *the total number of downloads almost doubled in a month and a half.* The next largest increase appears to be in Anzhi, where *the total downloads were nearly twice in number within 2 months.* More specifically, in Table 3.2 we see that in the two Chinese appstores, the monthly growth rates of down-

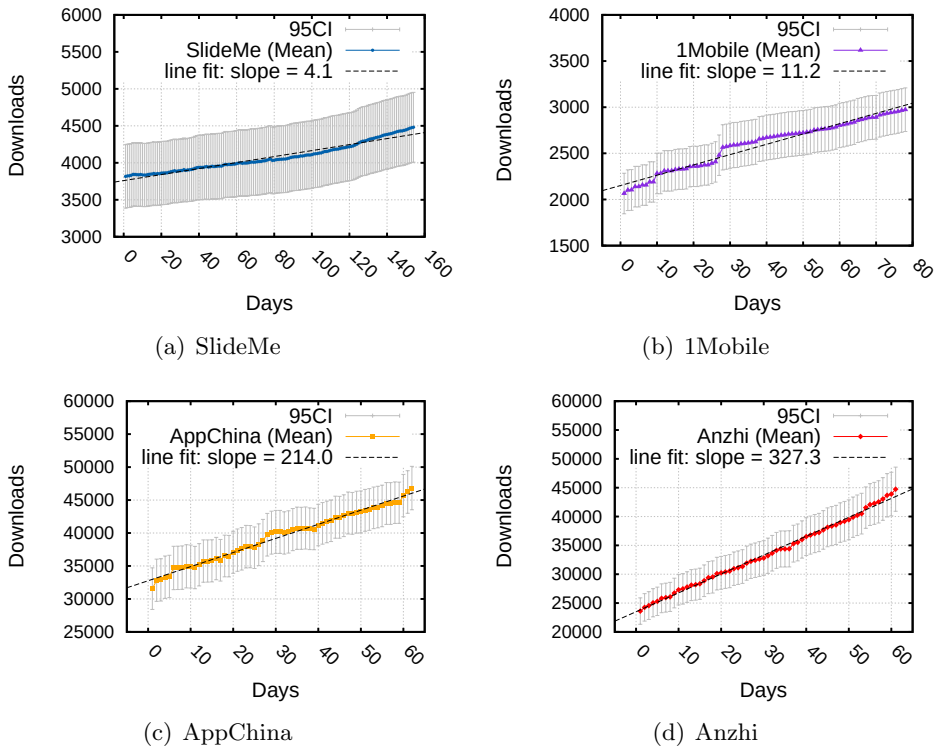
loads (77% for AppChina, 47% for Anzhi) are much higher than those of the other two marketplaces (11% for SlideMe and 25% for 1Mobile). This result confirms the huge size of market share in terms of active smartphones, in China. Moreover, these appstores seem to be very promising for mobile app developers who aim to increase the number of downloads (hence popularity) of their apps. For instance, the video game company *Trilena Games*[1] states that by putting one of their applications in Anzhi marketplace resulted in collecting over 1100 downloads within a single week [10].

| | **Downloads' Growth Rate** | | |
|---|---|---|---|
| **Marketplace** | **Total** | **Monthly** | **Daily** |
| SlideMe | 59.9% (154 days) | 11.6% | 0.4% |
| 1Mobile | 65.9% (78 days) | 25.3% | 0.8% |
| AppChina | 159.5% (62 days) | 77.2% | 2.6% |
| Anzhi | 96.2% (61 days) | 47.3% | 1.6% |

TABLE 3.2: Total downloads growth rate.

[1] http://www.trilenagames.com/

## 3.3 Mean Downloads Through Time




(a) SlideMe (b) 1Mobile

(c) AppChina (d) Anzhi

FIGURE 3.3: Mean downloads of apps through time.

In this section, we try to explore a little deeper the perception that Chinese marketplaces are very propitious for app developers, who intend to earn many downloads from their apps, thus to gain much popularity, within a short time period. In order to investigate deeper this issue, we plot the mean number of downloads of apps in each appstore through time. The results are illustrated in Figure 3.3. We see that regression lines of fitted data curves for AppChina and Anzhi are steeper than those of two other appstores, with significantly greater slope values. Surprisingly, we see that slope values of fitted lines in Chinese appstores are *one order of magnitude* greater than those of the other monitored marketplaces.

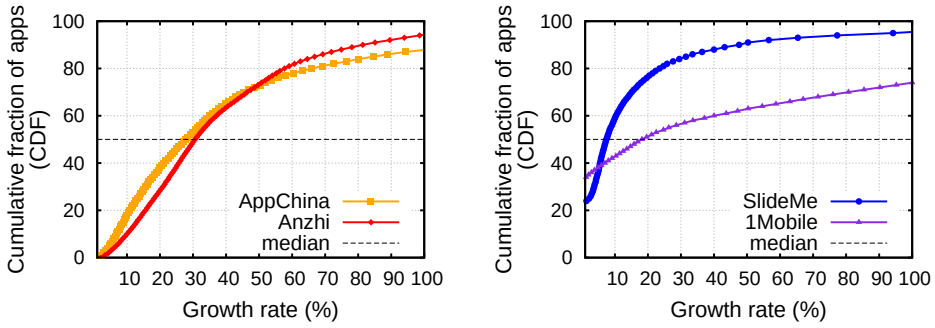
| Marketplace | Mean Downloads' Growth Rate | | |
|---|---|---|---|
| | Total | Monthly | Daily |
| SlideMe | 17.4% (154 days) | 3.4% | 0.1% |
| 1Mobile | 65.9% (78 days) | 16.9% | 0.5% |
| AppChina | 48.3% (62 days) | 23.4% | 0.7% |
| Anzhi | 89.4% (61 days) | 43.9% | 1.4% |

TABLE 3.3: Mean downloads growth rate.

Table 3.3 shows the growth rate of mean downloads of apps for the measurement period (total), as well as the per month and per day growth rates. The monthly growth rate of mean downloads for AppChina and Anzhi are greater than those of SlideMe and 1Mobile indicating that Chinese marketplaces can benefit an app developer in terms of a faster popularity growth of her app (more downloads). Stunningly, the Anzhi appstore appears to be the best option with a monthly growth rate of 44%. This outcome confirms the information of Trilena Games company that noticed a huge rise of her apps' downloads in Anzhi within a few days.

## 3.4 Growth Distribution Among Different Apps

Although Figure 3.2 clearly shows that the total number of downloads grows linearly with time, we would be interested in understanding how this growth is distributed among different apps. That is, are there applications whose number of downloads grow very quickly? Are there any applications that are not downloaded at all? Figure 3.4 plots the CDF of growth rate of




(a) Chinese marketplaces

(b) SlideMe and 1Mobile

FIGURE 3.4: CDF of Growth Rate of apps.

different apps in the four appstores for a two months period. The growth rate of an app is defined as the ratio of the number of times the app has been downloaded within the measurement interval, over the number of downloads

the app had at the beginning of the measurement. For example, if at the beginning of the interval the app had 100 downloads, and at the end of the measurement it had 130 downloads, then its growth rate is (130-100)/100 = 30%. Figure 3.4 shows that apps at different appstores enjoy different growth rates. For instance, in AppChina and Anzhi the median app ($y$ axis at 50) enjoys a growth rate of 27% and 30% respectively. The upper 10% of apps in these appstores enjoy a growth rate greater than 80%, while the upper 5% of the apps enjoy growth rates higher than 130% for AppChina, and higher than 250% for Anzhi (not shown in the Figure). It is very interesting to note that all apps in AppChina and Anzhi had a positive growth rate: all applications were downloaded at least once. On the contrary, in SlideMe and 1Mobile there is a large percentage of apps which have not received any downloads at all, during the measurement interval. For example, as many as 23% of the applications in SlideMe and as many as 31% of the apps in 1Mobile, did not receive a single download during the measurements. Along these lines, the median app received only 18% growth rate in SlideMe and almost 7.5% in 1Mobile, and the upper 10% of the apps had a growth rate of at least 47% in former and 220% in latter, which are much lower than those of AppChina and Anzhi.

It seems that Chinese markets are a much better place for developers to upload their applications. Indeed, after visually inspecting the first page of AppChina, we observed that it hosted both popular and unpopular applications. Actually, next to apps that had millions of downloads we saw apps which had been downloaded only a few thousands of times. This strategy seems to exploit the popularity of high popular apps to enhance the popularity of unpopular ones.

## 3.5 The "Forgotten" Apps

Figure 3.4 clearly shows that for SlideMe and 1Mobile there exists a large percentage of apps which were not downloaded not even once during the measurements interval. It would be interesting to see how large is this set of "forgotten" apps and how does it change as a function of the length of the duration interval.

For example, if the measurement interval is too small, then it is reasonable to have a large number of applications that received no downloads. On the other hand, if the interval is too large, then it is reasonable for most, if not all, of the apps to receive at least one download. Figure 3.5 shows the change in the percentage of "forgotten" apps as a function of the duration of the interval. We see that in all cases this percentage drops fasts and after 5-10 days reaches a plateau that is either flat or drops very slowly. Especially, for Chinese marketplaces (AppChina and Anzhi) this plateau remains at 0%. The plateau for SlideMe is at 24.5% in the first 60 days, and

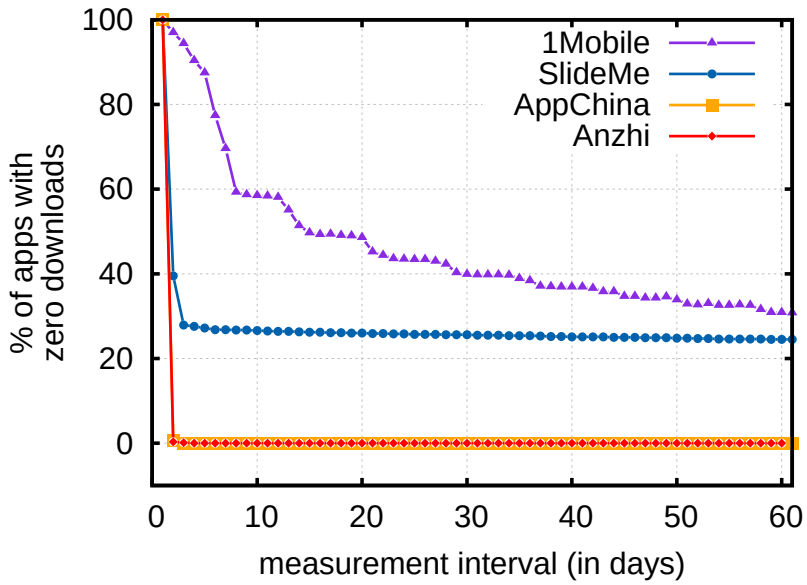



FIGURE 3.5: Percentage of apps with zero downloads for the entire duration of the measurement interval.

reaches the 21% in the end of its measurement interval, which is 154 days (5 months). In 1Mobile, this plateau approaches the 26%. We further analyse our data and found that this portion of apps largely correspond to very old old apps (more than 1 year old) for 1Mobile and to a mix of old and paid apps for SlideMe .

## 3.6 Summary

In brief, this chapter explored the number of available applications of the monitored appstores and how this number increases as a function of time. We observed high growth rates ranging from 1% (Anzhi) up to 36% (AppChina) per month. Also, we investigated the number of downloads of the apps through time and realized that Chinese marketplaces noted much higher growth rates than the rest monitored ones, indicating a more promising place for developers who want their apps to become popular (get more downloads) at a faster pace. Finally, we attempted to understand how the downloads' growth rate is distributed among different apps in our datasets and noticed that all of the apps of the two Chinese appstores had positive growth rates (at least one download per app), while the other two marketplaces had a proportion of apps with zero change (no downloads at all). By analysing our data, we concluded that the apps with no downloads were very old apps in 1Mobile, and a mix of old and paid apps in SlideMe. On the

other side, Chinese marketplaces seem to follow a smart strategy of placing unpopular apps in recommendations of popular ones, enhancing in this way the popularity of the former. This appears to be the reason why in these appstores were not observed applications with zero number of downloads.

# 4

# App Popularity

In this Chapter we study the app popularity based on the distribution of downloads of each app among the different appstores. We compare our results with findings from similar studies on other domains, like web content popularity [31], file sharing workloads [46], and user-generated video content analysis [32]. Furthermore, we examine which parameters may affect the app popularity, such as user ratings and the presence of the cost on some applications (*paid* apps).

## 4.1 Is There a Pareto Effect?

Previous studies [31, 32, 46] have shown that web content and file downloads usually follow the "Pareto Principle": that is, 20% of the documents are responsible for 80% of the downloads. Figure 4.1 shows the CDF of the percentage of app downloads as a function of the app ranking (apps ranked from most popular to least popular) for the different appstores. The results confirm that a small percentage of apps is responsible for a large percentage of downloads. For example, both in AppChina and Anzhi, about 10% of the apps ($x = 10$) are responsible for close to 90% of all downloads. Similarly, 10% of the 1Mobile apps are responsible for more than 85% of the downloads, and 10% of the SlideMe apps are responsible for more than 70% of all the downloads in this appstore.

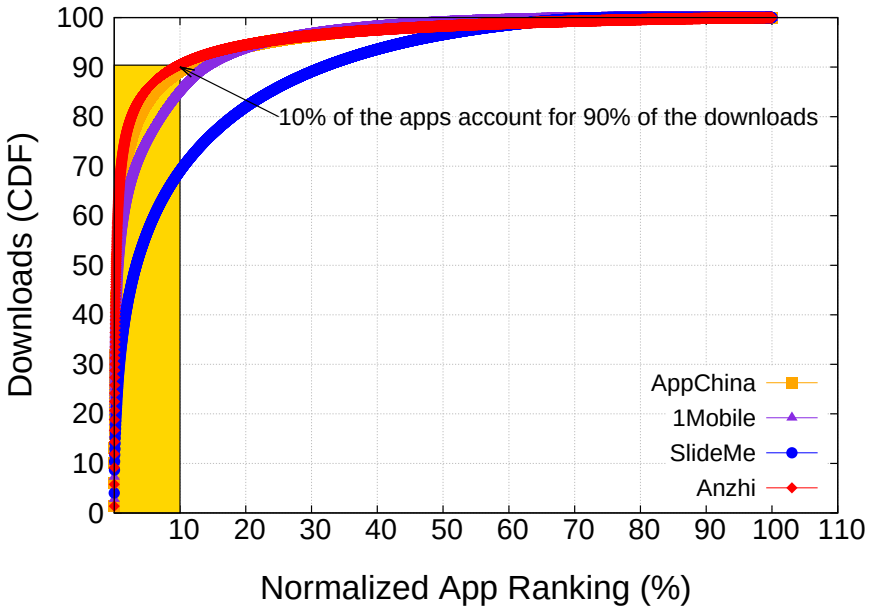



FIGURE 4.1: A few apps account for most of the downloads.

We see that this uneven distribution of popularity goes all the way into the top 1% of the applications. Indeed, in Figure 4.2 we see that as little as 1% of the apps are responsible for more than 70% of downloads in Anzhi, than 60% of downloads in AppChina, than 55% in 1Mobile and more than 30% of downloads in SlideMe.

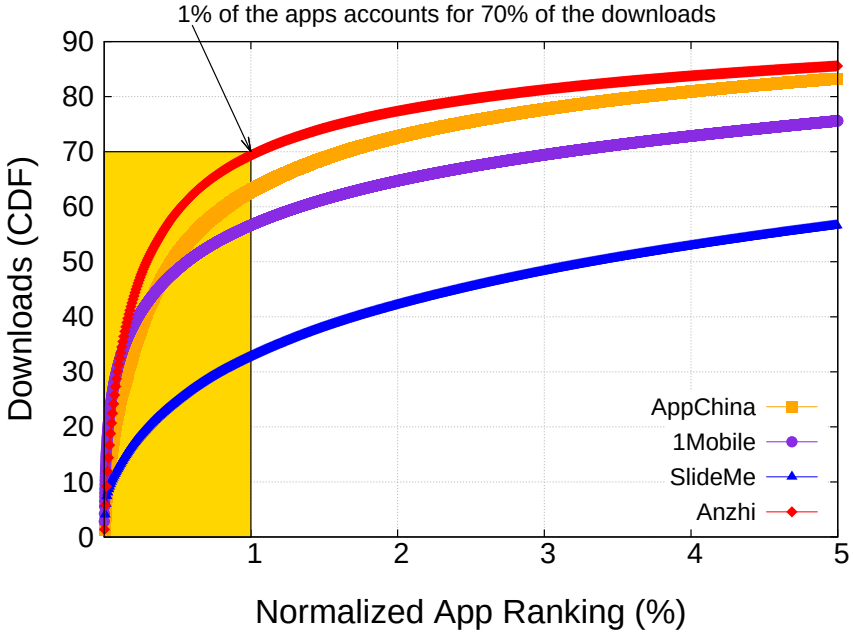
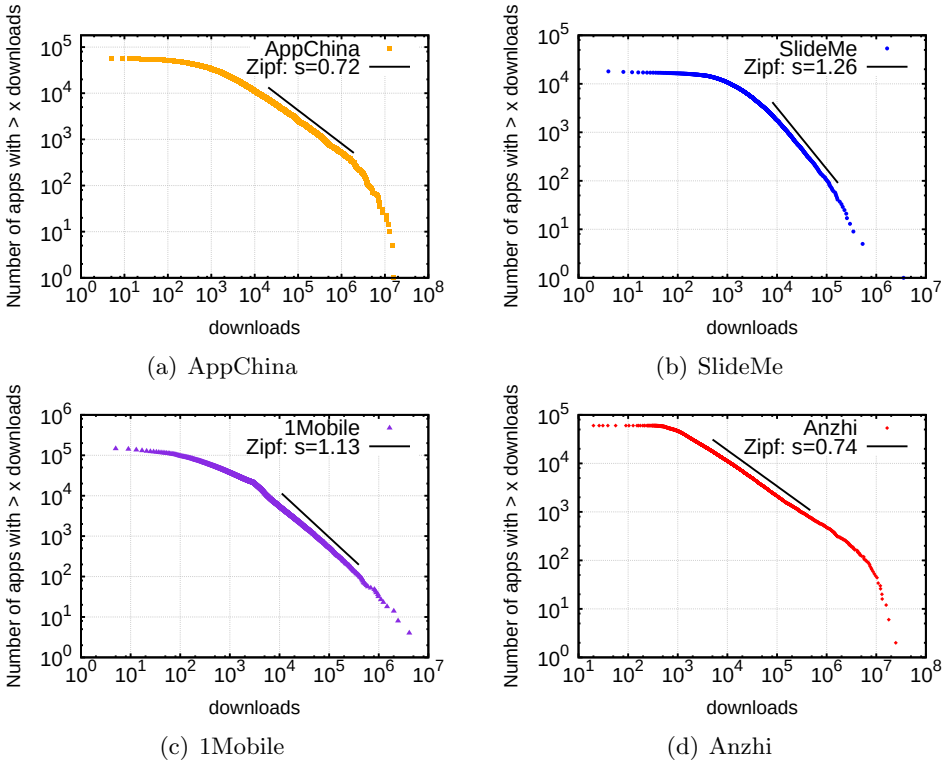



FIGURE 4.2: A few apps account for most of the downloads (Zoom).

