University of Nevada, Reno

POSN: A Cloud Based Privacy Preserving Decentralized Personalized Online Social Network

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Eric R. Klukovich

Dr. Mehmet Hadi Gunes, Thesis Advisor

May, 2016

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

ERIC KLUKOVICH

Entitled

**POSN: A Cloud Based Privacy Preserving
Decentralized Personalized Online Social Network**

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Dr. Mehmet Hadi Gunes, Advisor

Dr. Frederick C. Harris, Jr., Committee Member

Dr. Nicholas Seltzer, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

May, 2016

Abstract

As more users opt to use Online Social Networks (OSNs) to interact with friends and family, the privacy of the user is a growing concern. The current OSN architecture forces the user to trust the social network provider with his or her content, which can lead to privacy issues and concerns. When designing a new OSN, it is critical to understand how users interact over OSNs in order to provide the users with similar functionality while maintaining efficiency and privacy. Decentralized server architectures are an alternative that provide more privacy to the user by giving the user direct control of his or her content. Storage and data availability are a major concern in a decentralized system due to the burden of supplying resources is typically placed on the user. Cloud storage can be used to remedy that issue because it is available, and many cloud providers offer a significant amount of free storage to their users. As the usage of mobile devices is constantly growing, mobile devices can be paired with the clouds to create a decentralized social network. Encryption mechanisms can be used to guarantee that only the proper users can access the content, therefore enhancing the privacy.

In this thesis, POSN, Personal Online Social Network, is introduced and takes a new approach for a privacy-preserving decentralized OSN platform. POSN is an Android application that utilizes both cloud storage and mobile devices to enhance the user's privacy. Symmetric and asymmetric encryption schemes are used to ensure data confidentially and provide fine-grain access control. A feasibility study was carried out to analyze the data transfer performance of the two popular social networks and five cloud storage providers to determine if clouds can be utilized to store and deliver content to users. The design of the POSN architecture is discussed in detail, including how the files in the system are structured. The implementation of the functionality available in POSN is also dicussed.

Dedication

I dedicate my thesis work to my parents, Richard and Nadia Klukovich, my twin sister Rachel Klukovich, and my girlfriend Sarah Key for always supporting and encouraging me to push myself to pursue my dreams.

Acknowledgments

I would like to thank Dr. Mehmet Gunes, Dr. Fred Harris, and Dr. Nicholas Seltzer for being on my committee and providing me helpful feedback. I would like to give a special thanks to Dr. Gunes for giving me the opportunity to do research and advising me throughout my undergraduate and graduate studies. I would also like to thank Dr. Harris for allowing me to use the resources and to socialize in the HPCVIS lab. I would also like to thank Esra Erdin for collaborating with me with the design of this thesis, Gurhan Gunduz for assisting with the cloud measurement analysis, and James Bridegum for the initial cloud measurements. Finally, I would like to thank my friends: Cameron Rowe, Christine Johnson, Daniel Chavez, and Matthew Van-Compernolle for supporting me and assisting me throughout my studies at UNR.

This material is based upon work supported by the National Science Foundation under grant number EPS-IIA-1301726 and CNS-1321164. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Contents

Abstract **i**

Dedication **ii**

Acknowledgments **iii**

List of Figures **vi**

1 Introduction **1**

2 Background **4**

2.1 Social Networks 4

2.1.1 Centralized Architecture 4

2.1.2 Decentralized Architecture 5

2.1.3 Challenges 6

2.2 Libraries and APIs 9

2.2.1 Android Operating System 9

2.2.2 Social Network Connectivity 9

2.2.3 Cloud Storage Connectivity 10

2.2.4 Android Asynchronous HTTP Client 14

2.2.5 JSON 14

3 Cloud Measurements **16**

3.1 Measurement Application 16

3.2 Measurement Process 19

3.3 Wi-Fi Measurements - Varying File Sizes 20

3.4 Wi-Fi Measurements - Different Locations 27

3.5 3G/4G Measurements 33

3.6 Related Work 34

4 Personal Online Social Network (POSN) Architecture **35**

4.1 System Overview 35

4.2 File Organization 37

4.2.1 Owner Files 37

4.2.2 Friend Files 39

4.2.3 Group Wall Files 39
4.2.4 Group Archive Files 40
4.2.5 Temporary Comment Files 40
4.3 Related Work 41

5 POSN Application Implementation 44
5.1 Local Application Files 44
5.2 POSN Functionality 45
5.2.1 Initial Setup 45
5.2.2 Unique ID Generation 46
5.2.3 Friendship Establishment 46
5.2.4 Access Control 48
5.2.5 Content Sharing 49
5.2.6 Content Access 49
5.2.7 Friend File Update 50
5.3 Application Interface 50

6 Conclusions 55

7 Future Work 57
7.1 Cloud Measurement Study 57
7.2 POSN Security Enhancements 58
7.3 POSN Application 58

Appendices 60

A Complete Cloud Measurment Results 60
A.1 Different File Size Results Per Location 60
A.2 Different Location Results Per File Size 67
A.2.1 Photo Files 67
A.2.2 Video Files 78

Bibliography 86

List of Figures

2.1 Example of a centralized social network architecture 5
2.2 Example of a decentralized social network architecture 6
2.3 Example of the Dropbox user authentication process 11
2.4 An example of a JSON data structure definition for a list of wall posts. 14

3.1 Download time for different file sizes at a residential location (ISP 1) (log-log scale) 21
3.2 Upload time for different file sizes at a residential location (ISP 1) (log-log scale) 21
3.3 Download throughput performance for different file sizes at a residential location (ISP 1) (log-log scale) 22
3.4 Upload throughput performance for different file sizes at a residential location (ISP 1) (log-log scale) 22
3.5 Average download time comparison for all locations with different file sizes (log-log scale) 24
3.6 Average upload time comparison for all locations with different file sizes (log-log scale) 24
3.7 Average download throughput comparison for all locations with different file sizes (log-log scale) 25
3.8 Average upload throughput comparison for all locations with different file sizes (log-log scale) 25
3.9 Average cloud and OSN time comparison with different file sizes (log-log scale) 26
3.10 Average cloud and OSN throughput comparison with different file sizes (log-log scale) 26
3.11 TCP slow start and congestion control process 27
3.12 Download time for various locations with a 718.43 KB photo (log scale) 28
3.13 Upload time for various locations with a 718.43 KB photo (log scale) 28
3.14 Download throughput for various locations with a 718.43 KB photo (log scale) 29
3.15 Upload throughput for various locations with a 718.43 KB photo (log scale) 29
3.16 Download time for various locations with a 32 MB video (log scale) . 30

3.17 Upload time for various locations with a 32 MB video (log scale) . . . 30
3.18 Download throughput for various locations with a 32 MB video (log scale) . . . 31
3.19 Upload throughput for various locations with a 32 MB video (log scale) 31
3.20 3G/4G download performance for different devices on different cellular providers . . . 33

4.1 Example of how the data owner (such as Alice) uploads data into her cloud and how friend content is fetched from the friend's cloud 36
4.2 Diagram for the POSN file system . . . 38

5.1 Friendship establishment process between two users . . . 47
5.2 Initial login screen . . . 51
5.3 Account creation . . . 51
5.4 Cloud access . . . 52
5.5 Key generation and group creation . . . 52
5.6 Posting new content . . . 53
5.7 Wall and notification tabs . . . 53
5.8 Messaging tab . . . 54
5.9 Friend list tab . . . 54

A.1 Download and upload time for different file sizes at the airport (log-log scale) . . . 60
A.2 Download and upload throughput for different file sizes at the airport (log-log scale) . . . 60
A.3 Download and upload time for different file sizes at the bookstore (log-log scale) . . . 61
A.4 Download and upload throughput for different file sizes at the bookstore (log-log scale) . . . 61
A.5 Download and upload time for different file sizes at the cafe (log-log scale) . . . 61
A.6 Download and upload throughput for different file sizes at the cafe (log-log scale) . . . 62
A.7 Download and upload time for different file sizes at the library (log-log scale) . . . 62
A.8 Download and upload throughput for different file sizes at the library (log-log scale) . . . 62
A.9 Download and upload time for different file sizes at the mall (log-log scale) . . . 63
A.10 Download and upload throughput for different file sizes at the mall (log-log scale) . . . 63
A.11 Download and upload time for different file sizes for ISP 1 (log-log scale) 63
A.12 Download and upload throughput for different file sizes for ISP 1 (log-log scale) . . . 64

A.13 Download and upload time for different file sizes for ISP 2 (log-log scale) 64
A.14 Download and upload throughput for different file sizes for ISP 2 (log-log scale) 64
A.15 Download and upload time for different file sizes for ISP 3 (log-log scale) 65
A.16 Download and upload throughput for different file sizes for ISP 3 (log-log scale) 65
A.17 Download and upload time for different file sizes at the university (log-log scale) 65
A.18 Download and upload throughput for different file sizes at the university (log-log scale) 66
A.19 Download and upload time for various locations with a 2.42 KB photo (log scale) 67
A.20 Download and upload throughput for various locations with a 2.42 KB photo (log-log scale) 67
A.21 Download and upload time for various locations with a 4.89 KB photo (log scale) 68
A.22 Download and upload throughput for various locations with a 4.89 KB photo (log-log scale) 68
A.23 Download and upload time for various locations with a 9.89 KB photo (log scale) 69
A.24 Download and upload throughput for various locations with a 9.89 KB photo (log-log scale) 69
A.25 Download and upload time for various locations with a 21.98 KB photo (log scale) 70
A.26 Download and upload throughput for various locations with a 21.98 KB photo (log-log scale) 70
A.27 Download and upload time for various locations with a 37.81 KB photo (log scale) 71
A.28 Download and upload throughput for various locations with a 37.81 KB photo (log-log scale) 71
A.29 Download and upload time for various locations with a 79.08 KB photo (log scale) 72
A.30 Download and upload throughput for various locations with a 79.08 KB photo (log-log scale) 72
A.31 Download and upload time for various locations with a 105.19 KB photo (log scale) 73
A.32 Download and upload throughput for various locations with a 105.19 KB photo (log-log scale) 73
A.33 Download and upload time for various locations with a 165.38 KB photo (log scale) 74
A.34 Download and upload throughput for various locations with a 165.38 KB photo (log-log scale) 74
A.35 Download and upload time for various locations with a 200 KB photo (log scale) 75

A.36 Download and upload throughput for various locations with a 200 KB photo (log-log scale) 75
A.37 Download and upload time for various locations with a 400 KB photo (log scale) 76
A.38 Download and upload throughput for various locations with a 400 KB photo (log-log scale) 76
A.39 Download and upload time for various locations with a 718.43 KB photo (log scale) 77
A.40 Download and upload throughput for various locations with a 718.43 KB photo (log-log scale) 77
A.41 Download and upload time for various locations with a 1 MB video (log scale) 78
A.42 Download and upload throughput for various locations with a 1 MB video (log-log scale) 78
A.43 Download and upload time for various locations with a 2 MB video (log scale) 79
A.44 Download and upload throughput for various locations with a 2 MB video (log-log scale) 79
A.45 Download and upload time for various locations with a 4 MB video (log scale) 80
A.46 Download and upload throughput for various locations with a 4 MB video (log-log scale) 80
A.47 Download and upload time for various locations with a 8 MB video (log scale) 81
A.48 Download and upload throughput for various locations with a 8 MB video (log-log scale) 81
A.49 Download and upload time for various locations with a 16 MB video (log-log scale) 82
A.50 Download and upload throughput for various locations with a 16 MB video (log-log scale) 82
A.51 Download and upload time for various locations with a 32 MB video (log scale) 83
A.52 Download and upload throughput for various locations with a 32 MB video (log-log scale) 83
A.53 Download and upload time for various locations with a 50 MB video (log scale) 84
A.54 Download and upload throughput for various locations with a 50 MB video (log-log scale) 84
A.55 Download and upload time for various locations with a 100 MB video (log scale) 85
A.56 Download and upload throughput for various locations with a 100 MB video (log-log scale) 85

Chapter 1

Introduction

In today's society, social interactions are an integral part of daily life, and millions of users turn to social networks to interact with their friends and family. the popular Online Social Network (OSN) platforms utilize a centralized server architecture, where the data is processed and stored in the providers' servers. As a result, these providers have control of any data that is uploaded and the user is forced to trust the provider with his or her data and privacy. Since the providers have access to the data, they can use this data to filter advertisements based on the user's actions, or even directly transfer the data to a third party, violating the user's privacy [4].