## 4.2 Is There a Power-Law Behavior?

Although it is clear that app downloads follow a Pareto principle, we would like to explore whether app downloads follow a power law distribution much like web downloads do [31].



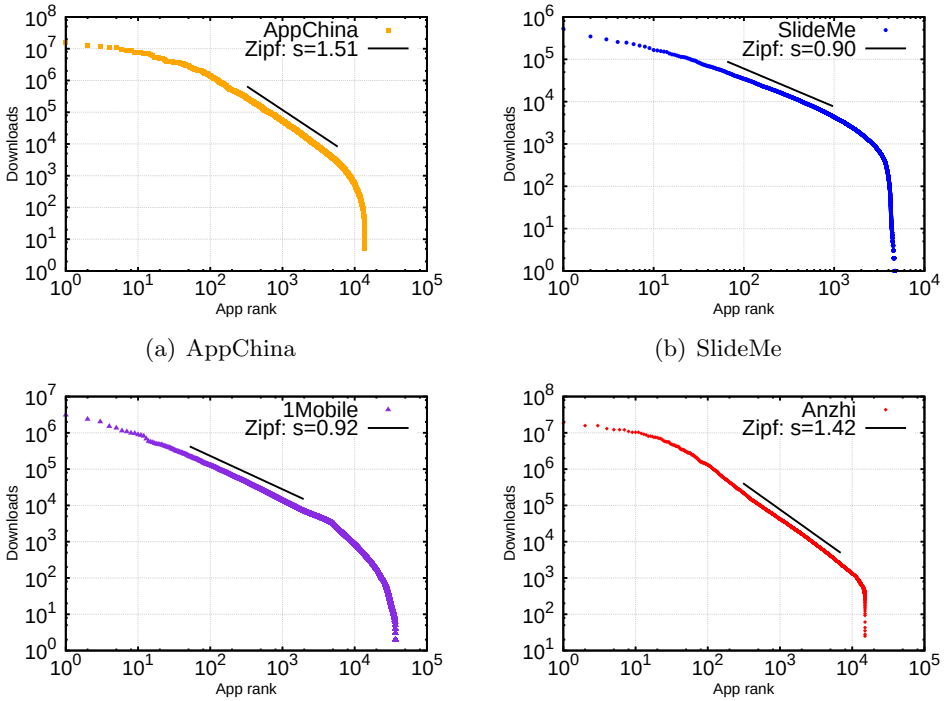

(a) AppChina (b) SlideMe

(c) 1Mobile (d) Anzhi

FIGURE 4.3: Cumulative distribution of total downloads per app.

Figure 4.3 shows the CDF of the number of apps ($y$ axis) which have exceeded a given number of downloads ($x$ axis). That is, the value on the $y$ axis is equal to the number of apps which had more than $x$ downloads. We see that all distributions share a similar pattern: their main "trunk" has a linear slope indicating a ZIPF distribution, which is truncated at both ends (*i.e.* for both for small $x$ and large $x$ values). The truncation for small $x$ values seems to follow similar patterns shown in the downloads of file sharing systems [46] and video downloads in YouTube [32]. This truncation, on file sharing systems as well as on user-generated video content systems was attributed to the "fetch-at-most-once" principle. That is, content shared in a file sharing system, such as videos, movies, *etc.*tend to be downloaded at most once by each user. Therefore, the curve for very popular content (*i.e.* small values in the $x$ axis) tends to be flatten out and reach a value close to the number of users in the system. To understand the shape of the download curve in our case, we replot the data in Figure 4.4 as the number

of downloads per app. All apps are sorted in the $x$ axis according to their rank, based on their total downloads.




(a) AppChina (b) SlideMe

(c) 1Mobile (d) Anzhi

FIGURE 4.4: Total downloads per app as a function of app's rank.

We see that for very small $x$ (*i.e.* very popular apps) the number of downloads stays almost horizontal (especially in AppChina and Anzhi), which is probably due to the "fetch-at-most-once" principle observed both in peer-to-peer systems and YouTube [32, 46]. It is reasonable to expect that users will also download each app at most once, apart from apps which are updated. To ensure that "fetch-at-most-once" property exists in appstores as well, we analyzed our data to find the percentage of apps that were updated during our measurements. We have this information, since we collect all the APK versions of the applications in our dataset. Figure 4.5 depicts the CDF of the number of updates per app in the four appstores for a period of two months. As we can see, only a small percentage of apps have updates during this period. In SlideMe and 1Mobile only 20% of the apps were updated, while in Anzhi this percentage is even smaller, equals to 15%. The larger proportion of updated apps is observed in AppChina, which is close to 34% of the total apps in the appstore. Nonetheless, we observe that few apps have been updated during our measurements and most of them have a very small number of updates (they are updated very rarely). Moreover,

we speculate that not all users update their downloaded apps. This entails that fetch-at-most-once property is a general property of marketplaces which limits the downloads of each user for the same app, because apps do not change so often (as *e.g.* web pages may do).

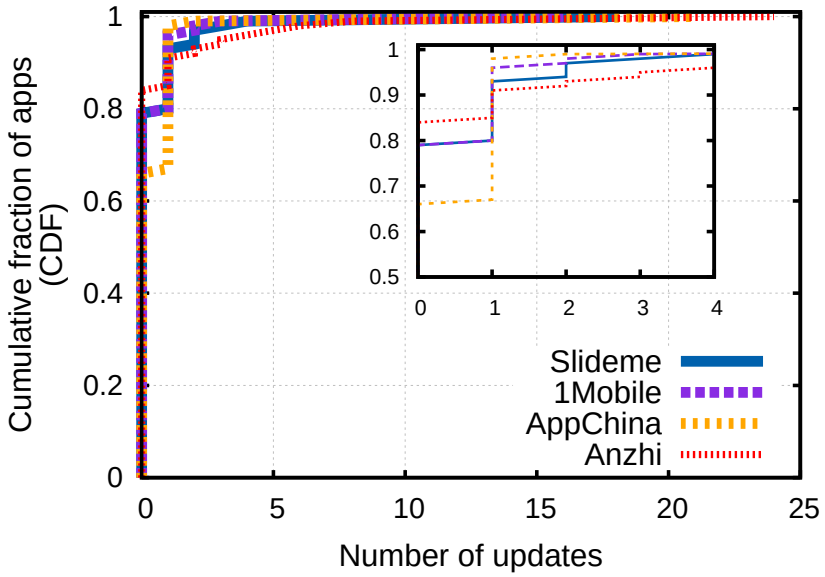



FIGURE 4.5: CDF of the number of updates per app for a period of two months.

We observed that only a small percentage of apps were updated during our measurements.

In addition to the truncation for small $x$ values, Figure 4.4 clearly shows that there is a significant curvature for large $x$ (less popular apps) as well. Observed in user-generated content downloads [32], but not in file sharing [46], this curvature has been thought to be attributed to search and recommendations engines [33,48]. We feel that this curvature is an instance of a more general phenomenon, which we call the *clustering effect*. The clustering effect, suggests that apps are grouped into (static or dynamic) sets. Apps within the same set (cluster) are correlated: if a user downloads one of them, then the same user will probably download another app of the same set rather than switching to another set, or to a totally unrelated app. These clusters can be formed, for example, by the semantic classification of apps to categories, by user communities as a result of positive comments, by the appstore as a result of the recommendation algorithm used, or by other grouping forces. To validate our clustering effect hypothesis, we define and approximate the temporal affinity of user downloads among several app categories in Chapter 5, we propose a model for appstore usage and app

download patterns, and we validate our model based on a simulation study comparing the simulated with the observed data in chapter 6.

## 4.3 The effect of User Ratings

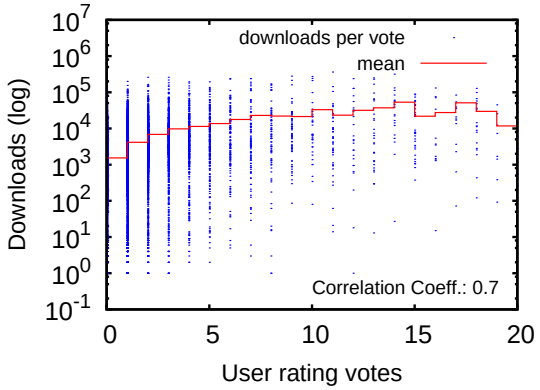
We have already show that the distribution of applications' popularity, in different mobile marketplaces, exhibits a power-law behavior which represents the *rich-get-richer* principle. However, in the resulted distributions we noticed some deviations from the *pure* power-law behavior. These deviations may arise because of various factors including results of search engines or recommendation systems, as already mentioned. In this section, we explore how the user ratings can affect the app popularity.

Of our monitored appstores, SlideMe, 1Mobile and Anzhi provided information about user ratings along with the exact number of *votes* (the number of users that rated an app) during our measurement period. User ratings indicate the number of times an app has been downloaded or evaluated by users. To examine the relationship between the user ratings and app popularity (app downloads), we calculated the average app downloads for different votes' bins. The results are illustrated in the Figure 4.6. We choose to illustrate only the *votes bins* that had at least 10 downloads' samples. As we can see, there is a connection between the number of downloads and the votes. In particular, we see that the mean downloads per vote is increasing for higher vote values (that is, number of ratings). This implies a relationship between the app popularity (app downloads) and the user ratings. To quantify this relationship we calculated the Pearson's correlation coefficient between the average downloads per vote and votes. The results are summarized in Table 4.1.

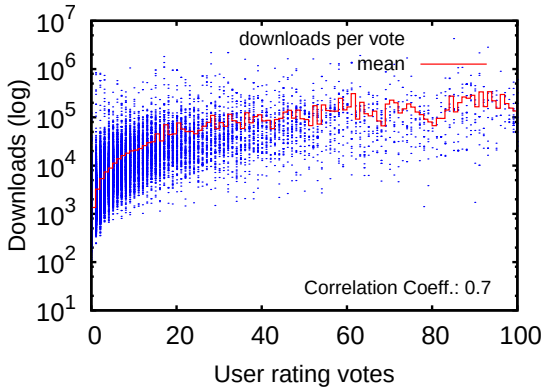
| Appstore | Pearson's correlation coefficient (R) |
|---|---|
| SlideMe | 0.69 |
| Anzhi | 0.78 |
| 1Mobile | 0.84 |

TABLE 4.1: Pearson's correlation coefficient products between app downloads and user rating for different appstores.

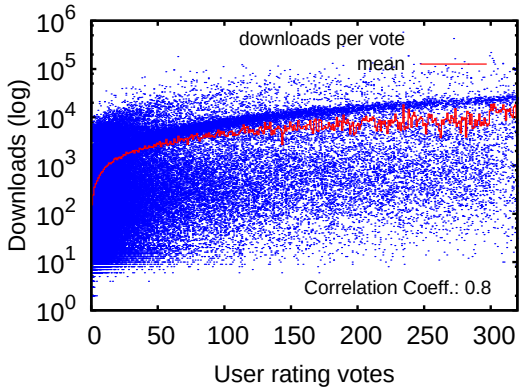
As we can see, there is a quite strong relationship between downloads and user ratings for the three different appstores. This interesting result implies that the average smartphone user tends to rate more often the most popular applications. Thus, the app popularity can be reflected by user ratings too. Moreover, it seems that user ratings boost the app popularity, as influences the user preferences. In other words, users are more likely to download an app that have high number of rating than a poor rated one.




(a) SlideMe




(b) Anzhi

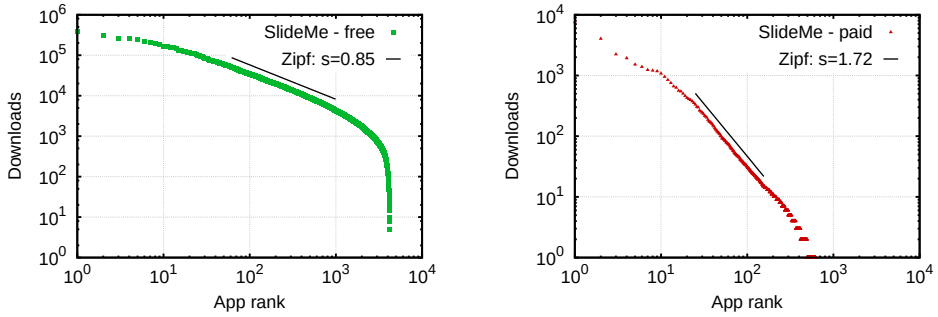



(c) 1Mobile

FIGURE 4.6: App ratings vs downloads.

## 4.4 The Influence of the Cost

Although most of the appstores that we study offer all apps free of cost, SlideMe provides both free and *paid* apps. The latter ones usually have somewhat more advanced functionality and usually do not include advertisements. To understand the download patterns of paid and free apps, we plotted the distribution of downloads for free and paid apps separately. The




(a) SlideMe - free (b) SlideMe - paid

FIGURE 4.7: Distribution of total downloads per app for free and paid apps.

We see that paid apps clearly follow a power-law distribution: few apps have a (relatively) large number of downloads, while a large number of apps have a tiny number of downloads.

results are illustrated in Figure 4.7. Interestingly, we see that paid apps follow more clearly a power-law distribution. This is probably due to the fact that users are more selective when downloading paid applications. Thus, less popular apps tend to stay that way and are deprived from any casual downloads. On the contrary, free apps enjoy more downloads and even less popular apps get a decent number of downloads. This inference is better depicted in Figure 4.8, where we can see the CDF of downloads distribution of paid and free apps at the end of our measurement period. We observe that more than half of the paid apps (55%) have not receive any downloads. There is a fairly large proportion of paid apps, which corresponds to 32% of total paid apps, that have got a number of downloads ranges from 1 to roughly 85 downloads and a very small percentage, the upper 3% in the graph, that have received an amount of downloads from almost 130 to a number greater 600,000. In contrast, the distribution of downloads for free apps appear to be more uniform than this of paid apps. The most obvious difference is the fact that there is no free app with zero number of downloads. Moreover, the largest percentage of the free apps have received a number of downloads that ranges from 60 to almost 70,000 of downloads. There is a very small fraction of applications (less than 1%) that have received almost 3 downloads, as well as a very small percentage (the upper 1% in the graph)

that has received a number of downloads up to 3,550,000. However the median free app has received about 1,600 downloads, while the median paid app has no downloads at all. From the above graph, it is clear the obvious conclusion that smartphone users prefer to download apps for free instead of paying for them.

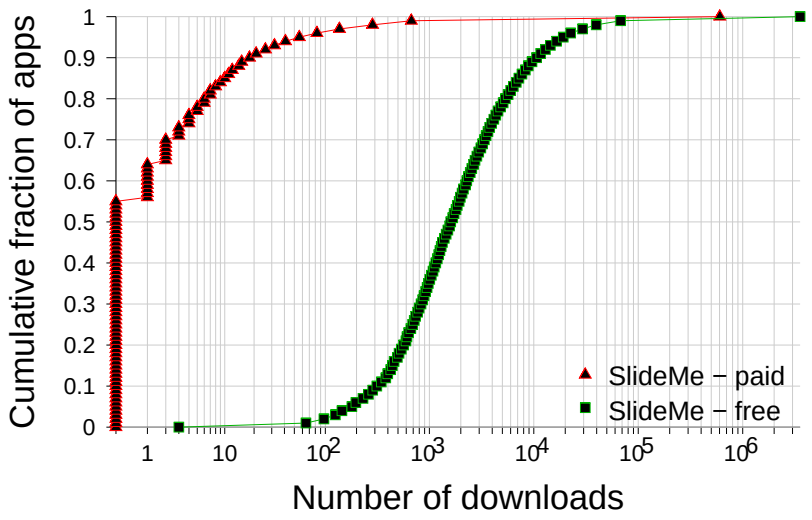



FIGURE 4.8: CDF of downloads distribution of paid and free apps in SlideMe appstore.

Further, we would like to see whether the popularity depends on the app's price. To this end, we compared the downloads of each app with its respective price. In Figure 4.9 we plot the average downloads per app's price (downloads are binned into prices) versus the corresponding prices. We see that the correlation between these two quantities is -0.22 indicating a negative relationship, which means that the higher the price of an app, the lower the probability of such an app to become popular. The average number of downloads for price bins greater than 50 (not shown in the figure) is equal to zero. Moreover, in Figure 4.10 we plot the percentage of apps per price bin versus price. Likewise, we find there is also a negative correlation between the amount of the apps, which are at the same price levels, and their price. Consequently, most of developers seem to give lower prices to their apps hoping to gain more popularity in the first steps.

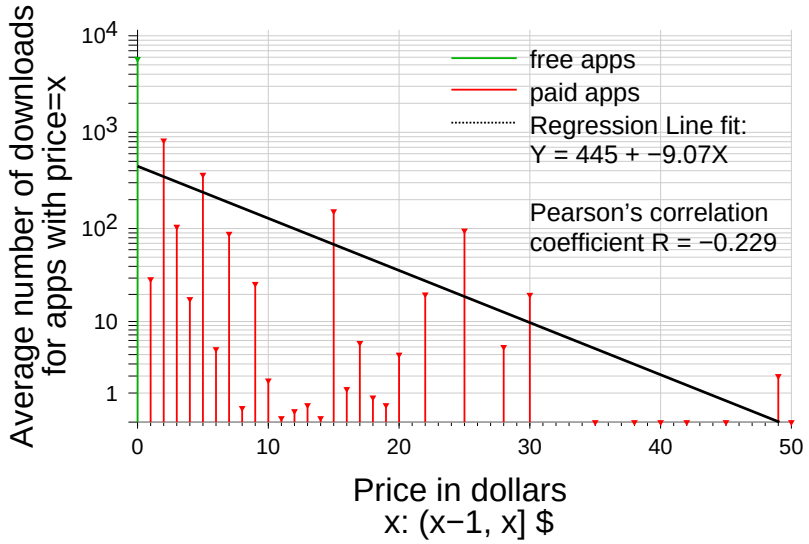



FIGURE 4.9: Average downloads versus prices.

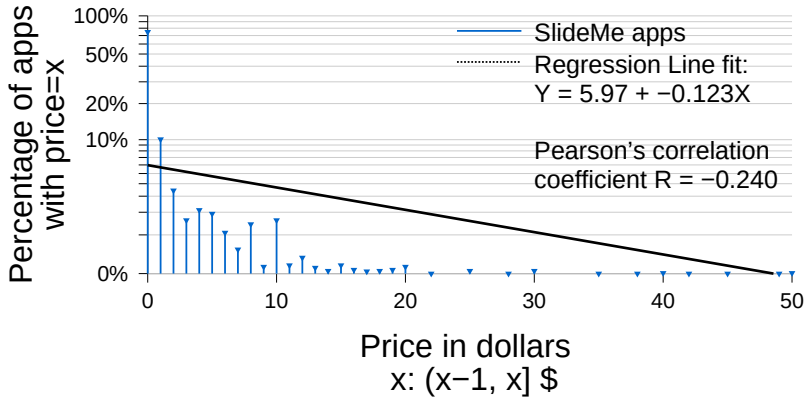



FIGURE 4.10: Percentage of apps versus prices.

In Chapter 7, there is a more detailed study on paid apps, where we explore how the income of paid apps is distributed among developers and among different categories. We also attempt to outline the behavior of app developers and recognize their different strategy patterns through statistics and observations from our dataset. Finally, we try to estimate whether free apps can make higher income from paid apps using different means than pricing, such as through advertisements.

## 4.5 Stability of TOP-10 and TOP-100 Apps Through Time

This section provides an analysis of the most popular apps of the four monitored appstores through time. The main question we attempt to answer is whether the most popular apps change considerably over time. For this purpose, we calculated the top-10 and top-100 sets of apps in terms of popularity, for each day of our measurement interval (2 months).

| appstore | set | unique apps | Mean (Std) per day | | |
|---|---|---|---|---|---|
| | | | residence (days) | set difference (%) | dice coefficient |
| SlideMe | TOP-10 | 12 | 50.0 ± 17.5 | 0.3 ± 1.8 | 0.9 ± 0.1 |
| | TOP-100 | 116 | 51.7 ± 15.0 | 0.3 ± 0.5 | 0.8 ± 0.1 |
| 1Mobile | TOP-10 | 13 | 46.2 ± 19.1 | 2.3 ± 4.2 | 1.0 ± 0.1 |
| | TOP-100 | 131 | 45.8 ± 20.2 | 0.6 ± 1.3 | 0.9 ± 0.1 |
| Anzhi | TOP-10 | 15 | 40.0 ± 23.3 | 2.0 ± 4.0 | 0.8 ± 0.2 |
| | TOP-100 | 165 | 36.4 ± 21.9 | 1.3 ± 1.2 | 0.6 ± 0.2 |
| AppChina | TOP-10 | 23 | 26.1 ± 23.4 | 4.7 ± 5.6 | 0.6 ± 0.3 |
| | TOP-100 | 184 | 32.6 ± 21.4 | 1.7 ± 1.7 | 0.5 ± 0.2 |

TABLE 4.2: Statistics of TOP-10 and TOP-100 apps through time.

Table 4.2 shows some statistics on the calculated sets which help as to understand their stability. In particular, we can see that the number of unique applications found in top-10 sets is slightly greater than 10 for each appstore. The only appstore that seems to deviate slightly from this behavior appears to be AppChina with 23 unique apps in top 10. This implies that the top-10 is sufficiently stable through time for all these marketplaces. A similar trend seems to exist for the top-100 set, as we see that the number of unique apps found in the calculated top-100 sets is a little higher than 100. To better understand the degree of this stability, we calculated the *set difference* between every two successive top-10 and top-100 sets. For example, suppose that the top-10 set of a given day is $A$ and this of the next day is $B$. The set difference of $A$ and $B$ is defined as the set of elements in $A$, but not in $B$:

$$A \setminus B = \{x \in A \,|\, x \notin B\} \tag{4.1}$$

We can see in Table 4.2, that the percentage of this difference in average, for both top-10 and top-100 sets, is very low for all the appstores and ranges from 0.3% to 4.7% for top-10, and from 0.3% to 1.7% for top-100, which indicates a very stable top-10 and top-100 set of apps through time. Although the top sets of most popular apps seem to be stable over time, it would be interesting to see whether there are rearrangements of apps within them. In order to investigate this possibility, we calculated the *Dice Similarity Coefficient (DSC)* between all consecutive top-10 and top-100 sets. The dice Dice Similarity Coefficient which first proposed by Dice [37], is a similarity

measure over sets, whose value ranges from 0, indicating that there is no spatial overlap between two sets, to 1, indicating a complete overlap. The formula for the calculation of the DSC is as follows:

$$s = \frac{2|A \cap B|}{|A| + |B|} \tag{4.2}$$

Table 4.2 shows that the average value of the dice coefficient for top-10 sets is equal to 1 for 1Mobile and 0.9 for SlideMe, which indicates that there is almost no rearrangements of the apps in these sets though time. For Anzhi appstore, this value is 0.8 which indicates a small number of rearrangements through time and in AppChina this number drops to 0.6, which suggests that there are constant changes in the positions of the 10 most popular applications. These changes seem to grow as we move to larger sets of apps. For instance, the average dice coefficient of the top-100 sets for each appstore has dropped by a percentage greater than or equal to 10%. Nevertheless, we can say that SlideMe and 1Mobile constituted of a kernel of popular apps that seems to be extremely stable through time, while Anzhi and AppChina include a stable set of popular apps with continuous changes in ranking order. This is better illustrated in Figure 4.11 which shows the CDF of Dice coefficient values of the calculated top-100 sets of apps among the monitored appstores. The ranges of CDF distribution for SlideMe and 1Mobile contain higher Dice coefficient values than those of the Chinese appstores, which indicates a more stable top-100 set of the former against the latter.

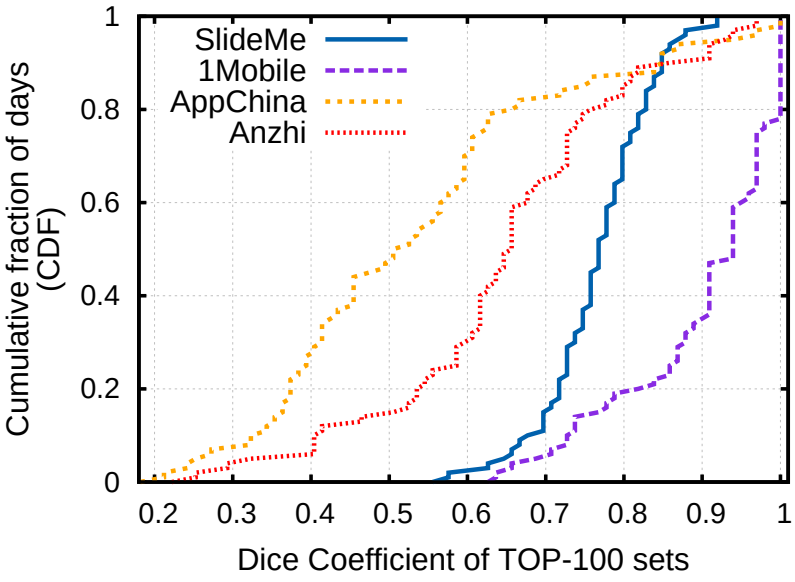



FIGURE 4.11: CDF of the different Dice Coefficient values between the calculated consecutive TOP-100 sets of apps.

## 4.6 Summary

In this section we attempted to understand the popularity of smartphone applications. We studied the app popularity based on the aggregate distribution of applications' downloads and we observed the presence of Pareto Principle in all of our datasets. That is, a small number of apps have a very large number of downloads, while the majority of applications has very limited downloads. Moreover, we show that app popularity in all of our datasets exhibit Zipf-like distributions with some deviations which are apparently caused by the "fetch-at-most-once" behavior of the users (as shown in other studies too). Moreover, we feel that these deviations are also caused by a more general phenomenon, which we call "the clustering effect" and is related with the grouping of apps into different categories. We will further explore the existence of this phenomenon in Chapter 5. Finally, we discussed which parameters may affect app popularity and we showed that there is a relationship between user ratings and app downloads. In addition, we studied the effect of pricing in app popularity and realized the app downloads' distribution varies from free to paid apps. Finally, we investigated the stability of most popular apps though time and noticed that top-10 and top-100 apps seem to be quite stable over time.

# 5

# User's Temporal Affinity to App Categories

In our next set of measurements we would like to explore whether the previously suggested clustering effect hypothesis can be validated. That is, we want to find out whether a user that downloaded an app of a particular category, will download another app from the *same* category with high probability. In other words, we are interested to find out if users tend to stay within a single category rather than switching to another one.

## 5.1 Temporal Affinity probability

Although we did not have access to appstore logs and could not know which apps were downloaded by each user, we used the comments accompanied by rating which users made as an indication of users' interests and ultimately as an indication of users' downloads. Of our appstores, only Anzhi supported comments with precise timestamps. We crawled the Anzhi appstore and recorded the stream of comments for each user (suppressing successive duplicates). For example, if a user commented on apps $a_1, a_2, a_3, a_3, a_1, a_4$ we kept the sequence $a_1, a_2, a_3, a_1, a_4$ which we call *app string*. In addition, we know that each app $a_i$ belongs to some category $c(a_i)$. Using this knowledge, we construct the string $c(a_1), c(a_2), c(a_3), c(a_1), c(a_4)$ which we will call *category string*. Given a category string of $n$ elements $c_1, c_2, c_3, ..., c_n$ with the respective categories of a user's $n$ comments in chronological order, we define the *temporal affinity* or *affinity probability* as the number of elements which are equal to their previous one over $n - 1$. The affinity probability is an indication of whether users tend to comment on apps from the same category or on apps from different categories. Indeed, if the temporal affinity

approaches the highest value (*i.e.* one), then users tend to comment on apps from one category only, indicating that the users may tend to download apps from only one category. On the contrary, if the temporal affinity is low, then users tend to switch from one category to another. Note, that if we have $C$ categories of roughly equal volume, then if users randomly wander from one category to another, the affinity probability will be around $1/C$.

Given a set with downloads (actually comments) of a user in chronological order, one can calculate the temporal affinity to app categories related with this user, which is the percentage of apps belonging to the same category as their previous one. For example, assuming a user that have downloaded (commented) $n$ apps of categories $c_1, c_2, c_3, ..., c_n$. To calculate the affinity probability of this user, we have to divide the above category string in $n-1$ consecutive pairs (sets of two categories). Then, affinity probability for this user can be calculated using the following formula:

$$P_A = \frac{\sum_{i=2}^{n} \text{Affinity}(c_i, c_{i-1})}{n-1} \tag{5.1}$$

where $\text{Affinity}(c_i, c_j)$ equals to one if $c_i$ and $c_j$ belong to the same category and zero otherwise. For example, a user with a category string $c_1, c_1, c_2, c_2$ will have affinity probability equal to 2/3. For example, given a user with category string $c_1, c_2, c_3$, temporal affinity related with this user will be 0. For a user with category string $c_1, c_1, c_3$, temporal affinity will be 0.5 (1/2). A user with $c_1, c_1, c_1$ will have temporal affinity equals to 1 (2/2). Moreover, for a user with category string $c_1, c_1, c_2, c_2$ the temporal affinity will be 0.66 (2/3), while for a user with this categories' series $c_1, c_2, c_1, c_2$, it will be 0.

As mentioned above, if the apps were evenly distributed among the different categories, then the affinity probability of a random wandering would be around $1/C$ (if $C$ is the total number of unique categories in the appstore). However, in practice, applications are not evenly distributed among different categories. Therefore, to calculate the actual affinity probability of a random wandering in Anzhi marketplace we used the actual distribution of apps to the $C$ different categories in this appstore from our collected data. Let $A$ be the total number of apps in the appstore and $A(i)$ the number of apps that belong to category $i$ (the sum of all $A(i)$ is equal to $A$). Given this distribution, the random walk affinity probability $P_{\text{random walk}}$, *i.e.*, the probability that two random app choices will belong to the same category, is equal to:

$$P_{\text{random walk}} = \frac{\sum_{i=1}^{C} A(i) \times (A(i)-1)}{A \times (A-1)} \tag{5.2}$$

since out of the $A \times (A-1)$ possible random app choices, the number of app choices for which the two apps belong to the same category are equal

to the sum of $A(i) \times (A(i)-1)$ for all categories ($\sum_{i=1}^{C} A(i) \times (A(i)-1)$). For instance, if a given appstore contains 10 apps ($A = 10$) with 3 categories ($C = 3$) and the following distribution of apps to categories: $|C1| = 3$ apps, $|C2| = 3$ apps, $|C3| = 4$ apps, then the random walk affinity probability will be $(3 \times 2 + 3 \times 2 + 4 \times 3)/(10 \times 9) = 0.266$. We need the affinity probability of a random wandering as a base case to compare with the actual affinity probability we measure in the Anzhi appstrore.

## 5.2 Temporal Affinity for Different Depth Levels

So far we have defined the temporal affinity as the probability of a user choosing two consecutive apps that belong to the same category. In this way, however, we miss users that may oscillate between few categories. For instance, the category string $c_1, c_2, c_1, c_2$ has temporal affinity equal to zero according to our previous definition, but we can see a clear affinity of this user to $c_1$ and $c_2$ categories. To address this issue, we define temporal affinity using the *depth* notion. That is, affinity probability $P_A$ of depth $d$ for a category string with $n$ elements is defined as the number of elements in category string for which at least one element among its previous $d$ elements belongs to the same category, divided by $n - d$:

$$P_A = \frac{\sum_{i=d}^{n} \text{Affinity}(c_i, \{c_{i-1}, c_{i-2}, ..., c_{i-d}\})}{n-d} \tag{5.3}$$

where $\text{Affinity}(c_i, \{c_{i-1}, c_{i-2}, ..., c_{i-d}\})$ is equal to one if at least one of the $\{c_{i-1}, c_{i-2}, ..., c_{i-d}\}$ belongs to the same category with $c_i$, and zero otherwise. For example, temporal affinity of depth equal to two is defined as the probability of a user selecting an app that belongs to the same category with one of the previous two selections. To calculate the temporal affinity for this depth, we first divide the $n$ apps of the category string into $n-2$ consecutive triplets (sets of three categories). Then, we check which of these triplets have affinity, *i.e.*, if the third element belongs to the same category with the second or with the first one in the triplet. Finally, we sum the triplets which have affinity and divide them with the total number of triplets, as shown in equation 5.3. In the same manner, to compute the temporal affinity probability of depth equal to three, we divide the user app choices in $n-3$ sets of 4 *etc.* A graphical example that shows the calculation of temporal affinity probability for different depth levels is illustrated in Figure 5.1. The equation 5.1, we defined in previous subsection, is the case of depth equal to one.

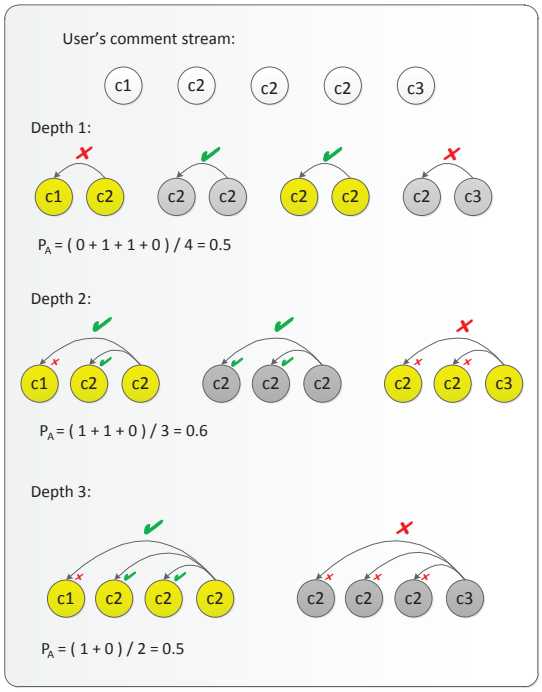



FIGURE 5.1: Example showing the calculation of affinity probability for different depth levels.

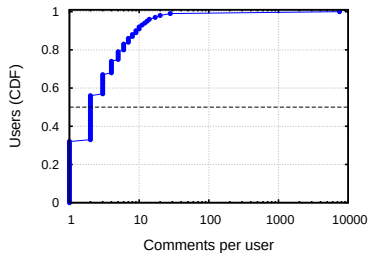
Using equation 5.2 we can find the affinity probability of a random wandering for depth equal to one. To calculate this probability for an arbitrary depth $d$ we should use the following equation:

$$P_{\text{random walk}} = \frac{\sum_{i=1}^{C} A(i) \times (A(i)-1) \times d \times \prod_{k=2}^{d} (A-k)}{\prod_{k=0}^{d} (A-k)} \tag{5.4}$$

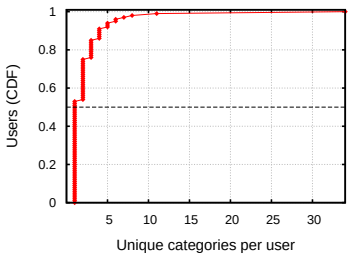
where each category $i$ has $A(i)$ apps, $A$ are the total apps, and $C$ the number of categories. For depth equal to $d$, all the possible app choices are $A \times (A-1) \times ... \times (A-d)$. The choices with affinity for each category $i$ are $A(i) \times (A(i)-1) \times d \times (A-2) \times ... \times (A-d)$ because $A(i)$ is the probability to select the last app from category $i$, $(A(i)-1) \times d$ is the probability that one of the previous $d$ apps belongs to same category $i$, and the rest $d-1$ apps can be selected from any category. Thus, equation 5.4 sums all the possible app choices which have affinity and divides them with all the possible ways

to select $d+1$ apps to compute the random walk affinity probability for any depth $d$. Equation 5.2, we saw in previous subsection, is the case with $d = 1$.

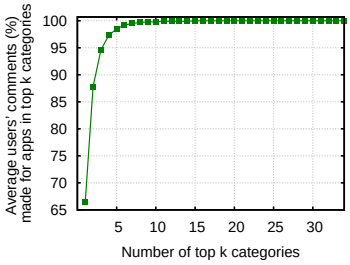
## 5.3 Results




(a) CDF of number of comments per user.




(b) CDF of unique categories per user.




(c) Average percentage of user comments made for the top $k$ categories, as a function of $k$.

FIGURE 5.2: Statistics on user comments.

The crawling process of user comments from Anzhi marketplace resulted in a dataset of 361,282 user comment streams to 60,196 apps in 34 categories. Figure 5.2(a) shows the distribution of the number of comments per user. We see that most users made up to 20 comments. There were few users with a very large number of comments in many different categories. We found that these users were posting spam, possibly using an automated script. Figure 5.2(b) shows the CDF of unique categories per each user that posted at least one comment. We see that 53% of the users commented on apps from a single category, and 94% of the users commented on apps from up to just 5 categories. Similarly, in Figure 5.2(c) we present the average percentage of user comments made for the top $k$ categories, as a function of $k$. We have excluded the users that made comments on a single app in this figure. We observe that the 66% of the comments of an average user were made for a single category, while 95% of the user comments were made for no more than 5 categories. These findings indicate that most users tend to

comment on apps from a single category or from very few categories of the 34 different categories we can find in Anzhi appstore.