Privacy breaches are another major concern for OSNs with a centralized platform. All of the user's data and personal information is stored in one central location, and it is the responsibility of the OSN provider to protect the data. Within the last year, Facebook suffered a breach in the user's privacy when a loophole was exploited to obtain thousands of names, pictures, and locations of users who linked their mobile phone number to their account [23]. The Facebook Application Program Interface (API) was used to query the data of different user accounts, even though the users set their data as private. Facebook was alerted twice about the issue before a response was given that stated it was not considered to be a security vulnerability, but they do monitor abuse of their APIs. As a user, this creates concern whether the OSN providers are actually concerned with protecting their users' privacy.

An alternative to the centralized OSN platform is a decentralized peer-to-peer architecture, which would allow the user to be more in control of his or her data and who can access it. This architecture requires a system to store the data (considered to be untrusted) so all the OSN users can access the data at any time. To guarantee the confidentiality of the data and the privacy, encryption schemes are commonly used. This type of architecture has been the focus for research efforts, and many different encryption schemes such as attribute-based encryption (ABE), predicate encryption (PE), broadcast encryption (BE), and symmetric/asymmetric encryption have been utilized in the different platforms. Each encryption scheme has its own drawbacks and benefits depending on how the data is stored and accessed. The primary goal for a decentralized architecture is to deliver the content in an efficient and timely manner, and to have minimal overhead to the users to maintain their content for regular OSN features [31].

This paper presents POSN, Personal Online Social Network, which is implemented as an Android application for that utilizes cloud storage and encryption to create a privacy preserving decentralized peer-to-peer online social network. The cloud is considered to be untrusted, therefore all of the data stored is encrypted using either symmetric or public/private key encryption schemes to allow for access control and data confidentiality. The POSN architecture also removes any third party servers used for authentication and access control in order to give the user full control over their data, and ensure that interactions happen only between the desired friends. The main contributions of thesis are summarized as follows:

- A feasibility study has been carried out to measure the performance of popular social networks: Facebook and Google+ and cloud storage providers: Dropbox, Google Drive, OneDrive, Mediafire, and Copy Cloud, in order to see whether clouds can provide enough communication efficiency to deliver the content to the users. An Android application has been developed to measure the upload and download time of a variety of different sized files using the different social network and cloud provider APIs.

- The design and architecture of POSN is described by including a discussion on how the files are organized and encrypted in order to guarantee the user's privacy and provide access control. A discussion on the implementation of the Android application is provided, including the different functionality that is implemented into POSN to carry out the traditional OSN functionality.

The remainder of this document is structured as follows, Chapter 2 covers the social network background, challenges in decentralized OSNs, similar studies in decentralized OSN, and the libraries and APIs used for the implementation of the cloud measurement and POSN applications. Chapter 3 presents the cloud measurement application, measurement process, the result and analysis, and related work in cloud measurement studies. Chapter 4 discusses the high level design overview of POSN. Chapter 5 describes the implementation details of the functionality and interface in the POSN application. Chapter 6 gives a summary of the presented work. Finally, Chapter 7 discusses some possible ideas for future enhancements to both applications.

Chapter 2

Background

2.1 Social Networks

An online social network (OSN) is an online platform that facilitates the building of social relations and interactions with people who share common interests, experiences, or personal relationships with the user. Each user in the network builds a public profile and creates connections with other users in the network. The user can share information and content within their circle of friends, or out to the public depending on their privacy preferences. OSNs commonly support many other features such as instant messaging, groups, and third party applications where users can interact and collaborate with their friends and others [10].

2.1.1 Centralized Architecture

The most popular social networks such as Facebook, Google+, and Twitter, use a centralized server architecture to provide service to their users. Figure 2.1 shows an example of how users interact in a social network with a centralized architecture. In this architecture, a social network provider creates and maintains servers that provide access to the social network through a client application (website or mobile app), and holds all of the personal data for each user. The user's client communicates with the server(s) to receive the data the user requested. When a user wants to add new data to the social network, the user has to upload it to the provider's server(s) before it can be shared with other users. This scheme forces all of the user content stored to be stored in a centralized repository and it is under the control of the social network provider, therefore putting the user's privacy at risk.

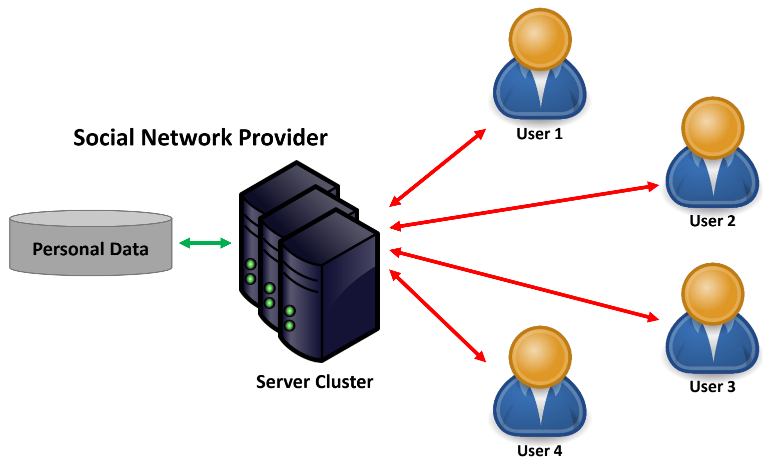


Figure 2.1: Example of a centralized social network architecture

2.1.2 Decentralized Architecture

A social network that utilizes a decentralized architecture moves the information away from a centralized repository, and moves it to a distributed system of trusted servers or to a peer-to-peer system. Figure 2.2 shows an example of how the users in a decentralized social network interact and how the data is stored. In this architecture, there is not a single provider, but instead a set of users in the network. Since there is no central authority, each user is responsible for maintaining and storing their own data, as well as providing computing power to carry out the OSN functionality. This has some benefits such as control of the content and privacy, but there are some additional challenges that will be discussed in the next session. This architecture also has the benefit of significantly lowering the cost of providing the service due to each user being responsible for his or her own resources, whereas the centralized provider must supply all of the resources and hence needs to generate revenue.

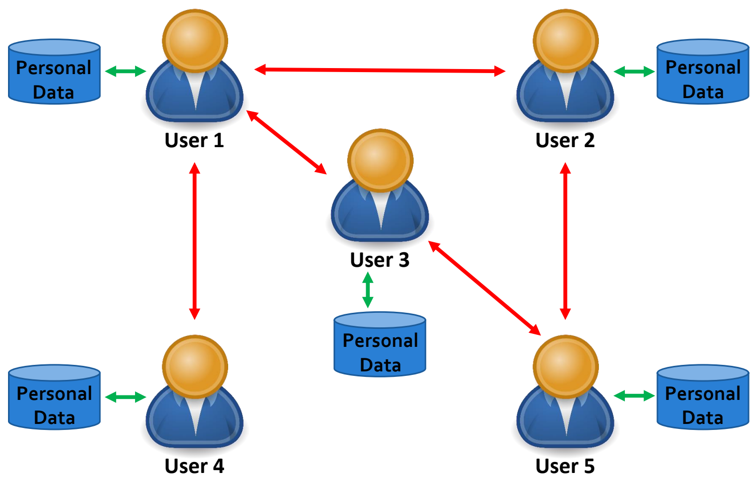


Figure 2.2: Example of a decentralized social network architecture

2.1.3 Challenges

By moving away from a centralized provider, decentralized social networks give more control and privacy to the user, but have several challenges that must be addressed. Many of these challenges have a trade-off between the amount of user privacy and the functionality and efficiency of the platform. In the section below, several challenges in a decentralized architecture are discussed along with how each are handled in the POSN architecture.

Storage and Availability

In a decentralized OSN, the data is not stored in one single repository. As a result, some applications store the data in a distributed manner, such as in a distributed hash table, trusted storage nodes, or at the peers directly. By distributing the data, there is not a single point of failure and it is significantly less susceptible to an attack. The downside, however, is that the users have to store the data on their cost, but this can be alleviated by using free cloud storage.

Another challenge with storing the data is the availability of the content. Peer-to-peer systems typically require a peer to be online to exchange data, which can impact the availability if the peer is offline. To overcome this issue, many decentralized OSNs utilize a distributed hash table (DHT), or a replication scheme, where a user's content is given to many different peers. Replicating the data puts a burden on the peers by having them store additional content, and the platform must be dynamic enough to choose which peers should be selected to have the optimal availability. In POSN, free cloud storage is used so that each user can store their own data and the clouds are always available since the cloud providers have consistent uptime and data replication mechanisms in place.

Privacy and Access Control

Decentralized OSNs focus on providing more privacy to the user and control over their content. Data privacy has two main factors: confidentiality and integrity. Data confidentiality means the data is protected against unintended or unauthorized access by friends, adversaries, or storage nodes. Data integrity refers to the data being accurate and it has not be tampered with by anyone, except authorized users. To guarantee the user's privacy, cryptography mechanisms can be used to encrypt all of the data that is stored and transmitted in the network. POSN utilizes symmetric and asymmetric encryption to encrypt all of the user data stored on the device and in the cloud.

Access control is another challenge in decentralized OSNs. The user should be able to select which friends or groups to share content with, and only those friends should be able to access that content. The access control mechanism also needs to be flexible so a friend's access can be changed or removed altogether. Access control can be created by having a file hierarchy and using encryption keys to manage the different friends' access. The mechanism needs to be as efficient as possible to handle a high churn rate and manage various keys to provide fine-grained access.

Friendship Establishment

Centralized OSNs typically have a search feature to find friends who are also using the network. Decentralized OSNs on the hand, focus on providing privacy and tend to be more careful with even revealing the existence of users. This makes finding friends a challenge and creates a trade-off of whether user's privacy is more important or having a search functionality to make the process easier. Some alternatives would be to have the users be registered in a directory service where it can be queried for a friend, or an email/SMS message can be sent to specific friends. The latter requires the user to personally know the friend and acquire their personal information. As a personal social network, POSN requires the users to have the email address of the desired friend to find other users in the system.

Another challenge is how to determine if the prospected friend is actually the person the user thinks he or she is. To overcome this issue, an additional verification process can be added to prompt for information that the friend would only know, such as an email address or phone number. Another alternative is to use encryption keys to encrypt the data so only the desired users can decrypt the data and validate that information. In POSN, a three step friend establishment process is used that relies on email identities to validate the friend.

Social Challenges

A friendship might not last forever or the level of friendship might change, resulting in a change of what content is shared with that friend. The platform should be able to handle both changing what content is being shared, or to completely remove a friend from the user's circle. To change the level of information shared with a friend, the group(s) that the friend belongs to could be updated to add or exclude them from the group(s). When removing a friend, the user may not want the friend to be aware that they were unfriended. The friend can be removed from all groups by establishing new groups with new encryption keys. This prevents the friend from accessing new content and they would not realize existence of new posts.

Another social challenge is online stalking. In order to stalk someone in a decentralized OSN, the stalker is required to know the victim in real life and become their friend on the OSN. Because of this, if the victim realizes the stalker's activity, they can remove them from their network. If the stalker is a friend of the victim's friend, they can only see the comments made by the victim, which are often limited. Since the user content is encrypted, it is not possible for the stalker to view the victim's personal content.