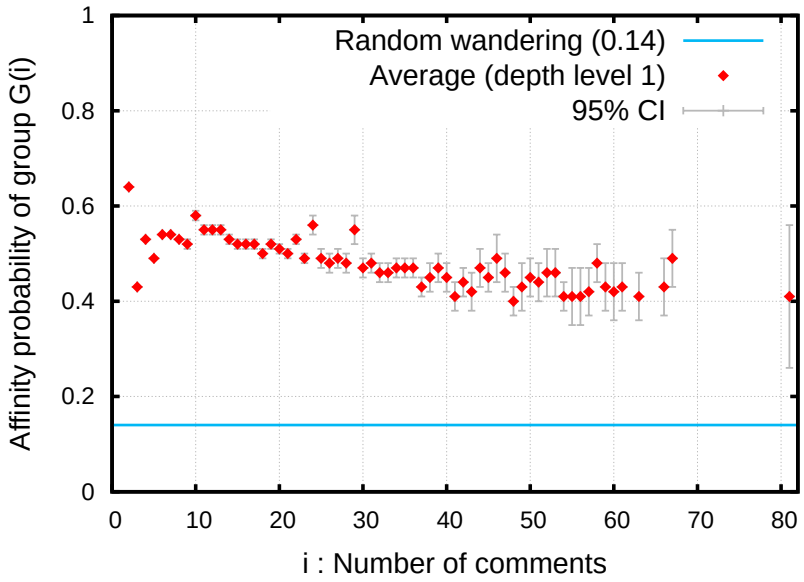
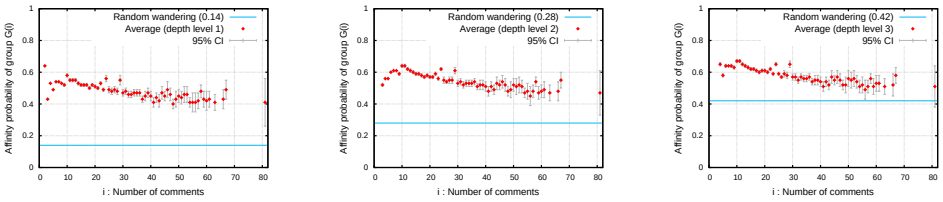



FIGURE 5.3: Affinity Probability for depth=1 of users grouped by their number of comments.

We see that the Affinity Probability ranges around 0.5. This implies that once a user comments on an app from one category, (s)he will comment on another app of the same category with probability close to 0.5.

However, the results from Figures 5.2(b) and 5.2(c) may be biased from the small number of comments made by most users. To overcome this issue we measure the affinity probability as defined in equation 5.3, based on the actual app distribution in the 34 categories of the Anzhi appstrore. Figure 5.3 shows the affinity probability for the users of Anzhi (for depth level equal to 1), as a function of the number of comments per user. We have grouped together all users that made the same number of comments, and we plot the average values and the $95^{th}$ confidence intervals from each group. We plotted only the groups that had more than 10 samples, excluding, in this way, the spam users as well.

We see that the average affinity probability for most groups is around 0.5. This implies that once a user comments on an app from one category, the same user will comment on another app of the same category with probability close to 0.5. To place these numbers in context, we also calculated the affinity probability of the base case as well, *i.e.*, the case where a user wanders from one app to another randomly. The random walk affinity probability on our dataset for depth one is 0.14, shown as a horizontal line in




(a) Depth=1 (b) Depth=2 (c) Depth=3

FIGURE 5.4: Affinity probability for depth levels from 1 up to 3 for users grouped by their number of comments.

The affinity probability increases with depth level, as it was expected.

Figure 5.3. Thus, we see that Anzhi users are 3.6 times more likely to stay in the same category compared to the base case of random wandering. This outcome indicates a strong affinity between users and categories.

To explore how this affinity probability increase for higher depth levels, we plot in Figures 5.4(b) and 5.4(c) the affinity probability of the same user groups, as well as the random walk affinity probability, for depths equal to two and three respectively. We see that affinity probability increases with depth, as it was expected, and remains higher than the respective random walk affinity probability. To explore how the affinity probability varies among users, we plot the cumulative distribution of affinity probability among all Anzhi users for the three different levels in Figure 5.5. We

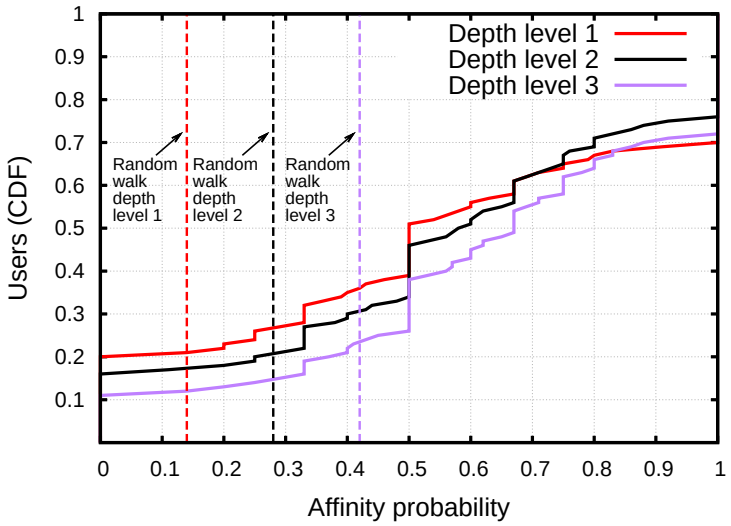



FIGURE 5.5: CDF of affinity probability for different depth levels.

observe that the median value for depth one is 0.5, while for depths two and three the median values are 0.58 and 0.67 respectively. The results of this

| | Affinity Probability | |
|---|---|---|
| **Depth level** | **Users' App Choices (Mean)** | **Random Wandering** |
| 1 | 0.55 | 0.14 |
| 2 | 0.58 | 0.28 |
| 3 | 0.67 | 0.42 |

TABLE 5.1: Affinity probability of users' app choices vs affinity probability of random wandering, for different depth levels

Figure 5.5 are also summarized in table 5.1, which provides a comparison between average affinity probability observed in the users of our dataset and the affinity probability of random wandering, for the three levels of depth. As mentioned previously, the affinity probability of users' app selections is growing together with depth level. The same behaviour is also observed for random wandering affinity probability, which appears to grow at a faster pace. Nonetheless, the probability of a user to choose an app that belongs to the same category of her previous choices seems to be greater than this of a random wandering.

Overall, our results show that there is a strong affinity of users' comments to app categories, which is a strong indication that users tend to download apps from the same categories. These observations validate our hypothesis about the app clustering effect in user downloads, as described in section 4.2.

## 5.4 Summary

Briefly, we attempted to validate our previously suggested clustering effect hypothesis, which proposes that users tend to download several apps from a single category rather than switching to another one. To do so, we suggested a new metric called user temporal affinity that measures the affinity of users to app categories. We performed measurements on a dataset of user comments with ratings (which implies downloads), and we show that users are 3.6 times more likely to stay in the same category compared to the base case of random wandering. Our findings show that there is a strong affinity of users' comments to app categories which validates our hypothesis about the clustering effect.

# 6

# A Model of Appstore Workloads

In Chapter 4.2 we hypothesized that the non-ZIPF behavior of smartphone app downloads can be explained by combining the fetch-at-most-once behavior of smartphone users with the *clustering effect* in user downloads. Then, in chapter 5 we validated our intuition that app clustering has a significant effect on user comments, and thus, on user downloads as well. This chapter presents a novel model of appstore usage (APP-CLUSTERING model), based on both fetch-at-most-once and clustering effect properties, that enables us to further explore our hypothesis. First, we describe the proposed model and its key parameters, and then we vary these parameters using a Monte-Carlo simulator to approximate the observed distribution of app downloads and validate our APP-CLUSTERING model.

## 6.1 Model Description and Analysis

Table 6.1 summarizes the key parameters used in our model. Note that we have no parameter related with time in our model: we demonstrate that the model is valid for every snapshot of our appstores. The number of apps $A$ is derived directly from the appstore's snapshot. Similarly, we can find the total number of downloads $D$ for all apps in the appstore until this date. If we set the number of users $U$ in our model, then the average number of downloads per user $d$ is $d = D/U$. We set the number of clusters $C$ to be equal to the actual number of app categories in the modeled appstore. We found through simulations that indeed this number of clusters give the best approximation of the actual downloads.

| Symbol | Parameter Description |
|---|---|
| $A$ | Number of apps |
| $U$ | Number of users |
| $D$ | Total downloads |
| $d$ | Downloads per user |
| $z_r$ | ZIPF exponent for generic app ranking |
| $C$ | Number of clusters |
| $p$ | Percentage of downloads based on the clustering effect |
| $z_c$ | ZIPF exponent for cluster's app ranking |
| $D(i,j)$ | Predicted downloads for app with total rank $i$ and rank $j$ in its cluster |

TABLE 6.1: APP-CLUSTERING model parameters and notation.

In our APP-CLUSTERING model, each app has a unique rank $i$ from 1 to $A$. Moreover, apps are distributed to the $C$ clusters in a way that an app belongs to exactly one cluster. Thus, each app has also a second rank $j$ compared with the other apps in the same cluster, based on its overall rank $i$ and the overall ranking of the other apps in this cluster. Then, each user randomly selects and downloads $d$ apps. Apps are selected for downloading by the users based on an overall ZIPF distribution, using their rank $i$ and the $z_r$ ZIPF exponent, with two exceptions: (i) *fetch-at-most-one*, *i.e.*, each user downloads the same app at most once, and (ii) *clustering-effect*, *i.e.*, once an app $a$ is downloaded, the user subsequently downloads a number of apps which belong to the $a$'s cluster. In the latter case, apps from this cluster are also selected based on a ZIPF-based distribution, using the apps' rank $j$ in this cluster and the ZIPF exponent $z_c$. The parameter $p$ in our model defines the percentage of the user downloads that are selected based on the clustering effect, while the first $(1-p) \times d$ app downloads of each user are selected using the overall ZIPF distribution. Based on these parameters, the APP-CLUSTERING model estimates the total number of downloads $D(i,j)$ for each app with overall rank $i$ and rank $j$ in this cluster.

We believe that the main abstraction introduced by our approach with respect to the previous download models is the abstraction of a cluster. The cluster may be used to model (i) app categories, (ii) recommendation systems, (iii) user groups, and other possible grouping forces. For example, several appstores categorize apps into different groups: games, wallpapers, ebooks, utilities, lifestyle, and other. Once a user downloads an app from a particular category, this user may stay in the same category a bit longer before switching to another category.

**Analysis.** We assume an appstore with $U$ users, where each user downloads $d$ apps. For simplicity we assume that all users download the same number of apps. Out of the total $D$ downloads, $(1-p) \times D$ are app selections based on a pure ZIPF distribution. That is, an app with rank $i$ has a probability to be selected for downloading equal to $1/i^{z_r}$. Similarly, when apps are selected from a specific cluster, an app with rank $j$ in this cluster will be

selected with probability equal to $1/j^{z_c}$. Overall, the expected downloads $D(i,j)$ for an app with overall rank $i$ and rank $j$ in its respective cluster with the APP-CLUSTERING model are:

$$D(i,j) = \sum_{u=1}^{U} 1 - \left(1 - \frac{1/i^{z_r}}{\sum_{k=1}^{A} 1/k^{z_r}}\right)^{(1-p)\times d} \times \left(1 - \frac{1/j^{z_c}}{\sum_{l=1}^{S_C} 1/l^{z_c}}\right)^{p\times d} \quad (6.1)$$

The expected downloads per each app are estimated by adding the probability of all users to download this app. The probability of a single user to download this app is one minus the probability for not downloading this app with $(1-p)\times d$ ZIPF-based selections and $p \times d$ clustering-based selections. We see that the number of downloads for very popular apps are limited by the number of users $U$. We found that the results from analytical evaluation, based on Equation 6.1, are very close to the simulation results for the same parameters, so in the next section we present only the simulation-based results.

## 6.2 Simulation-based Model Validation

To evaluate our model and understand the impact of clustering effect in the app downloads distribution, we developed three Monte-Carlo simulators of an appstore, using ZIPF, ZIPF-at-most-once, and APP-CLUSTERING models. In the ZIPF simulator we set the number of users $U$ equal to the number of total downloads $D$, which results to $D$ independent random app selections. In the ZIPF-at-most-once simulator we set the number of users to a proper value, and each user randomly selects $d$ apps for downloading. However, the same app can not be selected more than once by the same user. For both ZIPF and ZIPF-at-most-once simulators we group all apps in a single cluster, and thus all app selections are made based on a pure ZIPF distribution with the $z_r$ exponent. In the APP-CLUSTERING simulator we randomly group the $A$ apps to $C$ clusters, and each user downloads $d$ apps based on the simulation parameters $z_r$, $p$ and $z_c$.

Then we ran these simulators for all appstores in our dataset while varying their key parameters, in order to approximate the observed distribution of app downloads as close as possible. To measure how close each simulation approaches the actual downloads distribution of $A$ apps we calculate the distance between the observed and simulated downloads of each app using the mean relative error:

$$distance = \frac{1}{A}\sum_{i=1}^{A} \frac{|D_o(i) - D_s(i)|}{D_o(i)} \quad (6.2)$$

where $D_o(i)$ and $D_s(i)$ are the observed and simulated downloads respectively for the app with overall rank $i$.

## 6.3 Choosing the Right Number of Users

In our first set of simulations we would like to explore how the number of users $U$ influences the simulation results. Unfortunately, since we do not have access to the logs of the appstores we do not know the actual number of users who have accessed each appstore. We know the total number of downloads, the total number of apps, but we do not know the total number of users. Nevertheless, we will conduct simulations in order to explore whether there is a reasonable range for the number of users in the appstores studied that results to very close approximations of the actual downloads distribution per app. Given that different appstores have a different actual number of users, we express the number of users as a function of the total number of downloads of the most popular app.

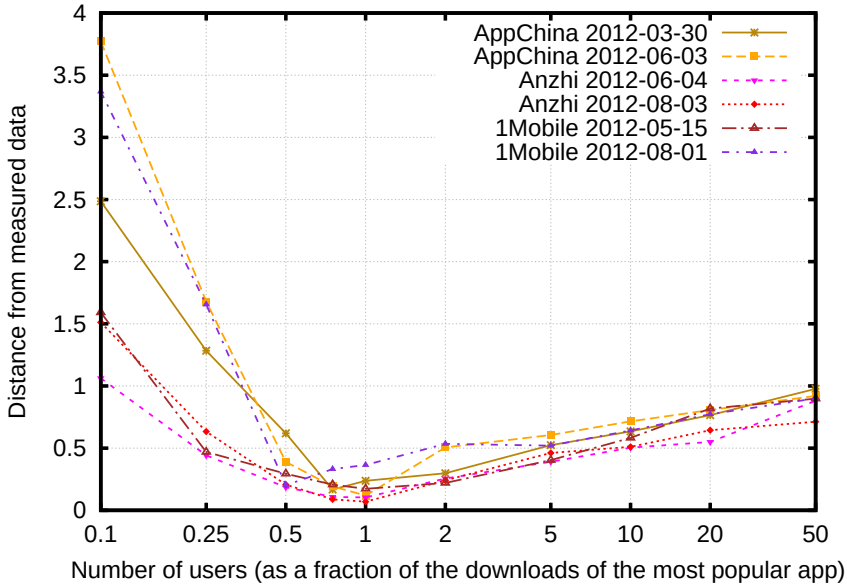



FIGURE 6.1: Effect of the number of users on the accuracy of the simulation. We see that we have the minimum distance from the actual downloads when the number of users is close to the downloads of the most popular app.

Figure 6.1 shows our simulation results using the APP-CLUSTERING model while varying the number of users $U$ and setting the rest simulation parameters to the values that produce the minimum distance from the actual results for each dataset. The $x$-axis is the number of users simulated, as a ratio of the total number of downloads of the most popular app, and the $y$-axis reports the distance between the simulation results and the measured downloads, using the mean relative error as shown in equation 6.2. We plot the simulation results for the app downloads of the first and last day of the AppChina, Anzhi, and 1Mobile appstores. We see that in all cases the

minimum distance from the measured data is achieved when the number of users is very close to the number of downloads of the most popular app. Thus, in the subsequent simulations we set the number of users equal to the downloads of the most popular app for each simulated dataset, which seems like a good approximation of the actual number of users.

## 6.4 Comparing Modeled and Actual Downloads

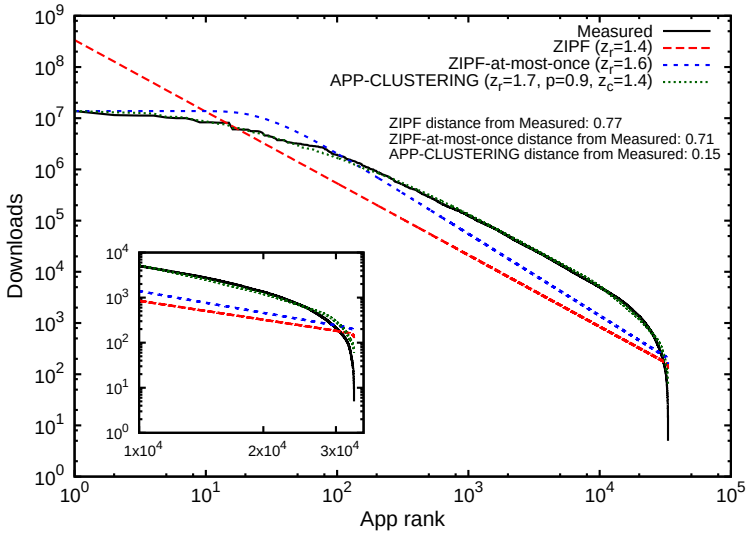
In this section, we compare the simulation results of APP-CLUSTERING, ZIPF and ZIPF-at-most-once models with the measured downloads as a function of app ranking, for the first and last day of our crawling period in AppChina, Anzhi and 1Mobile. The results of AppChina are summarized in Figure 6.2, while the results of Anzhi and 1Mobile are illustrated in Figure 6.3 and Figure 6.4 respectively. We plot the results of each model using the parameters that produced the minimum distance from actual results. We see that a pure ZIPF distribution clearly deviates from the measured data. For example, for small $x$ (*i.e.*, popular apps) it overshoots the measured data by more than an order of magnitude. ZIPF-at-most-once fits the data better but deviates from the actual data mainly for large $x$ values. On the other hand, APP-CLUSTERING matches the data very close, better than any other model, both for large $x$ and for small $x$ values (*i.e.*, least popular apps). Indeed, we see that APP-CLUSTERING has the smallest distance from the measured data for all appstores. For instance, in the AppChina dataset of the first day (Figure 6.2(a)), APP-CLUSTERING predictions result to distance 0.15 from the actual downloads, that is a factor or 4.7 improvement over ZIPF-at-most-once and a factor of 5.1 improvement over ZIPF. Similar results are seen for the simulations of the other appstores as well.

| | **Distance from Measured** | | |
|---|---|---|---|
| **Dataset** | **ZIPF** | **ZIPF-at-most-once** | **APP-CLUSTERING** |
| AppChina 2012-03-30 | 0.77 | 0.71 | 0.15 |
| AppChina 2012-06-03 | 0.79 | 0.70 | 0.18 |
| Anzhi 2012-06-04 | 0.36 | 0.32 | 0.05 |
| Anzhi 2012-08-03 | 0.30 | 0.19 | 0.07 |
| 1Mobile 2012-05-15 | 0.49 | 0.49 | 0.16 |
| 1Mobile 2012-08-01 | 0.74 | 0.72 | 0.28 |

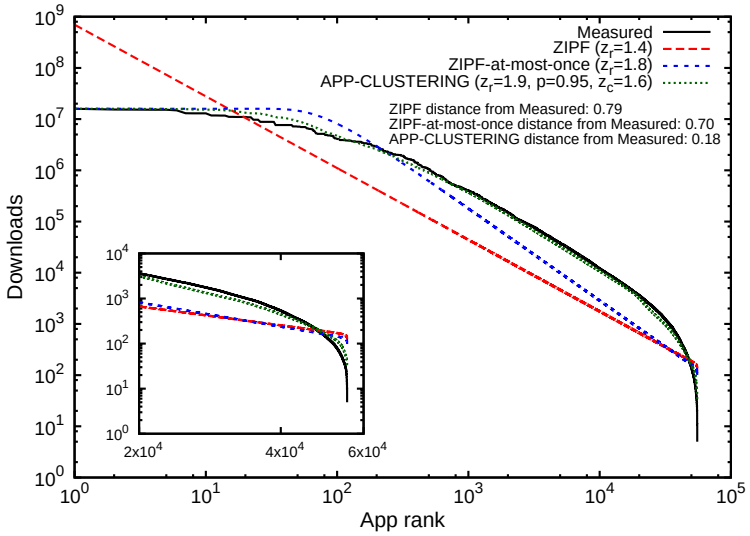
TABLE 6.2: Distances of the different models from Measured data.

Moreover, we observe that our APP-CLUSTERING model is able to approximate the actual results of an appstore very well from the first day up to the last day of our measurement period. We see that the best approxi-

mations of the actual data are achieved when the percentage of simulated downloads based on clustering is 90% and 95%. This outcome shows the great extent to which clustering actually affects the app downloads distribution. Moreover, the distance of each model from the measured data shows that APP-CLUSTERING is able to approximate the actual downloads up to 7.2 times closer than ZIPF and up to 6.4 times closer than ZIPF-at-most-once. An overview of distances of the different models from the actual data is presented in Table 6.2, as well as in Figure 6.5.



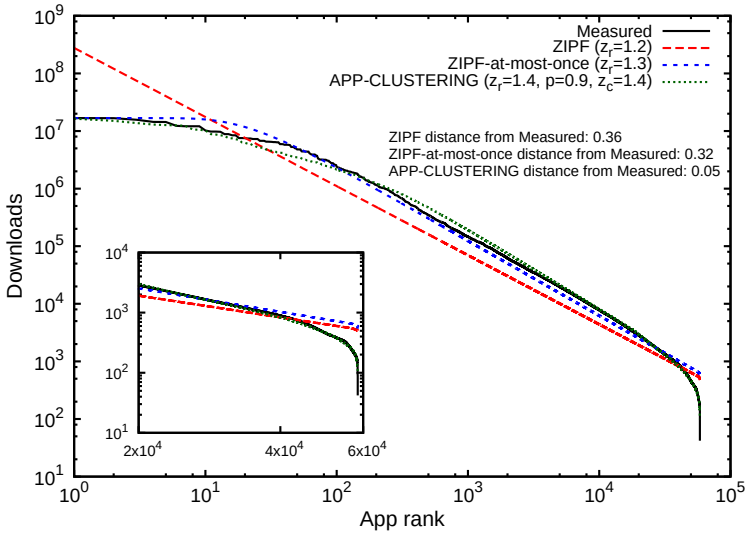

(a) AppChina 2012-03-30



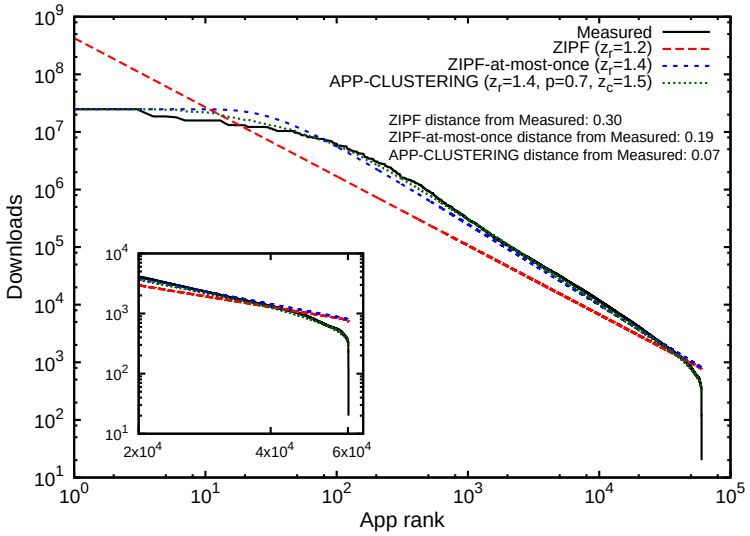

(b) AppChina 2012-06-03

FIGURE 6.2: Predicted versus measured app popularity of AppChina appstore for two different days.

We see that APP-CLUSTERING fits very close the measured data. ZIPF-at-most-once fits the data better than a pure ZIPF distribution, but diverges for large $x$ values.



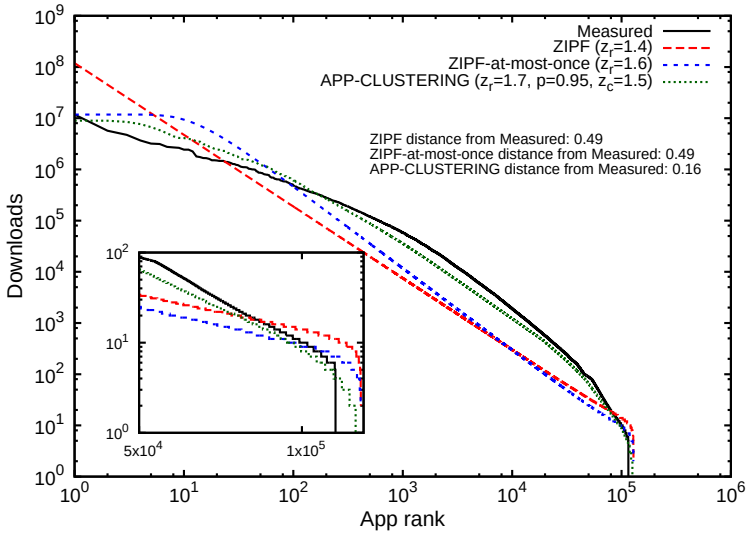

(a) Anzhi 2012-06-04



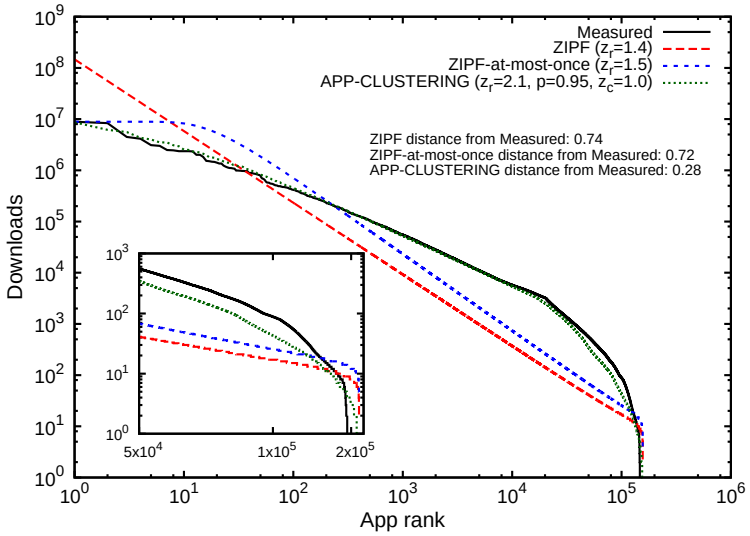

(b) Anzhi 2012-08-03

FIGURE 6.3: Predicted versus measured app popularity of Anzhi appstore for two different days.

We see that APP-CLUSTERING fits very close the measured data. ZIPF-at-most-once fits the data better than a pure ZIPF distribution, but diverges for large $x$ values.




(a) 1Mobile 2012-05-15




(b) 1Mobile 2012-08-01

FIGURE 6.4: Predicted versus measured app popularity of 1Mobile appstore for two different days.

We see that APP-CLUSTERING fits very close the measured data. ZIPF-at-most-once fits the data better than a pure ZIPF distribution, but diverges for large $x$ values.

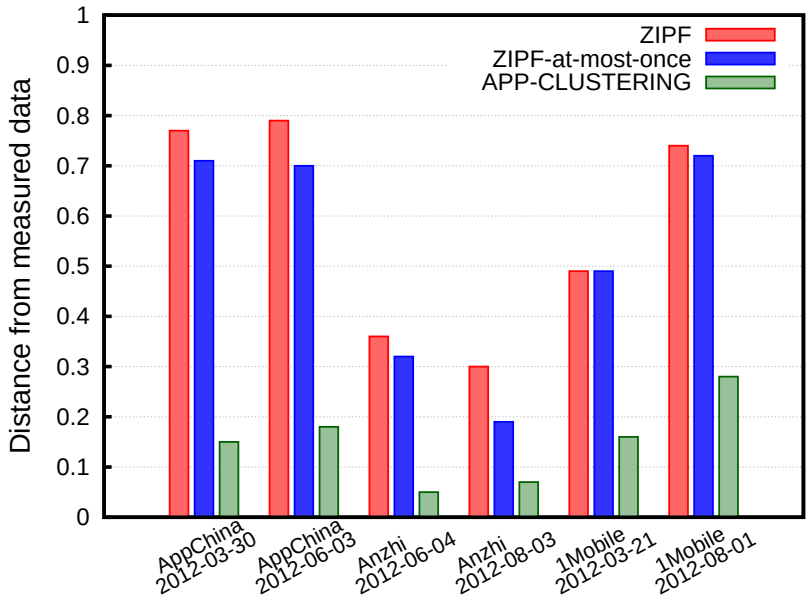



FIGURE 6.5: Comparison of different models distances from measured data.

*APP-CLUSTERING model has the smallest distance from measured data.* The distance of each model from the measured data shows that APP-CLUSTERING is able to approximate the actual downloads up to 7.2 times closer than ZIPF and up to 6.4 times closer than ZIPF-at-most-once.

## 6.5 Summary

To summarize, in this chapter we present a novel model of appstore workloads based on fetch-at-most-once property and our suggested clustering effect. We describe our APP-CLUSTERING model and we discuss its key parameters. Then, we validate our model through a series of simulations attempting to approximate the actual distribution of app downloads in the appstores of our dataset. Our findings show that the combination of fetch-at-most-once behavior with the clustering effect can approximate more closely the actual data, which implies that this combination is the reason why aggregate popularity of apps deviate substantially from Zipf behavior that has been observed in the Web [31]. Moreover, we believe that such a model will be helpful for appstore owners to estimate the app popularity (downloads distributions) of their hosted apps, as well as will help them to pinpoint "problematic" applications, *e.g.* those that do not seem to attract the interest of the users, and either help them through recommendations to have a chance of gaining popularity (downloads) or remove them completely from their marketplace.

# 7
# App Pricing

In this chapter we present a detailed study of the role of pricing in smartphone applications. For this purpose, we are using data collected from SlideMe marketplace (the only one of our monitored appstores that contains paid apps), and we are focusing on several aspects such as: the income of paid apps, how this income is distributed across different app categories and across developers, which are developers' trends: Do they target at a limited number of app categories (user audiences)? Do They appear to care about the *quality* or the *quantity* of their products? Furthermore, we try to answer the question: Can free apps make higher income than paid apps? To do so, we estimate the average income that a free app should produce from other means such as through *ads* (advertisements) or *in-app-billing*, so that to earn the same amount of money as the average income of a paid app.

## 7.1 The Developers

At first, we present the total number of developers that produce apps in SlideMe appstore and how they are distributed across paid and free apps. In Table 7.1, we can see that there are 5,106 developers in total, which is equal to one fifth of the number of available apps in the marketplace. We also observe that the majority of the developers (87,3%) produce free apps, while a small fraction (about 25.7%) of them deal with free apps. Notice that these percentages sum up to a higher value than 100%, as there are developers that produce both paid and no-cost apps. These developers account for 13% of the total number (about 664) of developers. Besides,

| Set | Number of apps | Percentage of apps | Number of developers | Percentage of developers |
|---|---|---|---|---|
| Free | 16,578 | 74.7% | 4,456 | 87.3% |
| Paid | 5,606 | 25.3% | 1,404 | 25.7% |
| Total | 22,184 | 100% | 5,106 | 100% |

TABLE 7.1: Percentage of developers across free and paid apps.

there is an equivalent proportion of developers (14.5%) dealing entirely with paid applications development.

Next, we are interested in finding how many apps are made by each developer. Figure 7.1 shows the CDF of the number of free and paid apps produced by developers of SlideMe appstore. Note that cumulative fraction of free apps has been calculated based on the set of developers that produced free apps, and not on the entire number of developers in the marketplace. Similarly, we computed the CDF of paid apps per developer. We can see that most of developers have produced only a single app. Particularly, almost 60% of developers that focus on free apps and about 70% of developers that focus on paid apps have only one application hosted in the marketplace. Moreover, developers that have made from 2 to 10 apps correspond to the second largest percentage, which is 35% for the free app developers and about 27% for the paid app developers. The small upper fraction in the

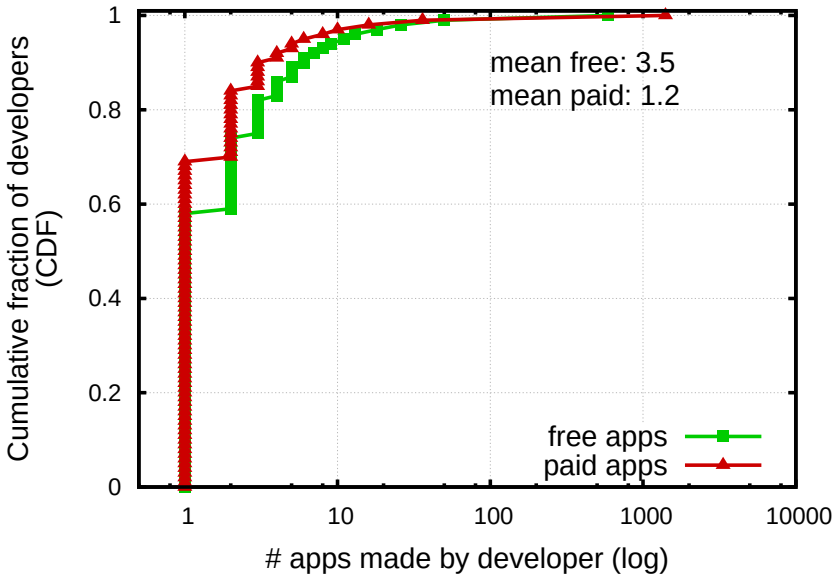



FIGURE 7.1: CDF of number of free and paid apps made by developers in SlideMe marketplace.

graph, that is 4% of free app developers and 2% of paid app developers have created a number of apps ranges from 11 to 50 and 16 to 36 respectively. Surprisingly, we see that 1% of free app developers account for 592 apps, while the same portion for paid app developers have made 1402 apps. By analysing our data we found that these two cases correspond to two different companies. Specifically, the 1402 paid apps were E-books applications that belong to *SmartEbook.com*[1] company (formerly known as *For-side.com*). SmartEbook.com is a Japanese mobile content provider company with operations primarily in Asia and North America, focusing on providing E-books worldwide. Furthermore, the 592 free apps belong to *Tristit*[2], an independent global mobile applications developer company with over 12,000 app titles in stock.

Having a vision of the number of apps per developer, it would be interesting to see what is the portion of the free and paid apps made across the whole population of app developers in SlideMe appstore. Figure 7.2 shows the CDF of percentage of free and paid apps made by developers. Note that the two distributions in the graph are symmetric to each other. We observe that only 12% of developers (upper 12% in the graph) produce only paid

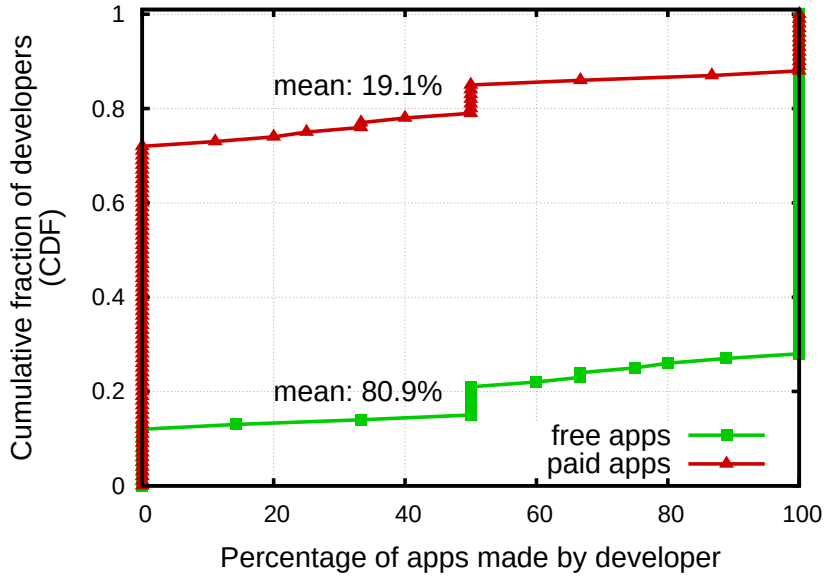


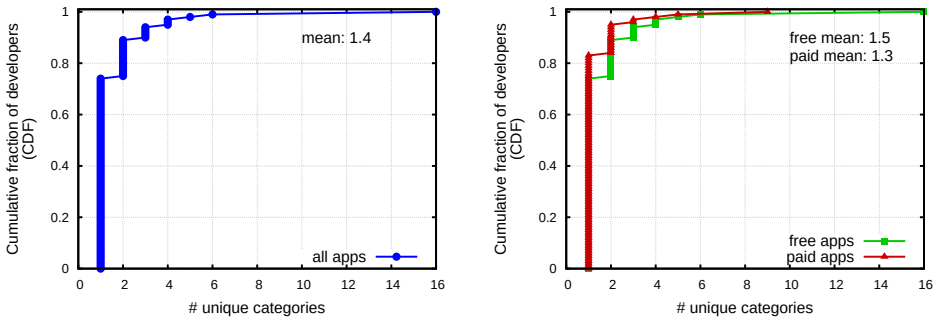

FIGURE 7.2: CDF of percentage of free and paid apps made by developers in SlideMe marketplace.

apps, where 72% of developers produce only free apps. From the sparse values along $x$ axis we can see that there are about 14% of apps developers that

[1] http://www.smartebook.com/eng/index.html
[2] http://tristit.com/

have made both free and paid apps. Those who have evenly spread rates of free and paid apps constitute about 7% of all developers. This leads to the conclusion that *developers seem to have a common strategy, and clearly prefer to offer either only free or only paid apps (mainly free).*

To understand the strategies followed by developers in the marketplaces we would like to see whether they focus on one or more categories. Figure 7.3 shows the CDF of *unique* categories of the apps made per developer. we can see that one fourth (75%) of developers focus on one category and 90% of them focus on one or two categories. Due to the fact that most developers of our dataset (SlideMe data) have made only one application, it is obvious to see most of them to focus on one category too. To get a more clear view

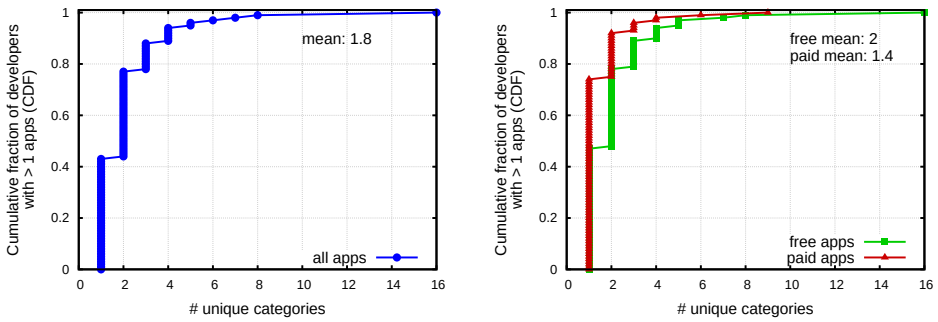



(a) All (b) Free vs paid

FIGURE 7.3: CDF of app categories per developer in SlideMe appstore.

of developers' preferences for app categories, we replot the CDF graph in Figure 7.4, only for developers that made *at least two apps.* The results are very similar with the previous plot. We can see that the 77% of developers prefer to focus on one or two unique categories.This conclusion appears stronger for the developers of the paid apps, as 75% of them have made apps belonging to one single category and 80% of them aim only to one or two categories.