2.2 Libraries and APIs

The following is a description of the libraries and APIs used for the implementation of both the cloud measurement and POSN Android applications.

2.2.1 Android Operating System

Both the cloud measurement and POSN applications are implemented to be used on the Android 5.0 Lollipop operating system [19]. Each application has been developed using the Android Software Development Kit (SDK) and the Java programming language. The Android operating system provides a flexible platform, while supporting a wider range of smartphones and tablets compared to other mobile operating systems for a low development cost. It was also chosen for its extensive support of built-in libraries to handle network communication, encryption, JSON formats, and file management. Many cloud providers have developed their own Android specific libraries to connect the application to their APIs to access their services.

2.2.2 Social Network Connectivity

Facebook

To carry out the Facebook measurements in the measurement application, the Facebook SDK [18] is used. The Facebook SDK provides an easy-to-use interface to integrate Facebook functionality into third party Android applications. It allows for users to be authenticated by using their credentials and for interactions with the Facebook social graph to be done by downloading and uploading files. Each application that connects to Facebook is required to be registered in the developer panel section of the developer's account. Once the application is registered, an application ID is created and the application's access permissions also need to be configured in order to use any of the Facebook API functionality and authentications. This application ID needs to be added to the application's manifest file.

Google+

Google+ uses a separate service called Picasa Web Services to handle the user's photos and video content for their Google+ account. This service can act as a standalone service or be integrated into third party applications to view, update, download, and share photos. The cloud measurement application only needs to access photos and videos on Google+, therefore the Picasa Web Services Java API [21] is used to implement that functionality. The API only requires the user's email address and password to authenticate the user and obtain access to his or her files. It is important to note that Picasa Web Services has been phased out in February 2016 and is replaced with Google Photos.

2.2.3 Cloud Storage Connectivity

Dropbox

Dropbox provides an Android Core API SDK [13] that has simplified Dropbox integration by providing wrapper functions to directly interact with their API instead of calling raw HTML calls. Both the measurement and POSN applications utilize the SDK to authenticate users to their Dropbox accounts and provide methods to download file, upload files, and manage directories. Each application that connects to Dropbox is required to be registered in the Application Console in the developer's Dropbox account. An application key and secret is generated upon registration, and these values are used to link the application to a user's account. These values need to be added to the application's manifest and used when the application generates an access token. The application's permissions also need to be specified, as well as what Dropbox folders the application can access. Figure 2.3 shows the authentication process for Dropbox [14].

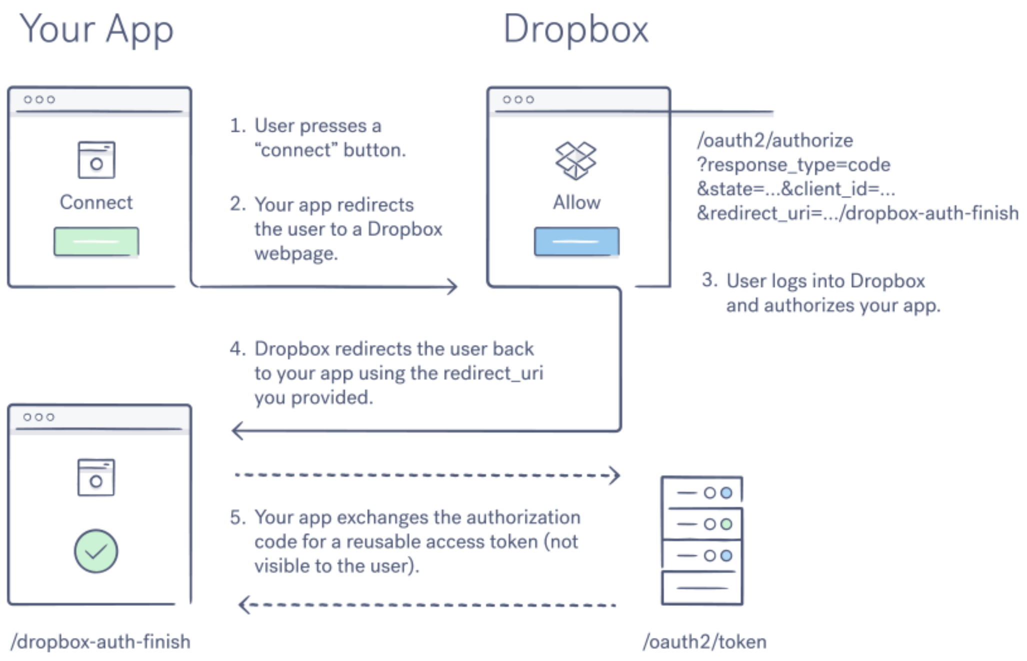

Your App
Dropbox
Connect
Allow
1. User presses a "connect" button.
2. Your app redirects the user to a Dropbox webpage.
/oauth2/authorize ?response_type=code &state=...&client_id=... &redirect_uri=.../dropbox-auth-finish
3. User logs into Dropbox and authorizes your app.
4. Dropbox redirects the user back to your app using the redirect_uri you provided.
5. Your app exchanges the authorization code for a reusable access token (not visible to the user).
/dropbox-auth-finish
/oauth2/token

Figure 2.3: Example of the Dropbox user authentication process

The Dropbox SDK allows for direct file paths to be specified to access files. The file path can be passed to the functions to download or upload files, for example: "/folderA/folderB/filename". This scheme removes any requirements to find a file ID or query an entire folder to locate files.

Google Drive

In order to interact with Google Drive, the Google Play services SDK is utilized to directly interface with the Drive API [20]. The Drive API simplifies the interactions by abstracting many of the complex tasks and treats the process of reading and writing files to Drive like a local file system. The application is required to be registered in the Google Developers Console in order to get a signing certificate. The Drive API and Drive SDK must be enabled for the application, and a OAuth 2.0 client ID must be created to make authorized requests in the user's account. The OAuth ID requires the developer's SHA-1 fingerprint in order to issue the certificate. Some additional permissions are required to be added to the application's manifest file in order to have the user sign in with their Google Account to access their Drive files.

The Drive API uses file and folder IDs to interact with content stored in the cloud, therefore, the ID must either be stored or queries must be created every time to fetch the folder and file IDs. Once the appropriate file ID is found, the file can be downloaded, or the folder ID can be used to upload files. This method of fetching the IDs adds additional overhead to the download and upload process and needs to be considered when designing the applications.

OneDrive

To interface with the OneDrive storage cloud, Microsoft provides access to the OneDrive APIs through the Live SDK for Android [29]. The SDK provides an easy-to-use interface to connect to OneDrive without the need to call the REST APIs directly. Similar to the other APIs, the Android application needs to be registered in the Microsoft account Developer Center to obtain a client ID that is used to send authenticated requests. The user also needs to supply their credentials to log into their OneDrive account.

Similar to Google Drive, OneDrive also uses ID numbers to keep track of the files and folders in the cloud. In order to get the file ID, the API requires a query to be created to list all of the files in a specific folder. The list of files can be searched to obtain the correct file ID and then the file can be downloaded. The upload function only requires the folder ID of where the file should be uploaded to and can be obtained by querying for the folder ID. These IDs can be stored to minimize the amount of queries and searching required to locate them.

Copy Cloud

Copy Cloud is a cloud storage service from Barracuda Networks. It is important to note that Copy Cloud will discontinue its services starting on May 1, 2016 and will not be accessible. An official Android SDK to directly interface with the Copy API [3] is not provided, but the API can be accessed by using raw HTML calls to their REST

API. To carry out these HTTP requests, the Android Asynchronous HTTP client [35] library is utilized. Details about the Asynchronous HTTP client is provided in the Section 2.2.4. Copy Cloud requires that the third party application is registered in the Copy Developers console in order to set the application permissions and obtain a consumer key and secret. Copy Cloud requires the user to log in to their account in order to get an access token using the OAuth Authentication flow. The process of getting this token requires the consumer key and secret to be provided during a three-way handshake with the authentication server. Once the token is received, API calls can be sent to access the files.

Accessing the files in Copy Cloud is similar to the direct file paths that Dropbox uses. Files can be downloaded by using the file and folder names to identify which file to download. Uploading files requires the name of the folder to be provided as well as the file content in a binary format. The binary file contents are required to be sent as a multipart file in order to upload correctly.

Mediafire

Mediafire is a free cloud storage provider that focuses on an easy-to-use system to host and deliver content. Similar to Copy Cloud, Mediafire does not provide an official Android SDK to connect to their Core API. Therefore, the same Android HTTP client library is used to make calls to the Mediafire's REST API [27] on Android applications. To send requests to the API, an access request token must be obtained by sending a special request with the user's username, password, application ID, and API key. Each application must be registered in the Mediafire Developers Console to obtain the application ID and API key. Once the access token has been generated, a second token, call an action token, is fetched which allows for download and upload API requests to be called.

Mediafire also uses IDs called quickkeys and folderkeys to reference different files and folders in their APIs. To download a file, the quickkey needs to be fetched by a query in the appropriate folder. Once the quickkey is found, another query is used

```
{ "posts": [
    { "type": 0,
      "friend": "ec3591b0907170cc48c6759c013333f712141eb8",
      "date": "Jan 19, 2015 at 1:45 pm",
      "content": "This is a test post."
    },

    { "type": 1,
      "friend": "c6b804d6f0b8164cb1c3fe6b50f5c88493216b86",
      "date": "Jan 20, 2015 at 3:12 pm",
      "content":"picture.jpg"
    }
  ]
}
```

Figure 2.4: An example of a JSON data structure definition for a list of wall posts.

to fetch a direct download link and the file can be downloaded. The upload process simply requires a folderkey to be provided along with the desired file contents to be given to the API request.

2.2.4 Android Asynchronous HTTP Client

In some cases, raw HTTP GET/POST requests need to be sent to a third party provider to interface with their APIs. The Android Asynchronous HTTP Client [35] library facilitates the process of sending and parsing network requests by providing robust and easy-to-use functions. All of the requests are carried out asynchronously and use callbacks to allow for multiple requests to be sent out simultaneously. All of the requests are sent outside of the main Android UI thread, but the callback logic will be executed on the same thread where the callback was created. The library supports JSON, binary, and file responses when the API reponds to the sent request.

2.2.5 JSON

JSON, or JavaScript Object Notation, is a self describing syntax that is used for storing and exchanging data. JSON objects store data as a collection of attribute-value pairs, and they can be serialized so that the data can be easily transferred across the network. Figure 2.4 shows an example of the JSON format. JSON is language independent so it can be utilized in many applications, including Android [6]. The POSN applications uses JSON to structure and easily parse the user data that is stored in files on the device and in the cloud. The application also needs to directly send user data across the network to another user's device, and the JSON format simplifies this process.

Chapter 3

Cloud Measurements

In this chapter, we present a measurement study of download and upload performance of the two popular OSNs, Facebook and Google+, to five of the popular storage clouds, Copy Cloud, Dropbox, GoogleDrive, Mediafire, and OneDrive. The focus of this study is to determine if cloud providers will be able to deliver files to end users as efficiently as OSNs. To determine if cloud providers are feasible, the file download and upload performance is measured and compared to OSNs. A discussion on the measurement application, procedure, analysis of the results, and related work is provided. With a lab partner, we also performed a measurement study of Facebook users to understand user interactions that guided POSN design [16].

3.1 Measurement Application

The measurement platform was implemented by developing an Android application that could measure the download and upload times of various photo and video sizes to different social network and cloud providers. The details of the SDKs and APIs used to integrate the different social network and cloud providers was discussed in Section 2.2, therefore, a brief discussion on the challenges that were encountered during the implementation is provided. In all cases, the time for token generation and directory listing were not measured, and measuring the actual file upload and download time was focused on. Similarly, to minimize external factors on the measurements, the application was forced to run in foreground, all unnecessary applications were terminated to minimize interference, and a small file was transmitted to ensure the device's radio was turned on prior to measurements.