In addition, we were curious about how the amount of developers is distributed among different categories of applications in SlideMe appstore. Figure 7.5 displays the total number of developers per app category, as well as the number of developers associated with free and paid apps separately. The relative number of these 3 sets of developers seem to be the same for each category. We see that the top app categories in terms of number of developers are *games* (at the highest position), followed by *utilities* and *entertainment*, with an almost similar developers' crowd, and then follow other categories like *educational, productivity, lifestyle, communications etc.* Overall, we observe that almost *half of the developers focus on a single category (games).* Moreover, the majority of them (about 85%) make apps for only 3 out of 20 categories (games, utilities and educational).




(a) All (b) Free vs paid

FIGURE 7.4: CDF of app categories per developer that made more than 1 apps in SlideMe appstore.

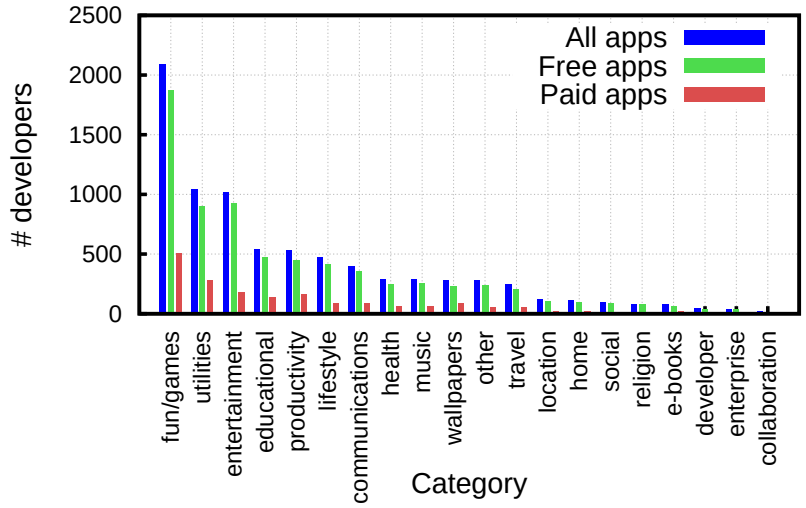
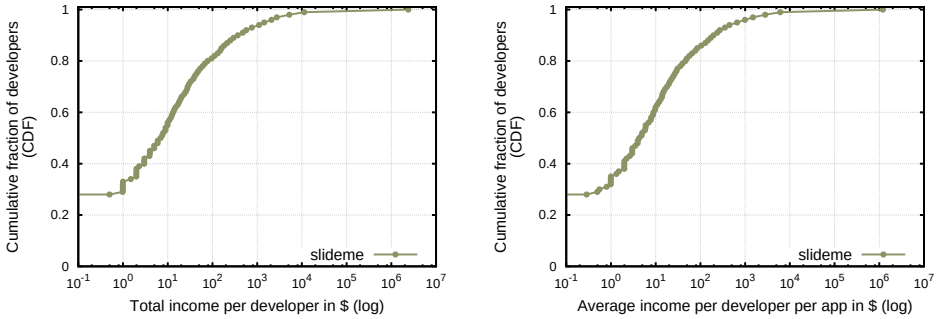



FIGURE 7.5: Number of developers per category in SlideMe marketplace.

## 7.2 Income per Developer

In this section, we focus our interest on the income that is gained by developers in SlideMe marketplace. To compute the income for each developer we relied on the total number of downloads (purchases) of all the paid applications during the measurements' period, as well as on their price. Figure 7.6 depicts the total income per developer along with the average income per developer per app. The two plots don't seem to have significant differences, as the number of apps of each developer is relatively small, as shown in the previous section. For this reason we will focus to the graph of Figure 7.6(a).




(a) Total income (b) Average income

FIGURE 7.6: CDF of total and average income per developer in SlideMe appstore.

According to the graph, 32% of the developers have not earned any income. In general, almost 60% of developers have gained from 0 to 12 dollars. Then there is a portion of developers, about 25% of them (from 60% to 85% in the graph), that have income ranges from 12 to 150 dollars, and a smaller portion, approximately 10% of them (from 85% to 95% in the graph) earned income between 150 and 1,500 dollars. The next 4% of developers have made funds from 1,500 to 11,260 dollars, where *1% of the total number of developers (upper 1% in the graph), have gained up to 2,400,000 dollars*! This tremendous amount of income apparently come from creators of the most popular applications out there. To obtain a clearer picture of this elite of

| Developer | Total income ($) | Number of apps |
|---|---|---|
| Hting | 2,400,294 | 2 |
| Yongzh | 674,663 | 1 |
| MobiSystems | 64,480 | 376 |
| AuroraSoftworks | 59,210 | 2 |
| Ramzixp | 50,296 | 3 |
| OneStepAhead | 25,135 | 81 |
| GuidedWays | 19,816 | 1 |
| FlyerIndustries | 19,806 | 2 |
| Cdeguet | 17,922 | 1 |
| Mantano | 15,822 | 2 |

TABLE 7.2: Top developers in terms of highest income.

developers, we listed the top-10 of them based on their income in Table 7.2. Table shows that the top 2 positions belong to developer *Hting* and *Yongzh*. *Hting*[3] actually is the developer of a very popular YouTube downloader

[3] `http://slideme.org/applications?text=hting`

app, while *Yongzh*[4] is a developer of game consoles' emulators software with her most popular app to be an emulator of *Nintendo 64* console optimized for the Android platform. In the rest top-10 of developers, we can distinguish several companies. For instance, *MobiSystems*[5], a multi-device mobile application development providing office and dictionary solution apps, *Aurora Softworks*[6], a company specializing in benchmark tools for mobile devices, *OneStepAhead*[7] that provides geospatial technologies and web solutions apps, as well as *FlyerIndustries* providing email and social network apps (*e.g.* a very popular *Facebook* client).

As we can see, the total income does not seem to be proportional to the number of applications of developer, if we exclude the cases of *MobiSystems* and *OneStepAhead*. To confirm this, we calculated the Pearson's correlation coefficient between the number of paid apps of each developer and her total income. We found that the coefficient value was 0.008, that does not imply a relationship between these two quantities. This is very interesting, as it implies that *quality is more important than quantity for developers income*. An illustration of this disassociation is depicted in Figure 7.7 where $x$ axis contains the average total income of the developers binned by number of apps, and $y$ axis includes the different app amount bins. As we can see, the graph does not show any obvious pattern.

## 7.3 Income per Category

Having studied the income of developers, we now focus on the total income of apps across different categories. Figure 7.8 shows the total income of apps for the different app categories along with the number of available apps in these categories.

[4] `http://slideme.org/user/yongzh`
[5] `http://www.mobisystems.com/`
[6] `http://www.aurorasoftworks.com/`
[7] `http://www.onestepahead.de/`

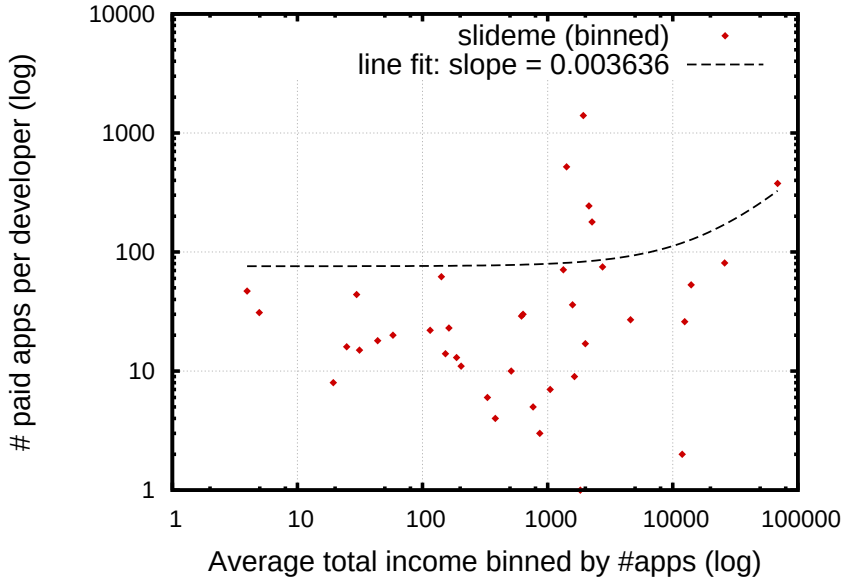



FIGURE 7.7: Number of paid apps vs income per developer in SlideMe marketplace.

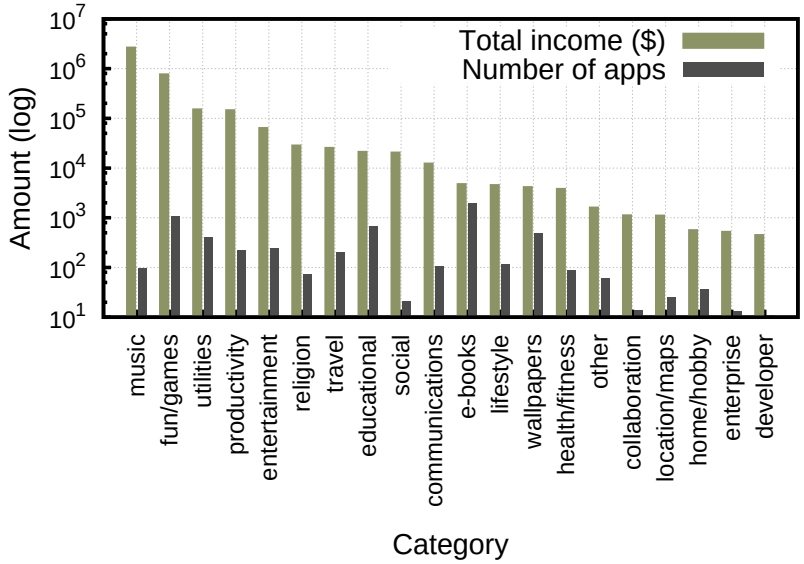



FIGURE 7.8: Total income per app category in SlideMe marketplace.

As we see from the graph, there is no correlation between the number of apps that a category contains and the income gained from the apps of the respective category. In other words, the fact that a category includes a large

portion of the total number of apps in the appstore, does not imply that this category will gain a large portion of the total income of all apps. The top category in terms of income seems to be *music*, followed by *fun/games* along with *utilities* and *productivity*, that look to have same levels of income. We attempt to see if the same view also exists for the average income per app per category. This result is shown in Figure 7.9. We kept the categories

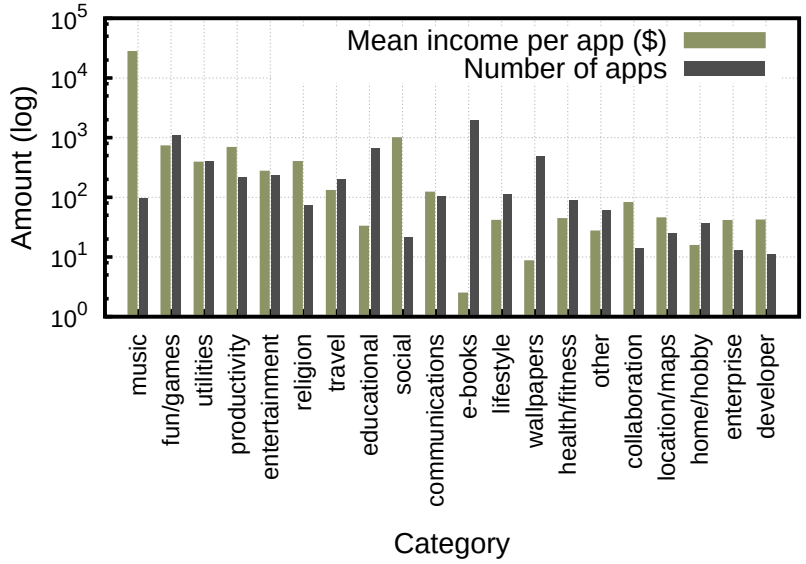
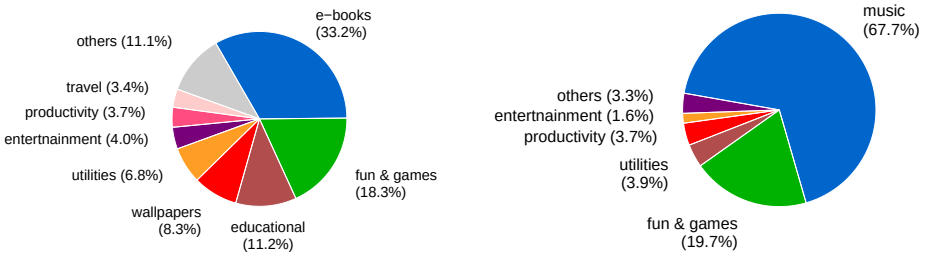



FIGURE 7.9: Average income per app category per app in SlideMe marketplace.

in $x$ axis ordered as in the previous Figure 7.8 for an easier side by side comparison. Comparing the two graphs (of Figures 7.9 and 7.8), there is not an obvious connection between the total income of apps per category with the average income per category. Moreover, the previous conclusion that the income is not correlated with the population of apps in a category applies here too. In order to have a better view of this inference, we illustrate the distribution of percentages of apps in the different categories along with the distribution of percentages of income across those categories in Figure 7.10. The figure clearly shows that although category E-books contains the highest percentage of apps (33% Figure 7.10(a)), has not proportionately the largest percentage of the total income. The category with the largest portion of income is music (that have the 67% of income, as we see in Figure 7.10(b)) and similarly music is not included in the categories with the larger number of apps. The same analogy applies also to the rest categories as shown in the graph, except the *games* category which seems to have an equal share of income and number of apps.




(a) Apps (b) Income

FIGURE 7.10: Distribution of percentages of apps and of in SlideMe appstore.

## 7.4 Can Free Apps Make Higher Income Than Paid Apps?

In this section we attempt to explore if there is a possibility for free apps to make higher income volume than paid apps. First, we provide some information on how apps are distributed across the different categories. Figure 7.11 shows which portion of apps belongs to paid apps and to free apps per category. We see that free apps dominate in all categories except from *collaboration* and *E-books*.

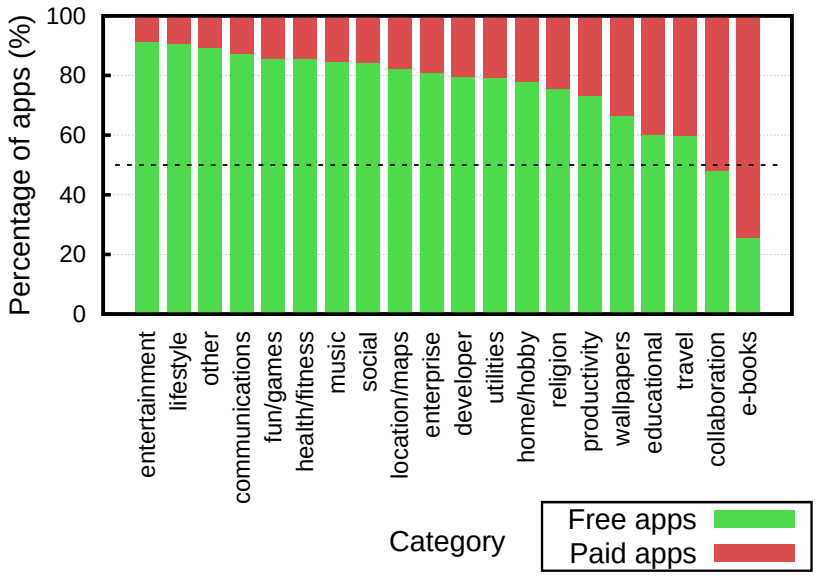



FIGURE 7.11: Percentages of free and paid apps across categories in SlideMe marketplace.

Then, we we examine whether the relative percentages of apps across categories are similar to unpaid ones. The results are shown in the Figure 7.12. We can see that the set of the 4 dominant categories of paid apps, which are *E-books, fun/games, educational and wallpapers*, is different from this one of free apps containing *fun/games, entertainment, utilities and lifestyle.*

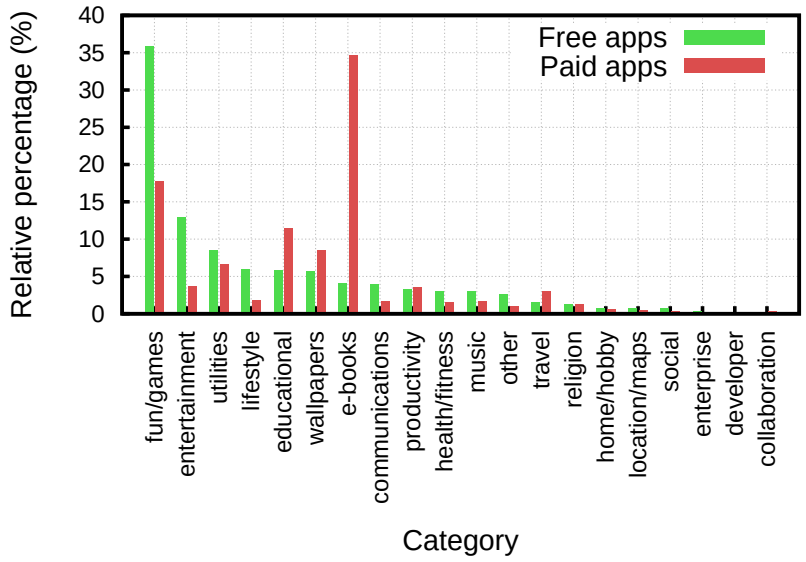



FIGURE 7.12: Relative percentages of free and paid apps across categories in SlideMe marketplace.