Facebook

For the Facebook measurements, the implementation uses the Facebook SDK. The application requires the user to log in with their Facebook account so a session token can be obtained in order to access their profile and content. To minimize the stored file sizes, Facebook compressed photos as they are uploaded onto its platform, therefore the test files were initially uploaded to Facebook and downloaded. The compressed Facebook files were used to measure upload and download performances of all platforms so that the measurements would use a consistent file size. The different test files were placed into a single album on user's profile beforehand. It was not possible to obtain photos larger than 1 MB due to Facebook's compression, therefore videos were utilized to represent larger file sizes (which had a side issue with Google+, as described below).

Google+

The measurement application uses the Picasa Web Services Java API to implement downloading and uploading of files from Google+. Google+ also compresses photos as they are uploaded, but it was observed that there was an insignificant change in file sizes when Facebook-rendered photos are uploaded. One major issue with the Google+ measurements is that the videos cannot be downloaded as a single file, and they must be streamed as a constant download. After some exploration, it was found that this is a limitation of the current version of their API. This unfortunately adds a considerable amount of time to the download process of large videos.

Copy Cloud

Copy Cloud does not provide an official Android SDK to integrate their services into Android applications, therefore raw HTTP calls to their API were used. This API has some limited documentation on how to pass a file binary data to the upload call, but it was successfully implemented when the data was sent as a multipart file.

Dropbox

Dropbox provides an Android SDK that simplifies the process of integrating their services into the application. No major challenges or issues with their APIs were encountered.

Google Drive

Google developed the Drive API SDK to easily add the Drive services into applications. The API requires that every development tool signature for every developer who is working on the application is registered in the Developer console. One of the challenges with this API is that the files and folders use IDs that need to be queried before the files can be downloaded. This adds additional overhead to the download and upload process. To minimize this overhead, the download and upload folder IDs were fetched ahead of time, and only one query was used when downloading each file.

Mediafire

Mediafire also does not provide an official Android SDK, therefore calls were made to the Mediafire's REST API. The API requires an access token to be fetched before any download or upload calls can be made. One major drawback is that this token is only valid for ten minutes and needs to be refreshed often. The application has this token refreshed before every batch of file downloads and uploads, and was not included in the measurement timings. The API also requires the file ID to be fetched, and then the download link to be fetched using the file ID. This adds some unavoidable overhead to the measurement results.

OneDrive

The Microsoft Live SDK was used to implement the OneDrive cloud services into the application. OneDrive also uses file IDs to be used to access any of the files, so they must be queried. The query can only be done on a specific folder and lists all of the folder contents. This forces a search to be carried out to find the correct file and obtain its ID. This introduces some additional overhead, but was avoided in the measurement application by providing the specific download and upload folder IDs in advance. The query of the list of files in the download folder was only queried once before the measurement process was started.

3.2 Measurement Process

The tests were carried out by the Android measurement application over Wi-Fi or 3G/4G networks as described in each subsection. To compare the performance of the OSNs and cloud providers, the measurements were performed with various file sizes to simulate realistic social media usage. The file sizes included 2 KB to 718 KB to simulate photos and 1 MB to 100 MB to simulate video sharing. Facebook compresses pictures during re-rendering and we were not able to upload and then download a picture larger than 718 KB. Hence, 1 MB and larger files are videos doubling in size whereas below 1 MB are pictures resized by Facebook. All of the measurements were performed during the morning or early afternoon to avoid the heavier Internet traffic times.

WiFi measurements were carried out on a Samsung Galaxy S4 phone at nine different locations around the Reno, Nevada area. The different locations had either a public Wi-Fi connection or a private one. The locations with free public Wi-Fi include: an airport, a bookstore, a cafe, a public library, and a mall. The private Wi-Fi locations were three different ISP providers at different residential locations and a university campus. These different locations were selected to simulate a scenarios where people would commonly access social networks or cloud storage on their mobile devices.

Every file size had twelve measurements, where the minimum and maximum values were removed to exclude outliers and the ten remaining times were averaged together. A small file was downloaded and uploaded to each provider to remove any network latency (such as sleeping radio or DNS resolution) that would occur when the device first started a set of tests. The download order of the different file sizes and providers were randomly shuffled for each set of tests in order to reduce any effects of caching and any bias towards downloading and uploading all the files to one provider.

3.3 Wi-Fi Measurements - Varying File Sizes

Figure 3.1 and Figure 3.2 present the average download and upload times, while Figure 3.3 and Figure 3.4 present the throughput results for one of the specific residential locations (ISP 1). The download times for the photos increased as the size of the photo became larger, except in the case of Google Drive and Mediafire which had much smaller increases. The download bandwidth for the photos increased linearly as the size of the photo got larger. This linear increase indicates that the server is not the bottleneck, but that the network and other factors affect the performance. Google Drive and Mediafire both had lower throughput and slower times when downloading any of the photos compared to the other providers. The reason for this difference in performance is likely due to how the API for each provider requires two connections for one download, one connection to query to get either the file ID for Google Drive or a query for the download link in Mediafire, and a second connection to download the file. The other providers can download files using only one connection by using the actual file name to download the file. The download bandwidth for the videos had similar performance, except for Google+. Google+ showed a significant decrease in performance after 16 MB video file sizes. This performance hit is correlated to the limitation of the Google+ API where videos are downloaded as a stream rather than a single file causing an impact on the performance. The variance in download performance can be linked to the round-trip time to the provider's server because each server will be located in different geographic locations.

The measurements showed that Facebook had the best overall performance, while Mediafire had the worst photo performance overall. The providers again had a linear throughput increase for the photos, but then stayed at a constant rate for the video files. The constant rate can be correlated to reaching the maximum upload bandwidth

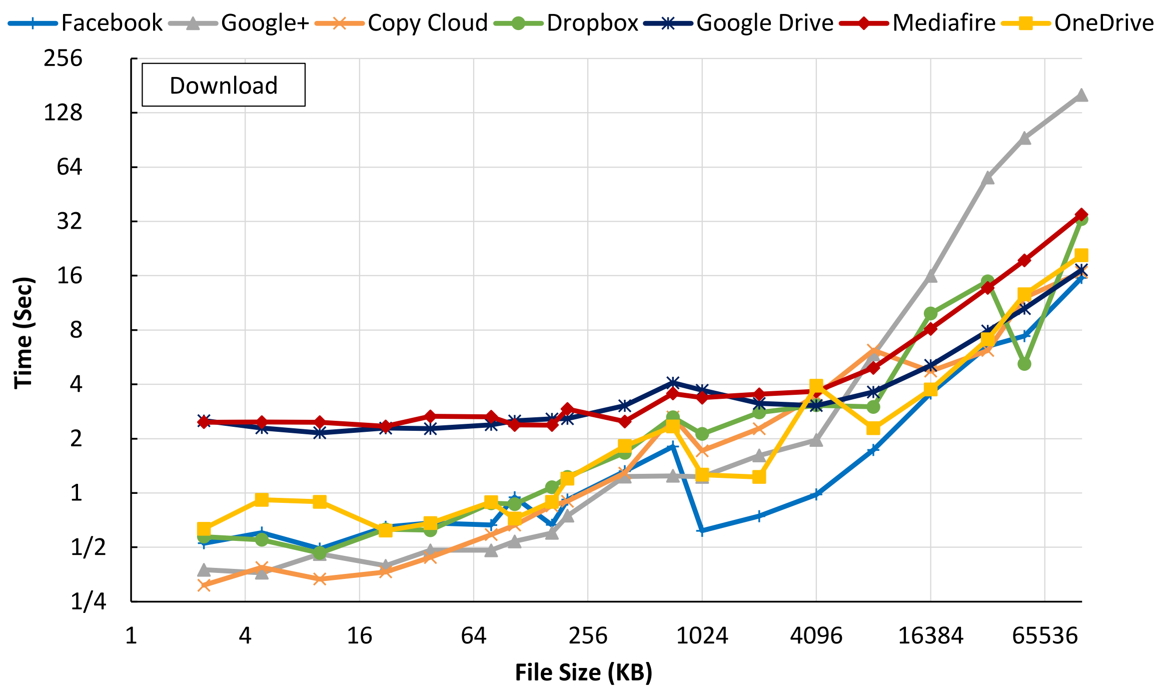


Figure 3.1: Download time for different file sizes at a residential location (ISP 1) (log-log scale)

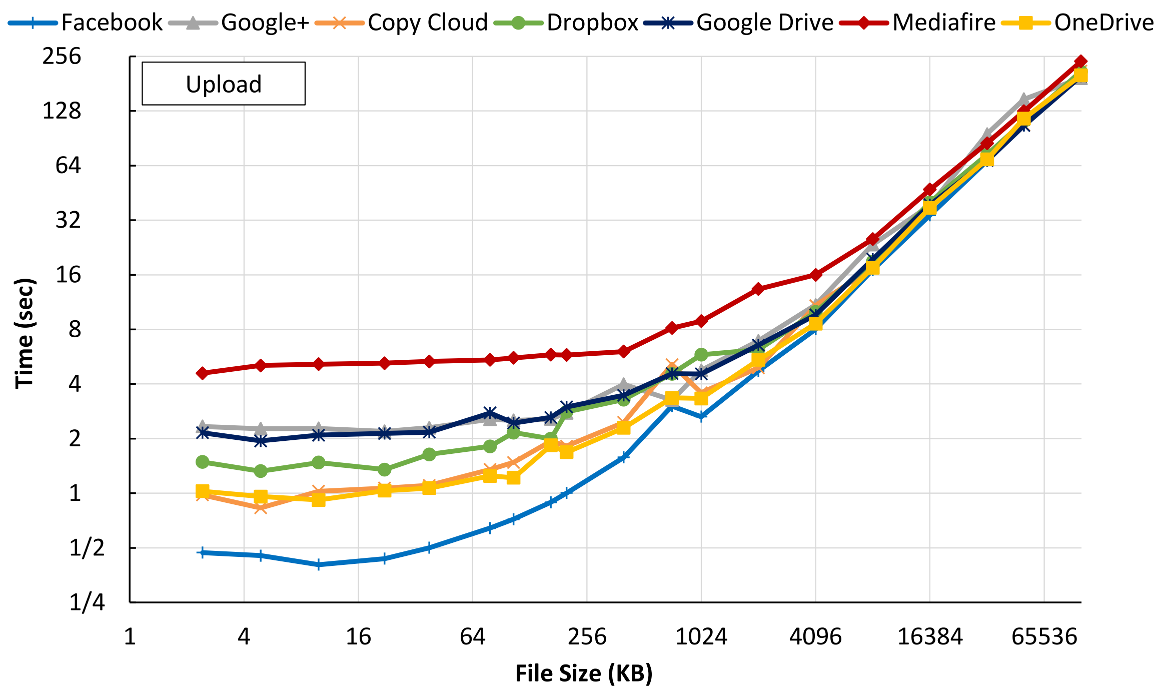


Figure 3.2: Upload time for different file sizes at a residential location (ISP 1) (log-log scale)