Afterwards we focus our interest in the income related with the free apps. Generally, several adverse incidents related with the use of advertisements in mobile applications have been reported, *e.g.* many ads that integrate advertisements seem to pose privacy and security risks [8, 45]. Moreover, a recent study [50] demonstrated that the 65%-75% of energy in free apps is spent in third-party advertisement modules. One major advantage of paid over free apps is that the former rarely use advertisements, while for the later, advertisements or simple *ads* constitute the main source of income. In order to compute the income of a free app that uses ads, the *Click-through Rate* is needed. *Click-through Rate* is given from the following formula:

$$CTR = \frac{Clicks}{Impressions}$$

The *Impressions* is the number of times an ad is displayed in a mobile device independently of the ad clicks. The *Clicks* on the above formula is the number of times a user have clicked on an add of a given app. Unfortunately, there is no such information available on the appstores in general, as it is

kept from the advertising networks unpublished. For this reason, the question we will attempt to answer is: *How much do developers of free apps need to gain per download in order to match the income of paid apps?* It would be desirable to compute this amount only for the free apps in SlideMe marketplace that uses ads. To this end, we needed to find which of our collected apps are equipped with ads. To do this, we used *Androguard* [6], an Android reverse engineering analysis tool capable of detect ad/open source libraries. For each collected APK file we examine which of the top-20 advertising libraries, according to this paper [45], were used. Figure 7.13 shows the CDF of the unique advertising networks per app in SlideMe appstore. Regarding the graph, 33% of the apps do not seem to use advertisements (at least one of the 20 most popular ad networks), and the rest 67% are equipped with ads. We can see that most apps use a few number of advertising networks, where a small percentage, 1% of them use up to 12 different ad libraries. Then, we would like to see how the different ad networks are distributed

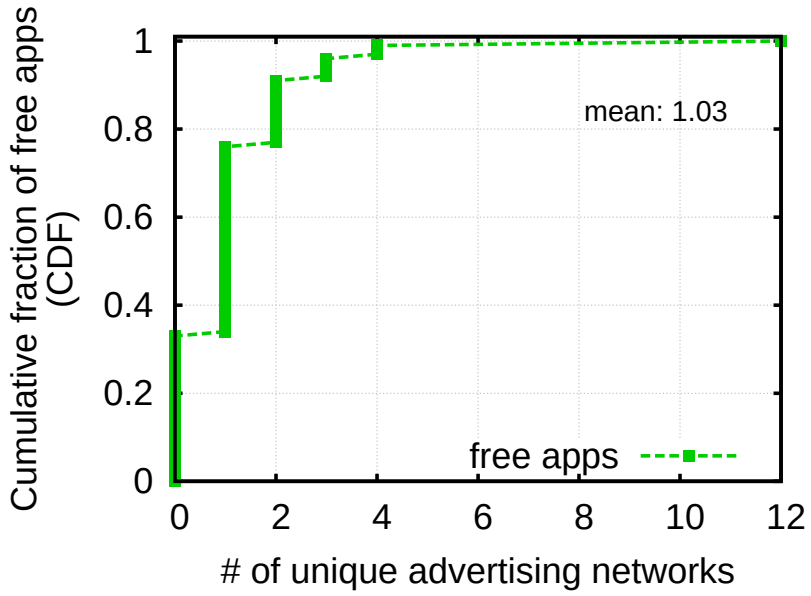
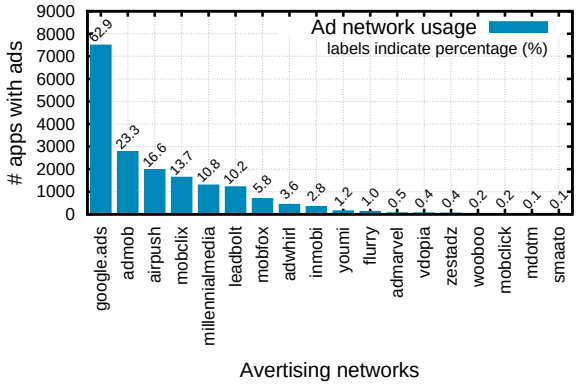
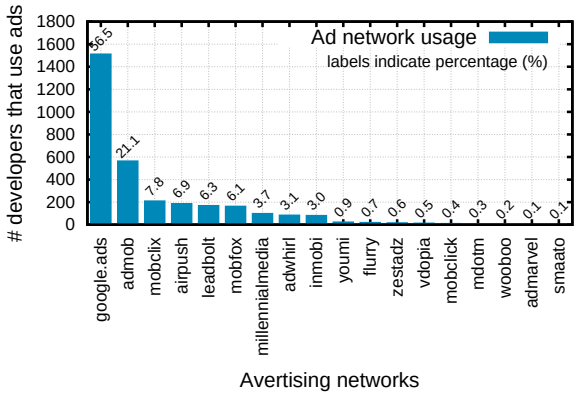



FIGURE 7.13: CDF of unique Advertising Networks per free app in SlideMe marketplace.

over the free apps in SlideMe appstore and across the developers producing these apps.




(a) Apps




(b) Developers

FIGURE 7.14: Advertising Networks usage among apps and developers in SlideMe appstore.

The results of these distributions are shown in Figure 7.14. We can see that ad networks' usage rates appear to be approximately the same among apps and developers. The most popular advertising networks seem to be *Google AdSense* along with *AdMob* and *Airpush* which are utilized from the majority of the free apps (Figure 7.14(a)). Notice that the percentages are not sum up to 100%, as there are several apps that use more than one ad network. Similarly, the most popular ad networks for developers seem to be *Google AdSense*, *AdMob* and *Mobclix* (Figure 7.14(b)).

Thereafter, having already isolated the free apps that use ads, we will proceed by estimating the necessary average ad income that a free app has to make, in order to match the income of a paid app. As we already discussed earlier in this section, we do not have the information of the *Click-through Rate*, which is essential for the computation of ad income for a free app.

Nevertheless, we are able to compute which is the necessary income for a free app (per download) in order to match the income of a paid app based on the number of downloads. We compute this value by equating the average income (per app per download) of free apps with the respective average income of paid apps:

$$Ad\,Income = \frac{\sum_{i=1}^{N_{paid}} Downloads(i) \times Price(i)/N_{paid}}{\sum_{j=1}^{N_{free}} Downloads(j)/N_{free}} \tag{7.1}$$

Thus, the average necessary ad income for a free app can be calculated based on the formula 7.1. To proceed our analysis, we calculate the necessary average ad income values for the last three months of our measurement period. The results are illustrated in Figure 7.15, where we can see that

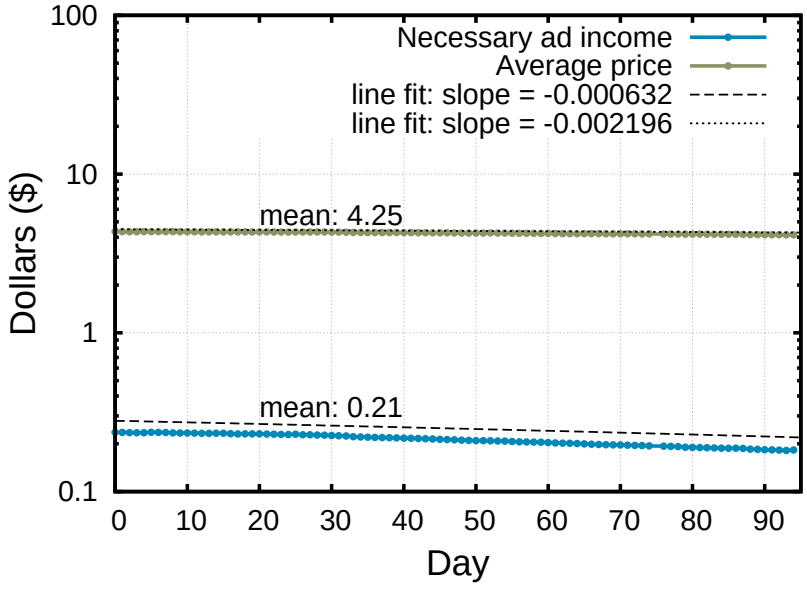



FIGURE 7.15: Average *necessary ad income* per app per download along with the average price of paid apps through time in SlideMe marketplace.

*necessary ad income* on average is 0.21 dollars. That is, 0.21 dollars per download, are needed by a free app in average so as to match the income of the average paid app. Moreover, we see that *necessary ad income* is dropping over time. It seems that this drop over time happens due to the fact that downloads of free apps are increasing much faster than those of paid apps. To confirm this hypothesis, we attempted to plot the downloads as a function of time, for all the new apps added in the SlideMe appstore during our measurement interval. In particular, we isolate all the new added apps, divide them into free and paid, and collect their downloads in different

bins (*day-bins*), one containing the number of app downloads for the first day of their lifetime, one for their second day, one for their third day, *etc.* Consequently, the *day-bins* will not contain download samples from the same number of apps. Actually, the number of apps decreases for each following *day-bin*, as there are apps with different lifespan. To put it differently, there are apps entered in SlideMe appstore the last day of the measurements, the penultimate day, and generally $N$ days before the end of the measurements. Thus, we expect the first *day-bin* to have the largest number of apps, as it contains download samples of the first day of lifetime of all apps, then the second *day-bin* will have fewer apps, as there are apps that are alive only for 1 day (the last one), *etc.* Then, we compute the average number of downloads of these bins for each day. Figure 7.16 depicts the results of this

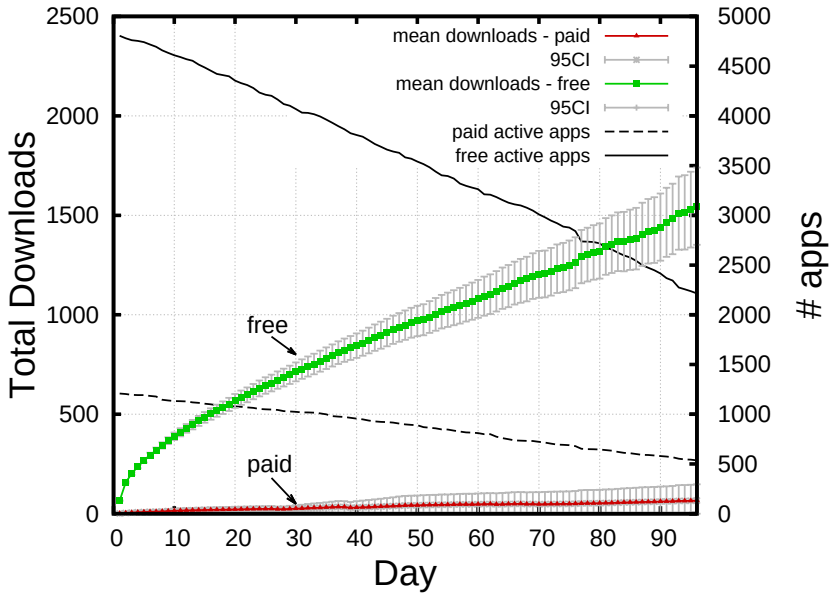



FIGURE 7.16: Mean downloads per app over time of free and paid apps that were added in the SlideMe appstore within our measurement period.

analysis. As expected, downloads of free apps are grow at a faster rate than those of paid apps. The thin black lines, that appear to drop downward in the second $y$ axis (on the right), indicate the set of active apps, on which the average income value is currently computed. As already mentioned, the number of apps is fewer for each subsequent *day-bin* because there are apps with different lifetimes.

Afterwards, we would like to see whether *free-app ads strategy* is better or not. That is, maybe this is true for very popular apps but not for the ones with a few downloads. To be able to draw conclusions, we attempted to replot the previous graph showing the *necessary ad income* through time, for apps in different popularity levels. To this end, we proceed to make a kind

of app binning based on the number of downloads (app popularity) that the apps had at the last day of the measurements. Thus, we separated the apps in 3 different *popularity bins*: *highly popular* which contains the top 20% of most popular apps, *medium popular* that contains the next 50% of the popular apps, and finally, *unpopular* that contains the 30% of least popular applications. Figure 7.17 shows the average *necessary ad income* for the 3

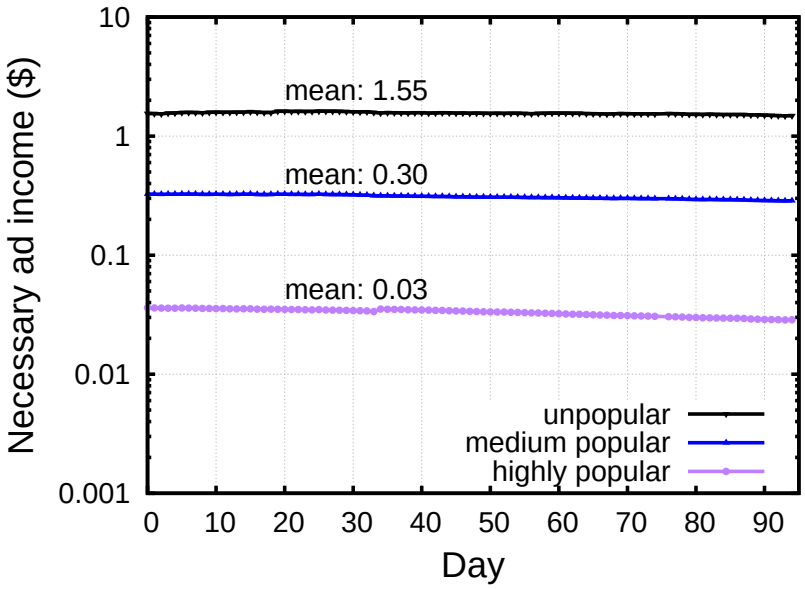
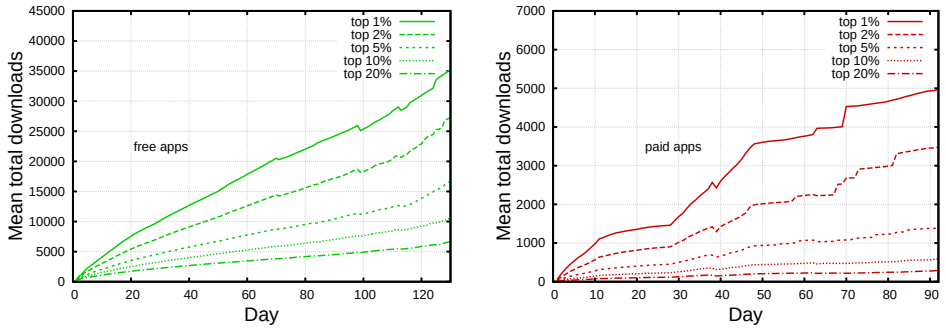



FIGURE 7.17: Average *necessary ad income* per app per download through time. in the SlideMe appstore within our measurement period.

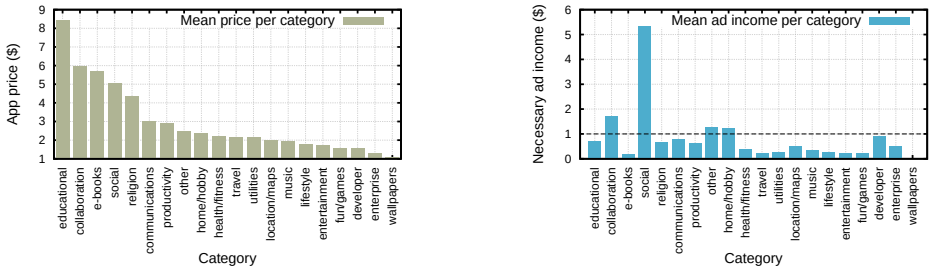
Average *necessary ad income* per download so that the income of free apps is matching the income of the paid apps. We have used 3 different bins of apps (popular: top 20% of popular apps, medium popular: next 50% of popular apps and unpopular: least 30% of popular apps).

different popularity bins. We can see that the bin with the most popular apps needs at least 0.039 dollars per download per app, so that the income of a free app surpasses this of a paid one. Moreover, this income value *becomes almost 9 times higher for an app in the medium popularity bin* and *42 times greater for an unpopular app!* This means that even for *medium-popularity* applications, it actually makes sense to choose the ads strategy. We also observe that the *necessary ad income* is dropping over time. This happens because of the faster growth of free apps' downloads versus those of the paid ones. We can confirm this by plotting the downloads as a function of time for different popularity app volumes, as shown in Figure 7.18. We observed similar behaviour exhibited by lower popularity volumes too.




(a) Free (b) Paid

FIGURE 7.18: Mean downloads per app over time for top (1%, 2%, 5%, 10% and 20% of) free apps that were added in the SlideMe appstore within our measurement period.




(a) Free (b) Income

FIGURE 7.19: Mean price per category for paid apps vs average *necessary ad income* per category of free apps, in SlideMe marketplace.

We already found that *high-* and *medium-popularity* applications are ideal for choosing the ads strategy in order to gain income. Accordingly, we would like to examine whether this conclusion changes by category. We first compute the *necessary ad income* for all apps per category (Figure 7.19), and then for the *popularity bins* of apps per category (Figure 7.20), as we have already defined. In Figure 7.19(a), we see the average price of paid apps per category, while in Figure 7.19(b), we see the average *necessary ad income* of the free apps among different categories. We kept the same order of categories in the two graphs for a better side-by-side comparison. We see that there is no correlation between the average price per category and the respective average *necessary ad income.* Moreover, we see that the average necessary ad income for a free app in a given category in order to match the income of a paid app in the same category is lower than 1 dollar, except from categories: *social, home/hobby* and *other.* Then we computed the

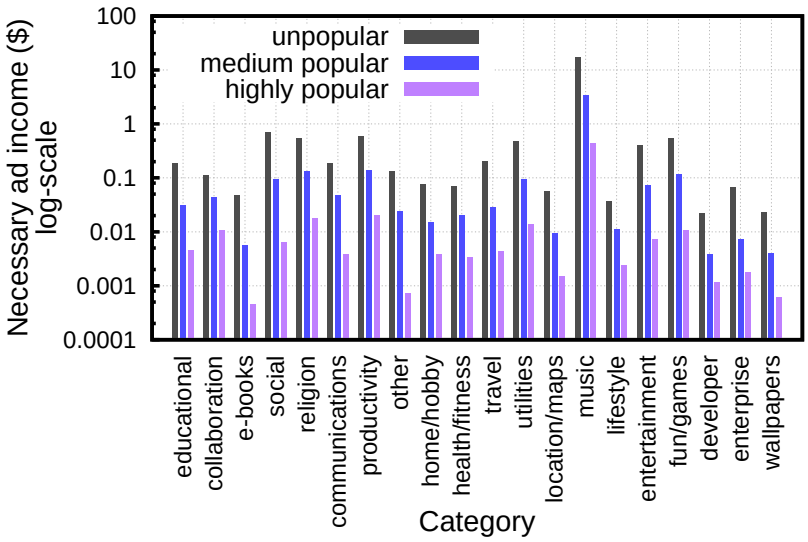



FIGURE 7.20: Average *necessary ad income* per category, for 3 different popularity bins in the SlideMe marketplace. The $y$ axis is in log-scale.

average *necessary ad income* for the different *popularity bins*. We created popularity bins for each category separately. The results are summarised in Figure 7.20. According to the figure, the difference in *necessary ad income* is obvious across different popularity levels (per category). Moreover, the most promising categories in terms of gaining profit, especially for the most popular apps, appear to be *E-books, wallpapers, developer, location/maps* and *enterprise*.