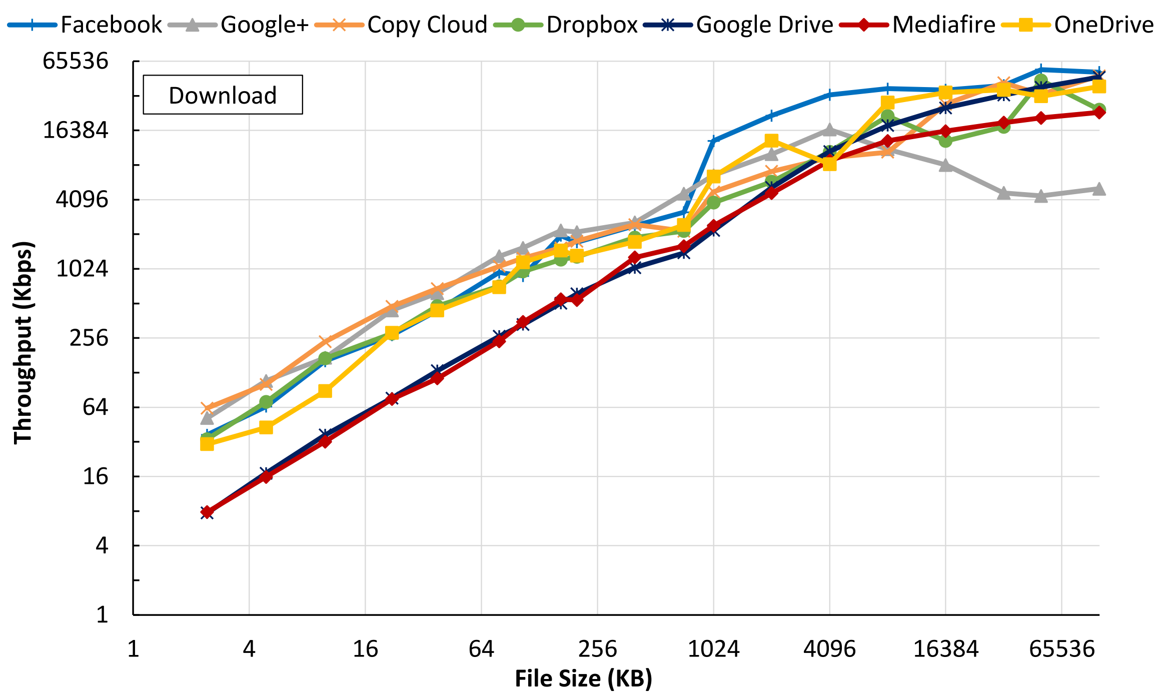


Figure 3.3: Download throughput performance for different file sizes at a residential location (ISP 1) (log-log scale)

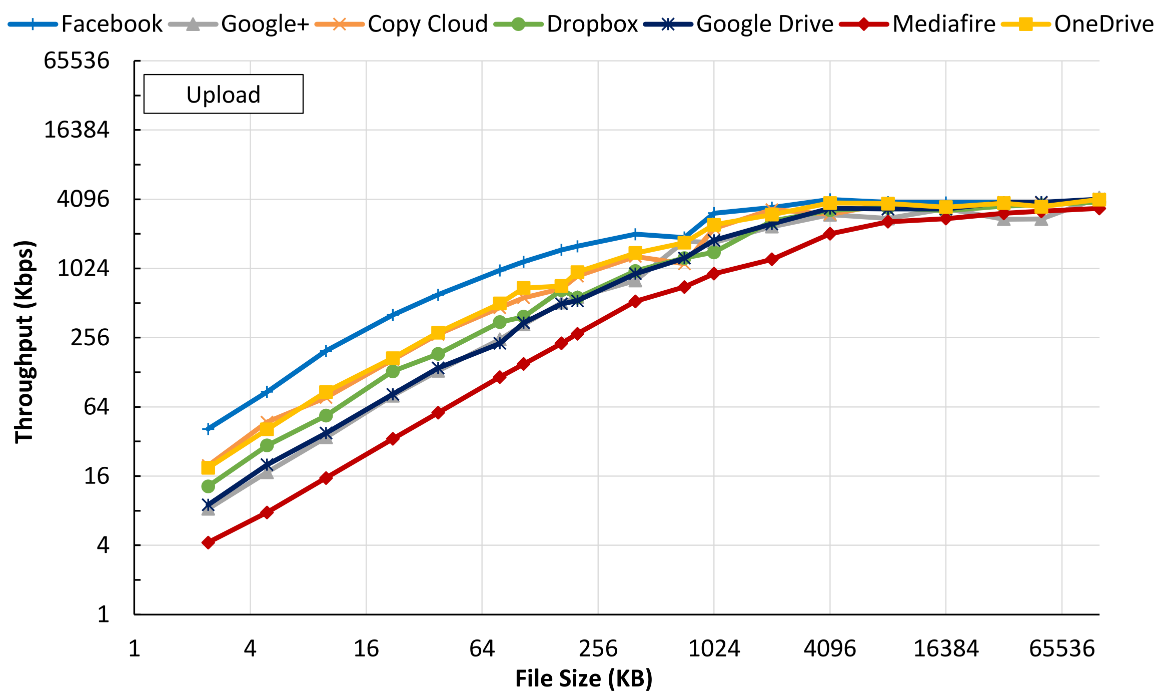


Figure 3.4: Upload throughput performance for different file sizes at a residential location (ISP 1) (log-log scale)

provided by the ISP. Additionally, similar results at the other eight locations were obtained for the file download and upload time and throughput, and they can be found in Appendix A. Overall, the upload performance was shown to be very comparable across the OSN and cloud providers.

Figure 3.5 displays the average download time and Figure 3.6 displays the average upload time for all the locations with different file sizes. Figures 3.7 and 3.8 present the average download and upload throughput performance for all the locations and file sizes. Copy Cloud, Dropbox, and OneDrive all had very similar download performances across all file types and sizes. Google Drive and Mediafire had the worst performance for files less than 1 MB, but then had similar performance with the other OSN and cloud providers for the video files. Facebook and Google+ showed very similar download performance except for the video files. When uploading the files, Facebook has the best upload performance for files less than 1 MB. The rest of the providers show similar performance, except for Mediafire. When the file size is over 1 MB, all providers had very similar upload performances. Overall, there was no significant performance difference when downloading and uploading a file to or from an OSN or a cloud provider.

The average time and throughput performance of the all OSN and cloud providers is compared in Figure 3.9 and Figure 3.10, respectively. Overall, the social networks out-performed the cloud providers for file sizes less than 1 MB. When the download file size is 1 MB or greater, the OSN and cloud providers are very comparable, but the cloud providers out-perform the OSN at 32 MB, likely due to the limitation of Google+ download streaming. The OSN upload performance also out-performs the cloud providers on average for the photos, but is similar for the videos. The performance difference of 1 second or 0.1 second for files less than 1 MB would not be significant for humans to observe in most applications.

All of the providers showed some interesting behavior as seen in Figures 3.1 through 3.10. As the file size goes from the largest image (718.43 KB) to the smallest video file (1 MB), there is a significant performance increase (increase in throughput

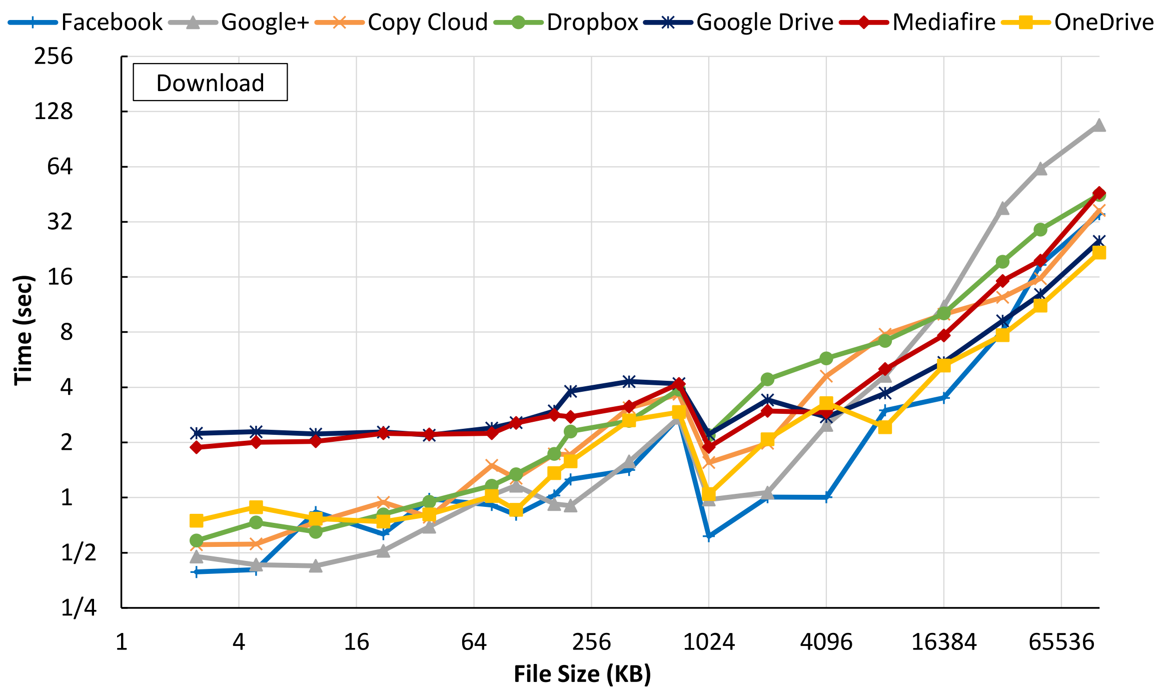


Figure 3.5: Average download time comparison for all locations with different file sizes (log-log scale)

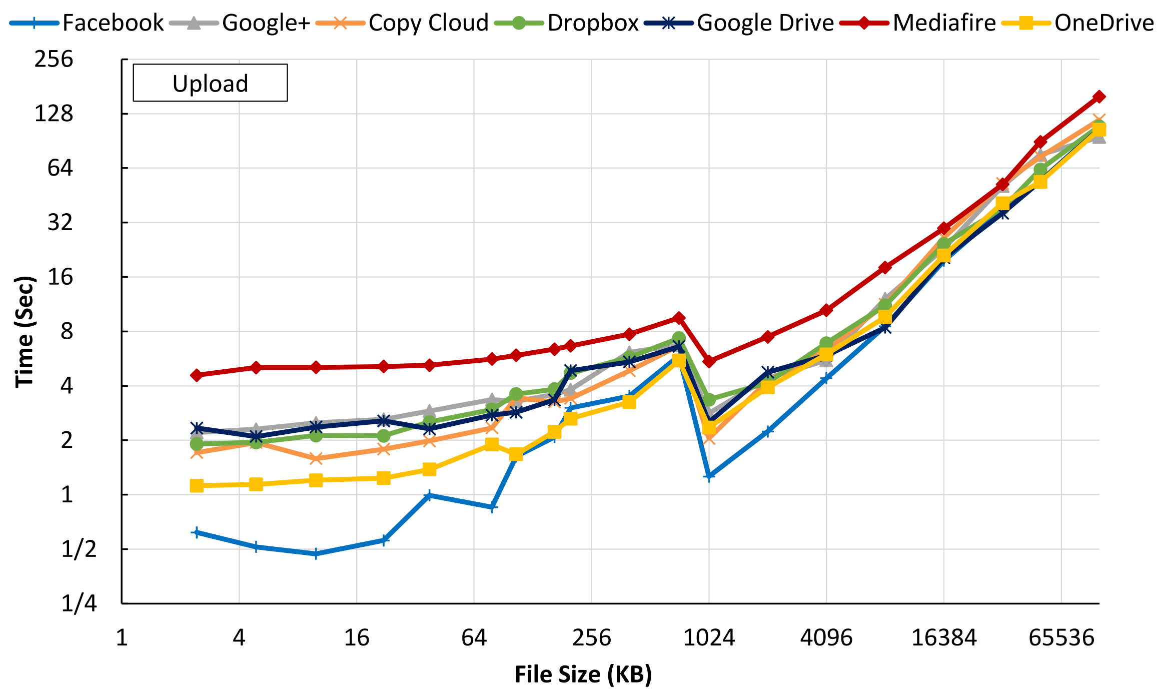


Figure 3.6: Average upload time comparison for all locations with different file sizes (log-log scale)