## 7.5 Summary

This chapter performed an analysis in order to explore the role of pricing in smartphone applications ecosystem. Our results show that most of developers produce a few number of applications. However, we found a small number of developers with a huge portion of apps hosted in SlideMe appstore and we ascertained that these developers correspond to companies. Moreover, we clarified that developers seem to have a common strategy, and clearly prefer to offer either only free or only paid apps (mainly free). In addition, we observed that most of developers (around 85%) produce apps for only 3 out of 20 categories. Furthermore, the majority of developers earned a few dollars from 0 to 10, while a very small portion, about 1% of them gained millions of dollars. Also, we showed that quality is more important than quantity for developers income (as the number of apps of each developer is not connected with her total income). Moreover, we saw that categories with highest income differ from categories with largest amount of apps. Finally, we estimated the necessary ad income of a free app per download in order to match this of a paid app, and demonstrated that it ranges across different popularity levels and across app categories.

# 8 Related Work

In this chapter we place our work in appropriate context by presenting work related to this thesis. We first overview similar studies on content popularity and then we present some systematic studies on mobile applications.

## 8.1 Similar Studies on Content Popularity

There are numerous studies focusing on content popularity for a variety of different Web 2.0 technologies and distributed systems, including information sharing systems, User Generated Content (UGC) systems, social networks and so on.

The most similar to our work is by Zhong and Michahelles [54]. This study is based on a large amount of data provided by *42matters AG*[1] that captures installations, updates and removal of apps in real time. We argue that their dataset is limited compared with ours, as they do not know the exactly number of downloads of each app, but a number of installations from a client-side monitor app. The results of this study propose that the Android market is a "Superstar" market where top 1%, 5% and 10% of the most popular apps count for the roughly 50%, 80% and 100% percent of the aggregate popularity. This *supremacy* of the most popular apps in Android market that seems to be stronger than the Pareto Principle (which claims that about 80% of the consequences are created by 20% of the causes) is also observed in our experiments too. The dominance of the most popular apps is also denoted from two phenomena that are illustrated in sales distribution of their experiments: the *natural monopoly* and the *double jeopardy*.

[1] `http://appaware.com/`

Natural monopoly states that not only the most popular apps are installed from overbalanced portion of of users, but also these users purchase more popular apps than unpopular ones. The double jeopardy describes that the unpopular apps attract the minority of the users and are characterized by lower satisfaction rate. As the overall conclusion of their work, it may be be stated that mobile app markets in general may be characterized by different structure features than other online e-commerce marketplaces. The Android Market appears to be a glaring example of such a Superstar market dominated by hit apps. The main differences with our work is that we gathered and analyzed data from 4 different alternative Android markets and we don't focus mostly on paid apps from a business/sales perspective, as they do. Moreover, they did not notice the fetch-at-most-once effect in the distribution of popularity, nor did they attempt to give an explanation about the drop for large download values which we explained with the clustering effect.

There are many other similar works in different fields and technologies. For instance, in [32] Cha *et al.* study the video contents of YouTube [29], the world's largest UGC system. Their key findings show that the popularity distribution follows power-law with an exponential cutoff, that is very similar to popularity distribution of our study. In addition, the existence of the Pareto Principle (80-20 rule), which was demonstrated in our work, is confirmed there as well. In a similar study [30], the authors systematically analyze the content of URL shortening services using traces derived from the services themselves and by crawling the Twitter [28] social network for a 3 month period. They show that the short URL click distribution follows log-normal, verifying the existence of the Pareto Principle too.

There are several studies showing that World Wide Web (WWW) content exhibits power-law and heavy-tailed distributions [31, 36]. Particularly, in [31], Breslau *et al.* analyze 6 traces from proxies deployed in different academic institutions, enterprise networks and Internet Service Providers (ISPs). The main finding of their work shows that the distribution of Web requests follows a Zipf-like distribution. Based on this outcome, they construct a simple model for the web accesses that assumes that the requests are independent and distributed according to a Zipf-like curve. They demonstrate that their simple model is sufficient to explain the certain asymptotic behavior of different properties (hit-ratio, interval-times) that observed in their collected web cache traces. Moreover, they investigate whether this model could be help in cache replacement strategies. The fact that the largest portion of the web requests distribution fits a straight line well, that is a perfect Zipf distribution (in log-scale), can be explained by the web users' activity. Users in the web tend to browse the same page continuously (*e.g.* the http://www.google.com). In contrast, this behavior (an exact Zipf curve) cannot be observed in our work. This happens because, as explained

in Chapter 4.2, one smartphone user typically will download one application only once (the *fetch-at-most-one* principle).

Similar power-law distributions with exponential cutoffs have been identified in various networks such as the live streaming media networks [35], the P2P networks [46], as well as the protein, e-mail, actor and collaboration networks [43]. Also there are many other works focusing on the explanation and understanding of the power-law distributions with examples outside the scope of WWW [48, 49].

## 8.2 Systematic Studies of Smartphone Applications

Recently, mobile systems and applications have attracted the interest of researchers that try to understand the behavior and functionality of this emerging technology. Several research efforts were made for collecting and analyzing a large set of mobile applications from multiple marketplaces, mainly focused on security and privacy-related analysis, such as malware detection [56], malware analysis [55], overprivilege identification [42], detection of privacy leaks [38, 44], malicious advertisement libraries [45], and vulnerability assessment [39]. In this work we collected and analyzed a similar large scale dataset, but our analysis was focused on characterizing and modeling the workload of the monitored marketplaces.

In a work closely related to ours, Xu *et al.* [53] study the usage behavior of smartphone apps by analyzing traces from a tier-1 cellular network provider. Their analysis is mostly focused on spatial and temporal locality, geographic coverage, and diurnal usage patterns. On the other hand, our analysis is focused on app popularity and pricing strategies. Moreover, we use a different dataset, by systematically crawling four third-party appstores, while Xu *et al.* use an IP-level trace, which leads to a different analysis.

Other related approaches focus on mobile traffic analysis, but they do not study mobile applications. Maier *et al.* [47] perform a study of residential DSL lines of a European ISP and find that mobile devices' traffic is dominated by multimedia content and applications' downloads. Falaki *et al.* [40] conduct a traffic analysis of 43 different smartphones. Their findings show that browsing contributes most of the traffic, and lower layer protocols impose high overheads due to small transfer sizes. They also study the factors that impact performance and power consumption on smartphone communications and propose several improvements.

Falaki *et al.* [41] analyze the behavior of 255 users in two different smartphone platforms in order to understand and characterize user activities and their impact on network and battery. They observed a diversity in user patterns, which implies that techniques for improving user experience or power consumption for the average case may be inefficient for a large fraction of users. Wei *et al.* [51] present a multi-layer system for monitoring

and profiling Android apps at runtime. While they provide a useful tool for monitoring individual apps, we present a large scale measurement study about the app usage and pricing models.

# 9
# Future Work

In this chapter we discuss ideas about the next steps of this work, and questions raising from our results.

In this thesis, we performed a systematic study on four different alternative Android marketplaces in order to understand the mobile applications' ecosystem. However, a plethora of other marketplaces exists. In fact, many of the appstores provide rough statistics on their apps, such as download ranges (*e.g.* 500 - 1000) of the apps, instead of the exact number as shown in Section 2.1. However, we visited many of these appstores and noticed that most of them have accurate information regarding the ratings of users. As we found in 4.3, app popularity and user ratings showed a linear relationship in all the monitored appstores. This observation can lead us to the conclusion we can approximate app popularity with the number of user ratings, which allows us to crawl other appstores (*e.g.*, *Windows Phone* [26] marketplace) that only provide number of ratings.

Furthermore, our collection of data contains the APK files of all versions of each app during the crawling period. An analysis on these files would be interesting as it would bring insight on the content of these applications. For instance, in our next steps, we are thinking of exploring the actual content of this apps. That is, some main characteristics of these apps, such as their size, the *Android Permissions* that they request, the number of different advertising networks they use and how these characteristics change from version to version. Some questions on these could be: *"How the size in MB of an application changes from version to version?" "Do applications use to request more permissions in subsequent versions?"* Moreover, having the size of all applications of a given appstore and its daily snapshot (all

the downloads of every app), we could compute the appstore's Bandwidth usage through time in terms of applications' downloads.

In Chapter 7, we measured the total income per developer for all the developers in SlideMe marketplace, but we said nothing about their effort. An interesting study would be to determine the effort for each developer and to compare it with her total income. For instance, E-books or wallpapers require less effort than games or geolocation apps. The challenge here is to find a way to quantify this effort. This could be approximated by the number of different objects found in an app (images, sound files), and the size of source code (*i.e.* by reverse engineering the APK files).

An other interesting analysis would be an attempt to model the app popularity of *single* apps through time. That is, we could classify the applications in different categories based on their downloads distribution through time (*e.g.* the apps with high rise in popularity through time, those that experience a popularity rise at their first steps and then their downloads reach a limit *etc.*). Afterwards, we can construct a model for these categories with respect to app popularity, with a view to be able to predict the popularity of an app through time, given the number of downloads of its first days.

Moreover, in Section7.4, we saw that the average necessary income of a free app, in order to match the average income of a paid app, is dropping over time, and proved that this happens due to the fact that downloads of free apps (*i.e.* popularity) are increasing much faster than those of paid apps. An interesting question on this observation would be: *"How much* time *is needed so as the income of free app to match the income of a paid app?"*

# 10 Conclusion

In this thesis we presented a systematic study in order to explore and better understand the mobile app ecosystem. To accomplish this, we crawled four alternative Android marketplaces for several months and collected information and statistics of their apps through time. To the best of our knowledge, this work is the first large-scale study in the literature that sheds light onto the explosive growth of application marketplaces.

Our results shows that appstores are dominated by a very small number of popular apps that receive a very large amount of download requests, while the majority of the apps are downloaded only a few times verifying the presence of the Pareto Principle. We found that the distributions of applications popularity in all monitored appstores exhibits a Zipf-like behavior with some deviations. This deviations appear to happen in part from "fetch-at-most-once" behavior of users as well as from a more general phenomenon, we call "the clustering effect". According to "clustering effect", which can be a result of recommendation systems or other grouping forces, the apps are grouped in different sets and if a user downloads one of them, then the same user will probably download another app of the same set rather than switching to another one. We verify our hypothesis of "clustering effect" through "user temporal affinity", a new metric that express the affinity that a user has to app categories in an appstore. We measured the user temporal affinity in a large dataset of user comments that implies user downloads, and we found that there is a strong affinity between users and app categories which validated our hypothesis of clustering effect. Then, we propose a novel model of appstore usage based on both "fetch-at-most-once" and "clustering effect" properties, and we evaluate our model through a series of simulations comparing its results with the actual appstore workloads

(applicatons' downloads). We find that our model approximate very well the actual app popularity distribution. Moreover, we present a study on the role of pricing in smartphone apps. We see that the popularity distribution of paid apps is different than this of free app, following a clearly power-law behavior. This is probably due to the fact that users are more selective when downloading paid applications. Furthermore, we outline the behavior of app developers and give information about their income and their strategies (developers seem to have a common strategy, and clearly prefer to offer either only free or only paid apps).

Overall, we believe that our study can be useful for both appstore operators and developers: (i) The existence of locality in user downloads, as this is manifested by the Pareto effect, can help appstores design efficient caching mechanisms that will improve the speed of delivering apps to end users. (ii) The understanding of download patterns, like clustering effect, can help appstores to design better recommendation systems, which can benefit apps and developers by giving them better opportunities for more suggestions and downloads. Moreover, users that prefer a variety of choices will have a better experience with recommendation systems that do not bombard them with the same set of most popular choices. (iii) Understanding the parameters that affects app popularity are of high interest for developers that want to predict, understand, and most importantly, improve the popularity of their apps. (iv) Understanding the temporal affinity of users to app categories can help the design of more efficient prefetching and advertisement methods. (v) Our model for app downloads will be helpful for appstores to estimate app popularity and future downloads of each app. Estimating the downloads per app will enable appstores to pinpoint problematic apps and either favor them through better recommendations or remove them from the market. (vi) Understanding which pricing models result in larger revenue can help developers to choose an appropriate pricing policy for their apps to increase app popularity and their income.

# Bibliography


[1] 18 places to download Android apps (that aren't the Google Play Store). http://mobiputing.com/2011/06/17-places-to-download-android-apps/.

[2] 8 Alternative Android App Stores From China. http://www.techinasia.com/8-android-app-stores-china/.

[3] Adoption of New Technology since 1900. http://visualizingeconomics.com/2008/02/18/adoption-of-new-technology-since-1900/.

[4] Alternative Android App Stores. http://www.boundbytech.com/alternative-android-app-stores/.

[5] Amazon Appstore. http://www.amazon.com/appstore/.

[6] Androguard. http://code.google.com/p/androguard/.

[7] Android activations hit 1.3M per day, says Google's Schmidt. http://news.cnet.com/8301-1035_3-57506722-94/android-activations-hit-1.3m-per-day-says-googles-schmidt/.

[8] Android apps with ads found to pose privacy and security risks. http://www.bgr.com/2012/03/20/android-apps-with-ads-found-to-pose-privacy-and-security-risks/.

[9] Android dominates China's smartphone market. http://www.zdnet.com/android-dominates-chinas-smartphone-market-7000000634/.

[10] Android Market alternatives. http://www.blog.trilenagames.com/?p=58.

[11] Android Market Blocked in China. http://phandroid.com/2011/10/10/android-market-blocked-in-china/.

[12] AndroLib. http://www.androlib.com/.

[13] AppBrain. http://www.appbrain.com/.

[14] China is Fastest Growing iOS and Android Market, Says Flurry. http://www.techinasia.com/china-smartphone-ios-android-flurry.

[15] Gartner Says 428 Million Mobile Communication Devices Sold Worldwide in First Quarter 2011. http://www.gartner.com/it/page.jsp?id=1689814.

[16] Gartner says smartphone sales grew 44 percent in Q1 2012. http://www.techsmart.co.za/features/news/Gartner_says_smartphone_sales_grew_44_percent_in_Q1_2012.html.

[17] Google Play Terms of Service. http://www.google.com/mobile/android/market-tos.html.

[18] iOS and Android Adoption Explodes Internationally. http://blog.flurry.com/bid/88867/iOS-and-Android-Adoption-Explodes-Internationally.

[19] Quarterly Device Sales in 2011. http://www.mobilestatistics.com/mobile-statistics.

[20] Scrapy framework. http://scrapy.org/.

[21] Selenium Remote Control (RC), a web application testing system. http://seleniumhq.org/projects/remote-control/.

[22] The 1Mobile Marketplace website . http://www.1mobile.com/.

[23] The Anzhi Marketplace website. http://www.anzhi.com/.

[24] The AppChina Marketplace website. http://www.appchina.com/.

[25] The SlideMe Marketplace website. http://slideme.org/.

[26] The Windows Phone Marketplace website. http://www.windowsphone.com/en-us/store.

[27] Tuning In: Communications technologies historically have had broad appeal for consumers. http://www.karlhartig.com/chart/techhouse.html.

[28] Twitter. http://twitter.com/.

[29] YouTube. http://youtube.com.

[30] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis. we.b: the web of short urls. In *Proceedings of the 20th international conference on World wide web*, pages 715–724, 2011.

[31] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM*, pages 126–134, 1999.

[32] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14, 2007.

[33] J. Cho and S. Roy. Impact of search engines on page popularity. In *Proceedings of the 13th international conference on World Wide Web (WWW)*, pages 20–29, 2004.

[34] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, pages 00–00, 2003.

[35] C. P. Costa, I. S. Cunha, A. Borges, C. V. Ramos, M. M. Rocha, J. M. Almeida, and B. Ribeiro-Neto. Analyzing client interactivity in streaming media. In *Proceedings of the 13th international conference on World Wide Web*, pages 534–543, 2004.

[36] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[37] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945.

[38] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation (OSDI)*, pages 1–6, 2010.

[39] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of android application security. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 21–21, Berkeley, CA, USA, 2011.

[40] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 281–287, 2010.

[41] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th*

*international conference on Mobile systems, applications, and services (MobiSys)*, pages 179–194, 2010.

[42] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 627–638, New York, NY, USA, 2011.

[43] T. Fenner, M. Levene, and G. Loizou. A stochastic evolutionary model exhibiting power-law behaviour with an exponential cutoff. *Physica A: Statistical Mechanics and its Applications*, 355(2):641–656, 2005.

[44] M. Grace, Y. Zhou, Z. Wang, and X. Jiang. Systematic detection of capability leaks in stock android smartphones. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS)*, 2012.

[45] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012.

[46] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 314–329, New York, NY, USA, 2003. ACM.

[47] G. Maier, F. Schneider, and A. Feldmann. A first look at mobile handheld device traffic. In *Proceedings of the 11th international conference on Passive and active measurement (PAM)*, pages 161–170, 2010.

[48] S. Mossa, M. Barthelemy, H. E. Stanley, and L. A. N. Amaral. Truncation of power law behavior in "scale-free" network models due to information filtering. *PHYS.REV.LETT*, page 138701, 2002.

[49] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, pages 323–351, December 2005.

[50] A. Pathak, Y. C. Hu, and M. Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, EuroSys '12, pages 29–42, 2012.

[51] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos. Profiledroid: multi-layer profiling of android applications. In *Proceedings of the 18th annual international conference on Mobile computing and networking (Mobicom)*, pages 137–148, 2012.

[52] Wikipedia. Google Play. `http://en.wikipedia.org/wiki/Google_Play`.

[53] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 329–344, 2011.

[54] N. Zhong and F. Michahelles. Long tail or superstar? an analysis of app adoption on the android market. In *Proceedings of the 5th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia*, 2012.

[55] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 95–109, 2012.

[56] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *Proceedings of the 19th Annual Network & Distributed System Security Symposium (NDSS)*, 2012.