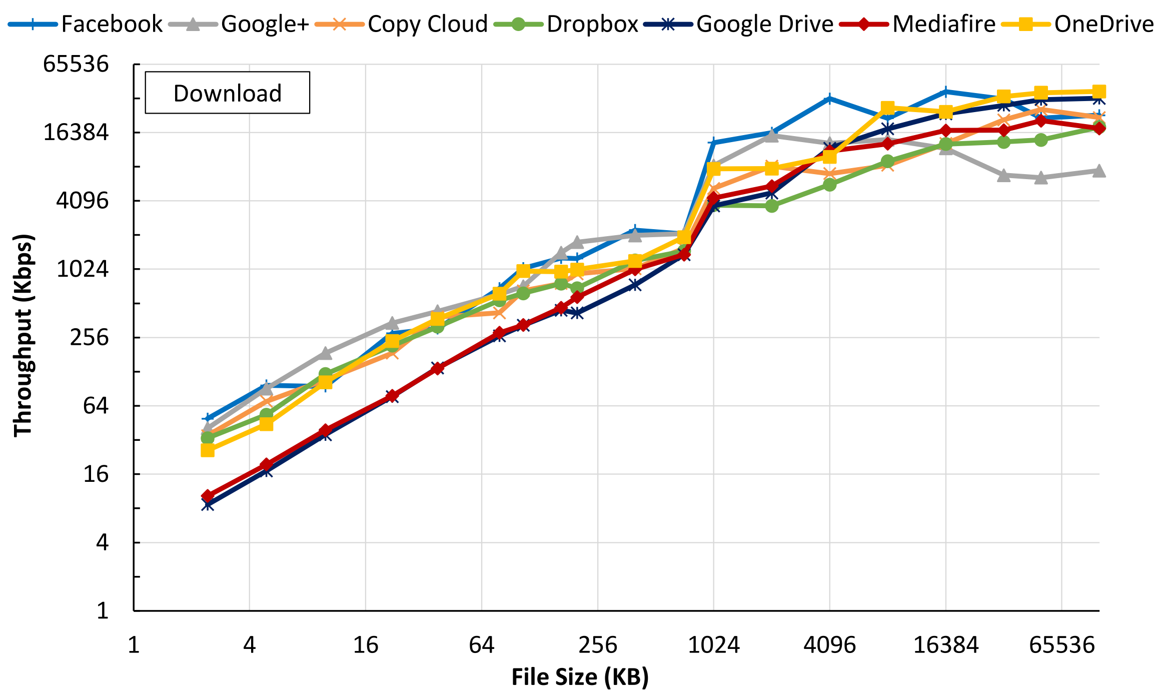


Figure 3.7: Average download throughput comparison for all locations with different file sizes (log-log scale)

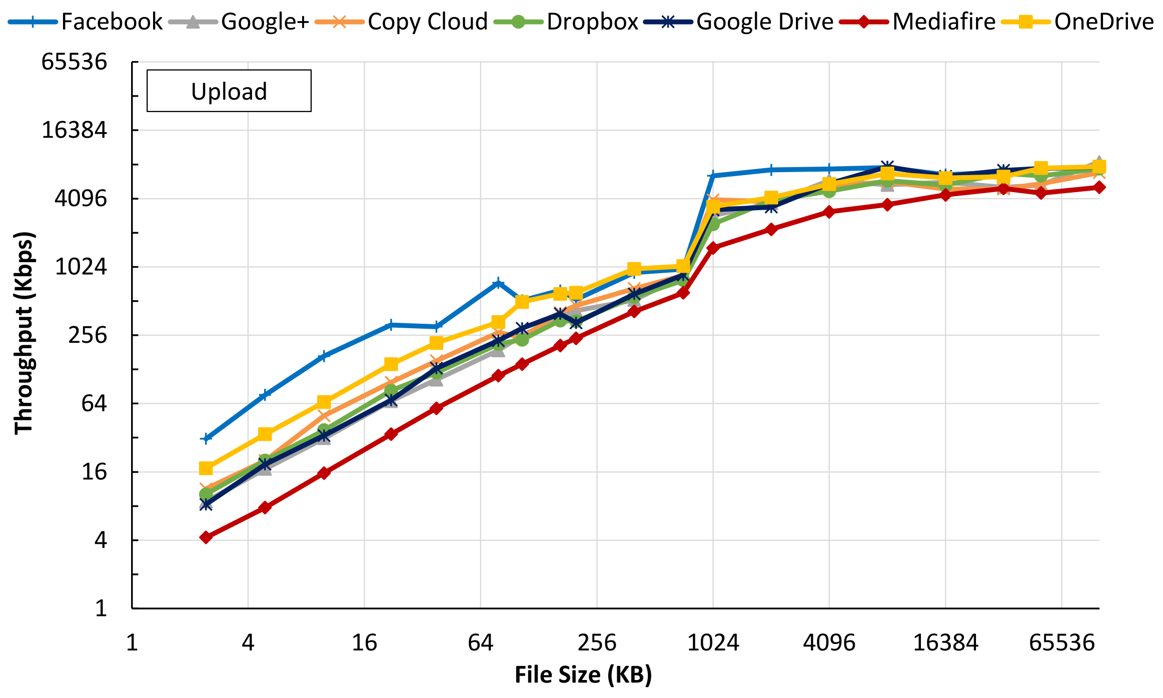


Figure 3.8: Average upload throughput comparison for all locations with different file sizes (log-log scale)

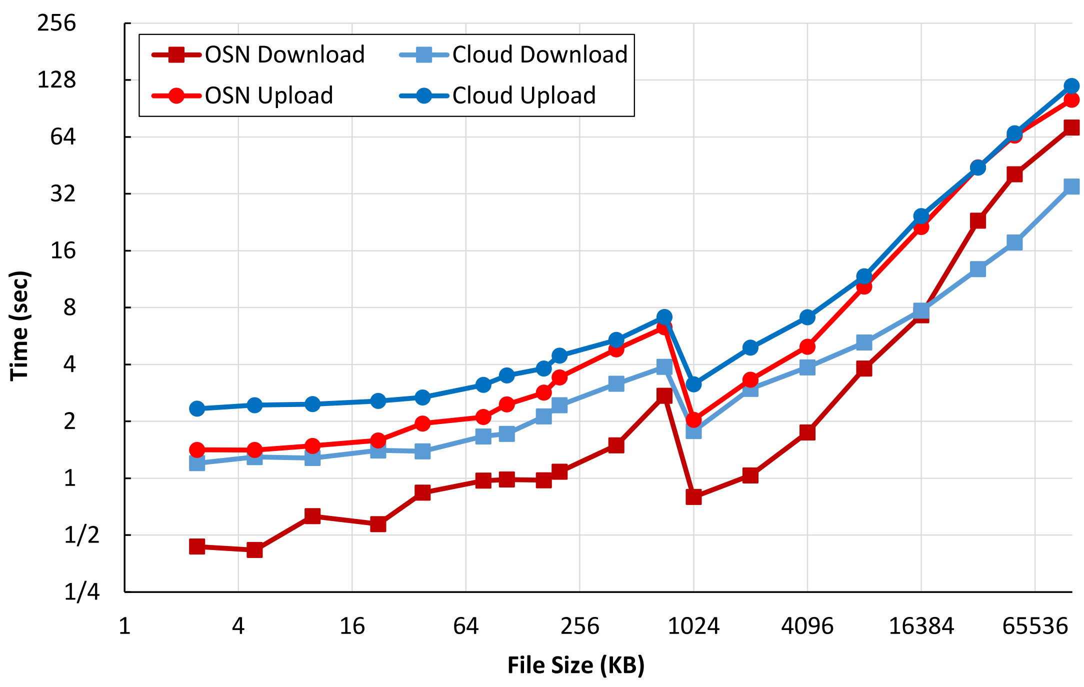


Figure 3.9: Average cloud and OSN time comparison with different file sizes (log-log scale)

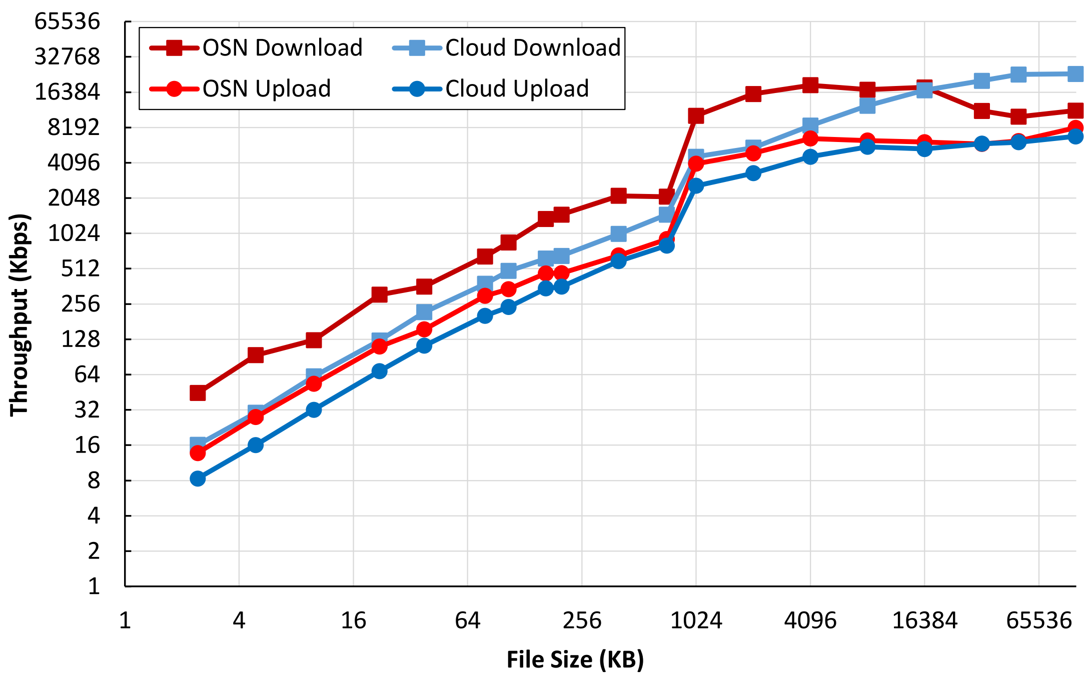


Figure 3.10: Average cloud and OSN throughput comparison with different file sizes (log-log scale)

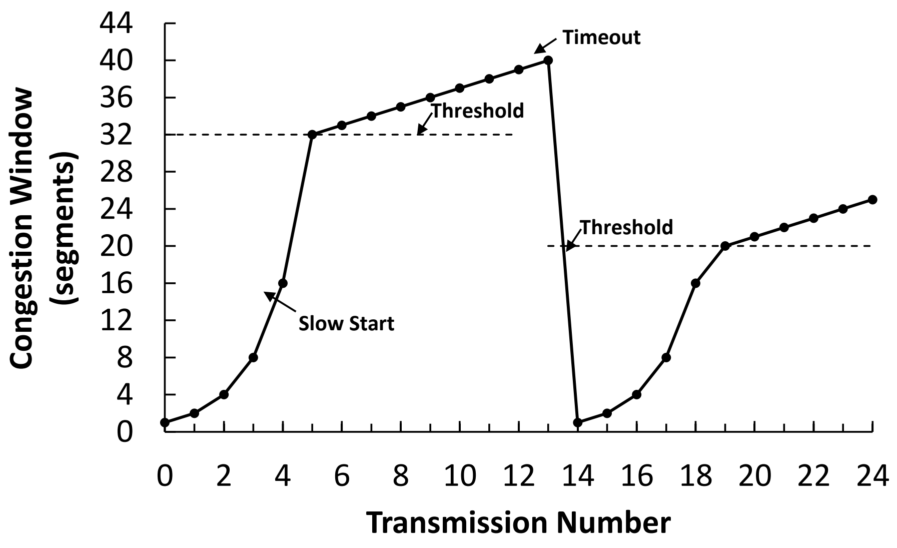


Figure 3.11: TCP slow start and congestion control process

or decrease in time). This performance change is likely due to TCP slow start and congestion control scheme and is shown in Figure 3.11. The process starts off by sending a few segments initially and waits for the handshake to occur. Once the acknowledgement is received, then more packets are sent, by an exponential amount, and the process repeats until the threshold is met. TCP takes awhile to achieve full steady state behavior and smaller sized files (less than 1 MB) may not even reach this state. When the file size is about 1 MB, then the TCP connection should have full bandwidth and outperform the smaller files. This should be considered when transferring data from the cloud to the users.

3.4 Wi-Fi Measurements - Different Locations

The download and upload time and throughput of the different locations for the largest image is shown in Figures 3.12 through 3.15. Figures 3.16 and 3.17 present the timing results and Figures 3.18 and 3.19 present the throughput results of a medium sized video. Results for all of the different file sizes were obtained, and can be found in Appendix A. It should be noted that similar patterns were observed for all figures. Each of the figures shown has the 95% confidence interval on each of the bars

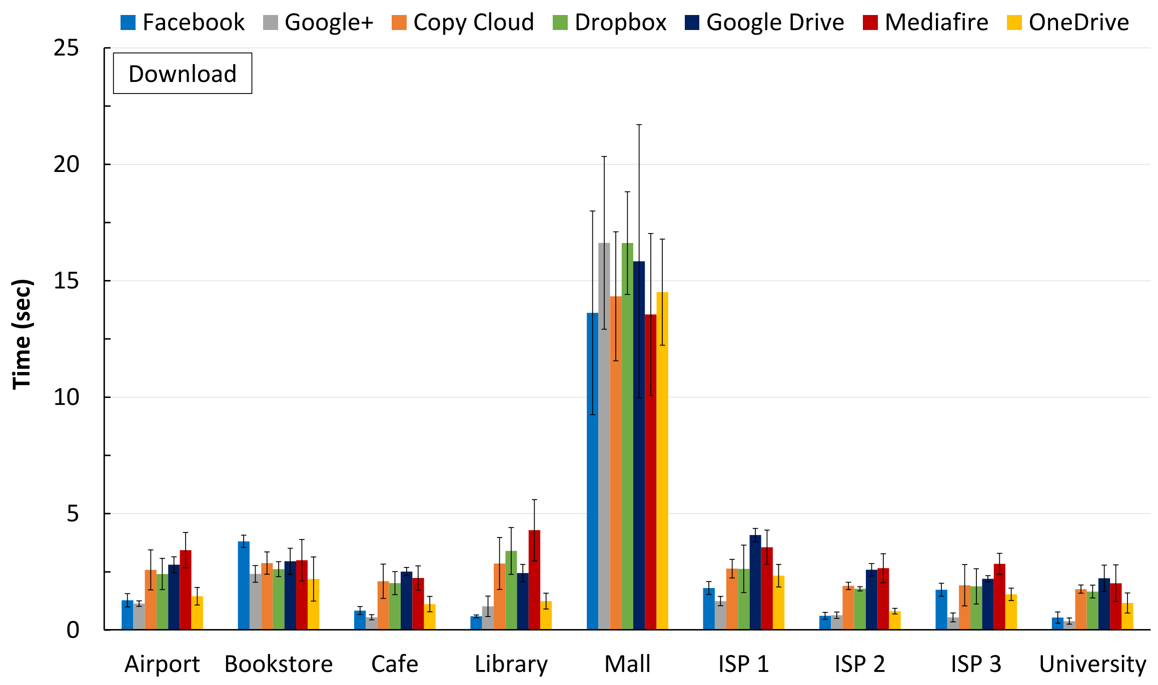


Figure 3.12: Download time for various locations with a 718.43 KB photo (log scale)

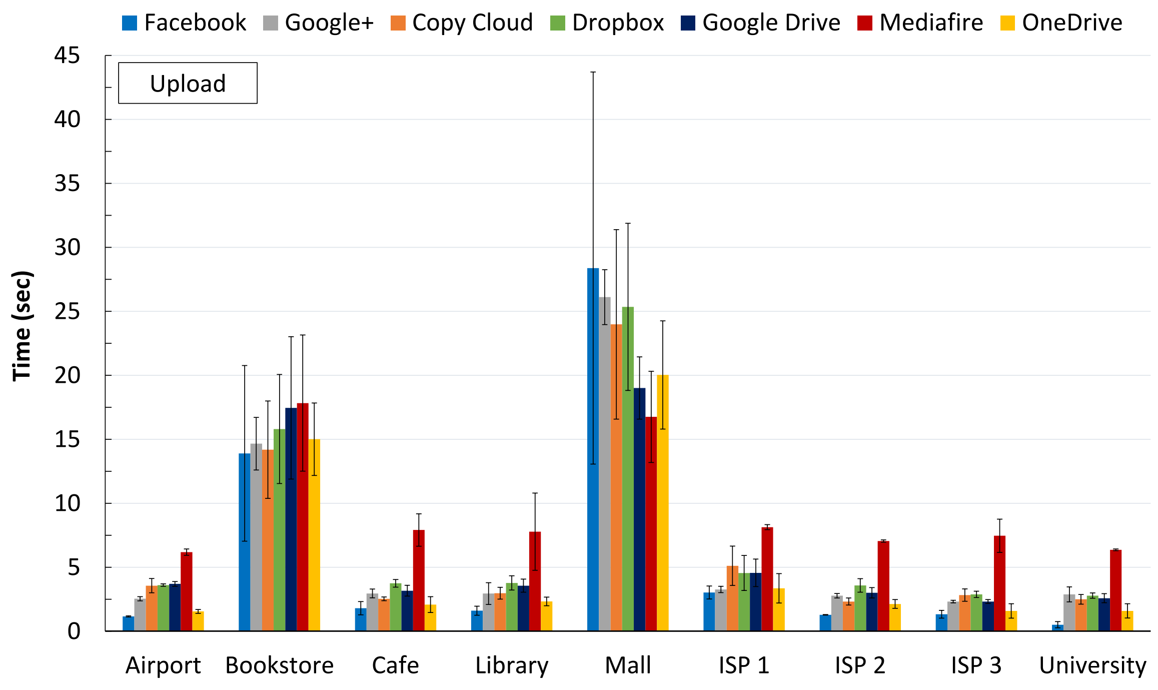


Figure 3.13: Upload time for various locations with a 718.43 KB photo (log scale)

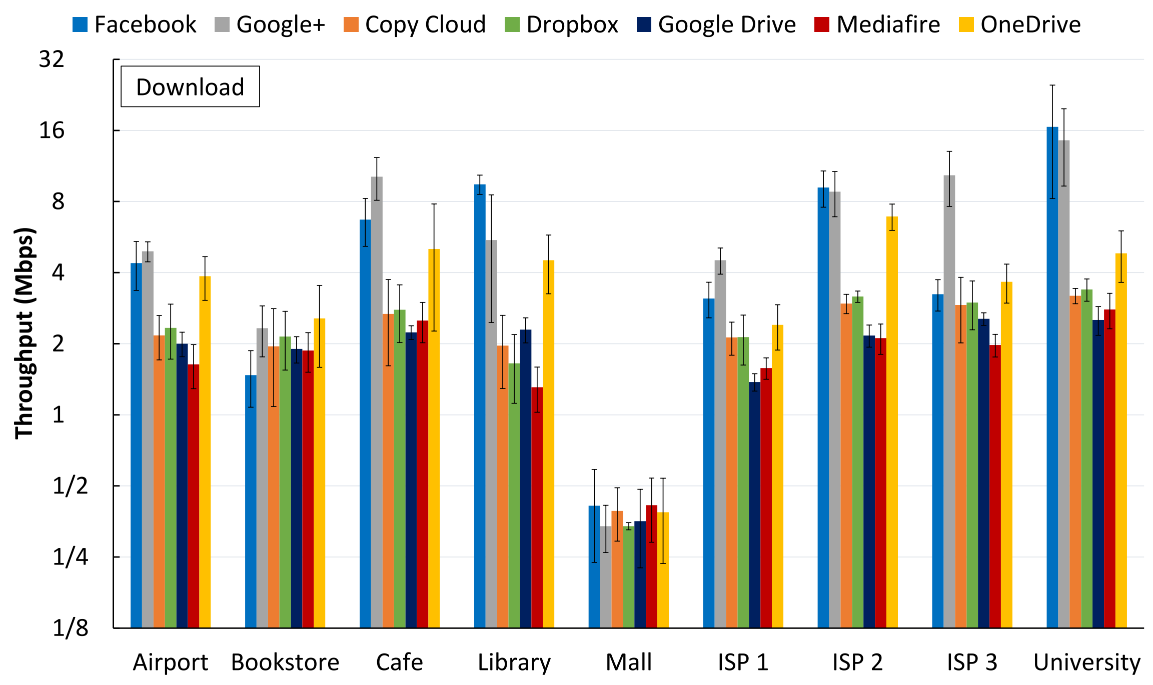


Figure 3.14: Download throughput for various locations with a 718.43 KB photo (log scale)

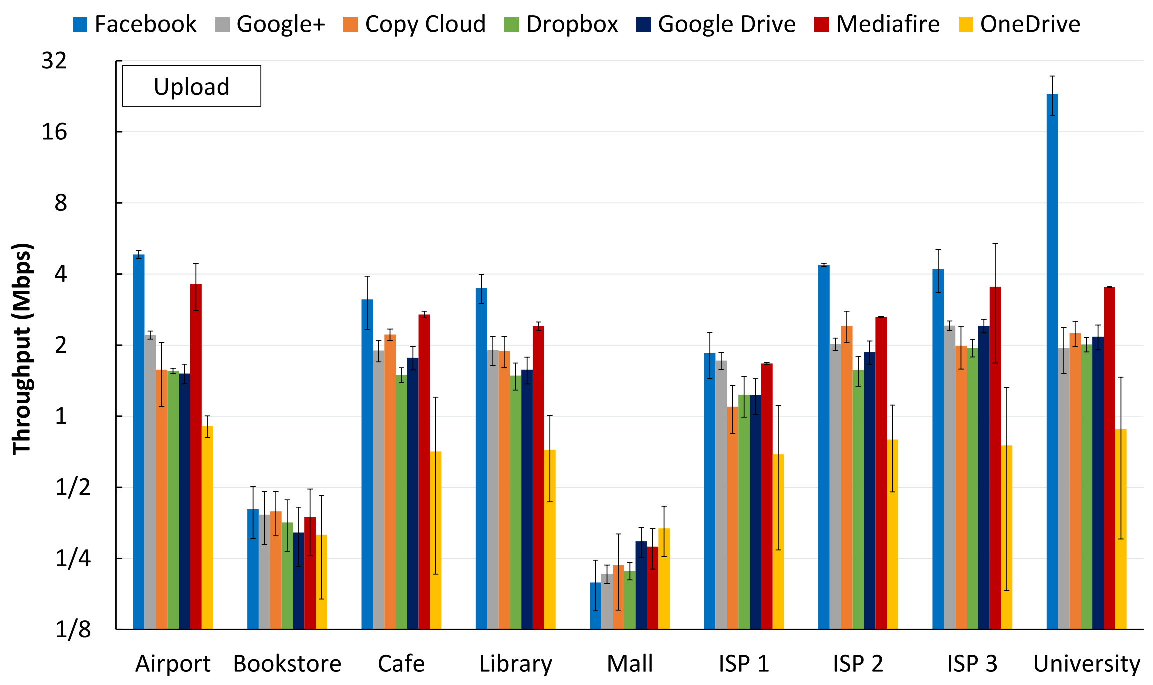


Figure 3.15: Upload throughput for various locations with a 718.43 KB photo (log scale)

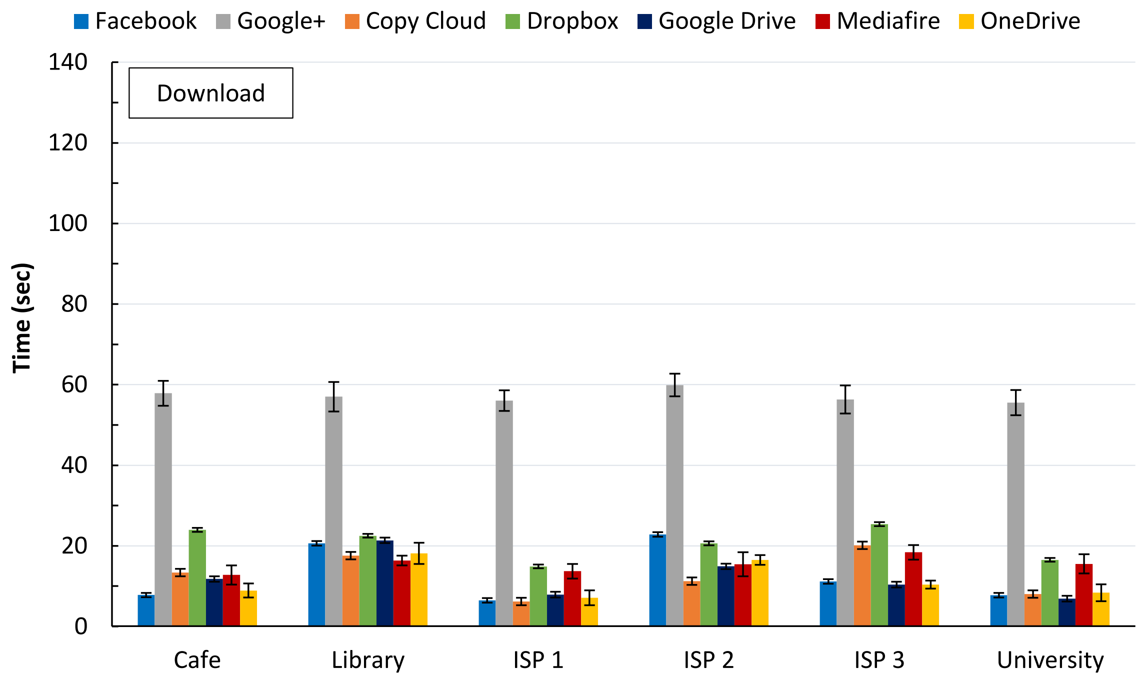


Figure 3.16: Download time for various locations with a 32 MB video (log scale)

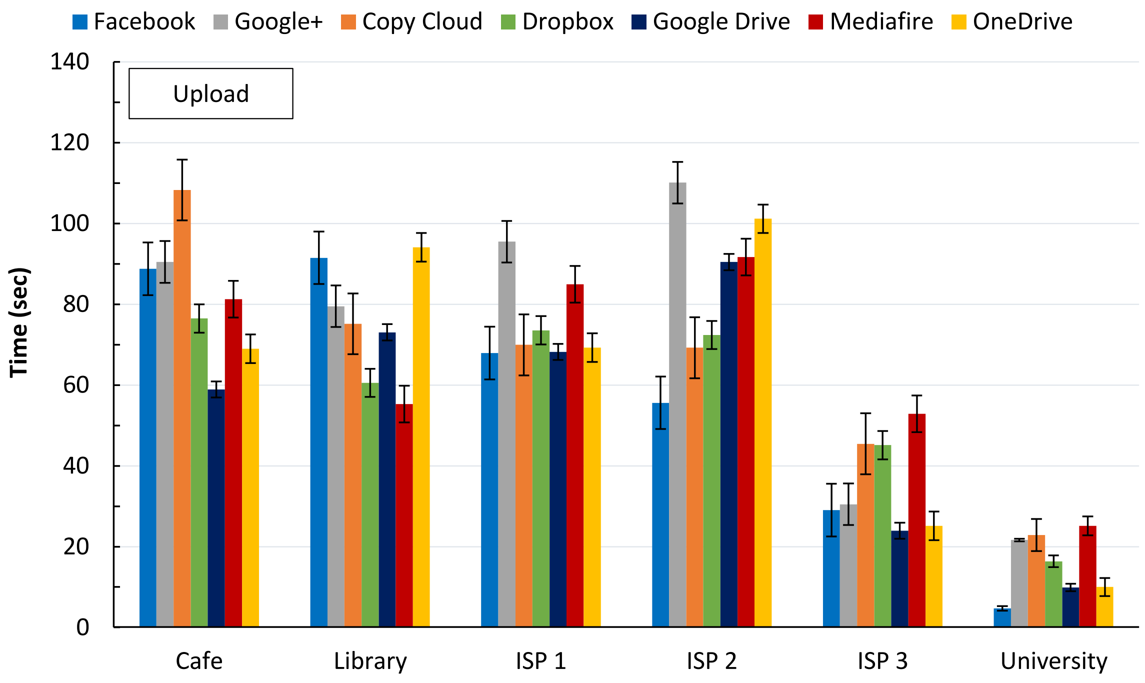


Figure 3.17: Upload time for various locations with a 32 MB video (log scale)

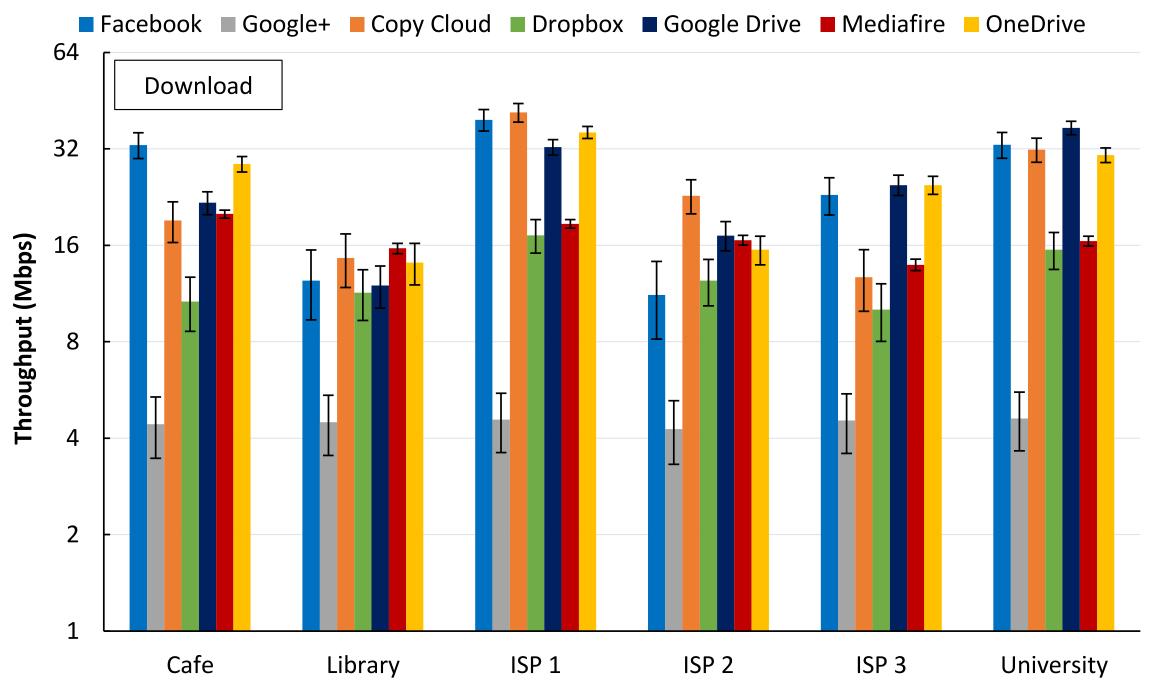


Figure 3.18: Download throughput for various locations with a 32 MB video (log scale)

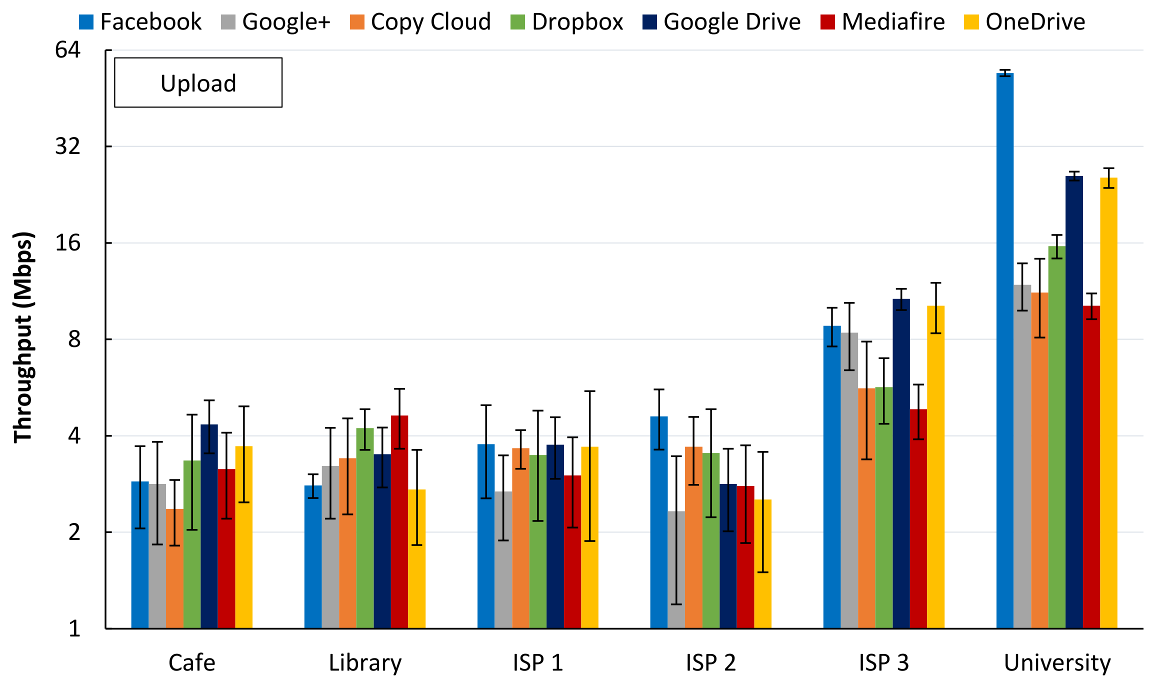


Figure 3.19: Upload throughput for various locations with a 32 MB video (log scale)

in black. As expected, the locations with public Wi-Fi had the worst performance when compared to locations with private Wi-Fi. This is due to the typically lower amount of bandwidth available to users. As a result, the video measurements could not be performed at the airport, bookstore, and mall locations and were omitted from the figures. For the images, the mall and bookstore showed the worst performance overall due to the very limited bandwidth available at these locations. In many of the public locations, the measurement results had a much larger variance from the average value. This is due to many factors, such as the limited bandwidth, more connected users, and the desire for the location to provide fair access for all users.

Facebook and Google+ were found to be the most efficient providers for downloading photos in almost all locations compared to the cloud providers. This is likely due to the optimization that the OSN providers have in place to deliver content in a timely manner. Facebook showed the best performance when uploading images compared to the other providers. OneDrive was found to be the most efficient in terms of download and upload throughput and Mediafire tended to have the worst performance for images. Mediafire has the additional query overhead and lack of an Android API to access their services, whereas the other providers created an optimized Android API for their services. The other cloud providers showed similar performance to each other. Dropbox and Mediafire showed the worst performance in many of the video tests. Google+ was found to have the slowest download times for the videos due to the limitations of their API.

Overall for the images, the bookstore and the mall had very similar performances with different providers while the other locations showed some variation in performances. Similarly, they were the slowest among the nine measurement locations. This can be attributed to these locations' own network or their ISP bandwidth capacity. The video downloads showed some performance variation between the providers at different locations, but showed consistent upload performance.

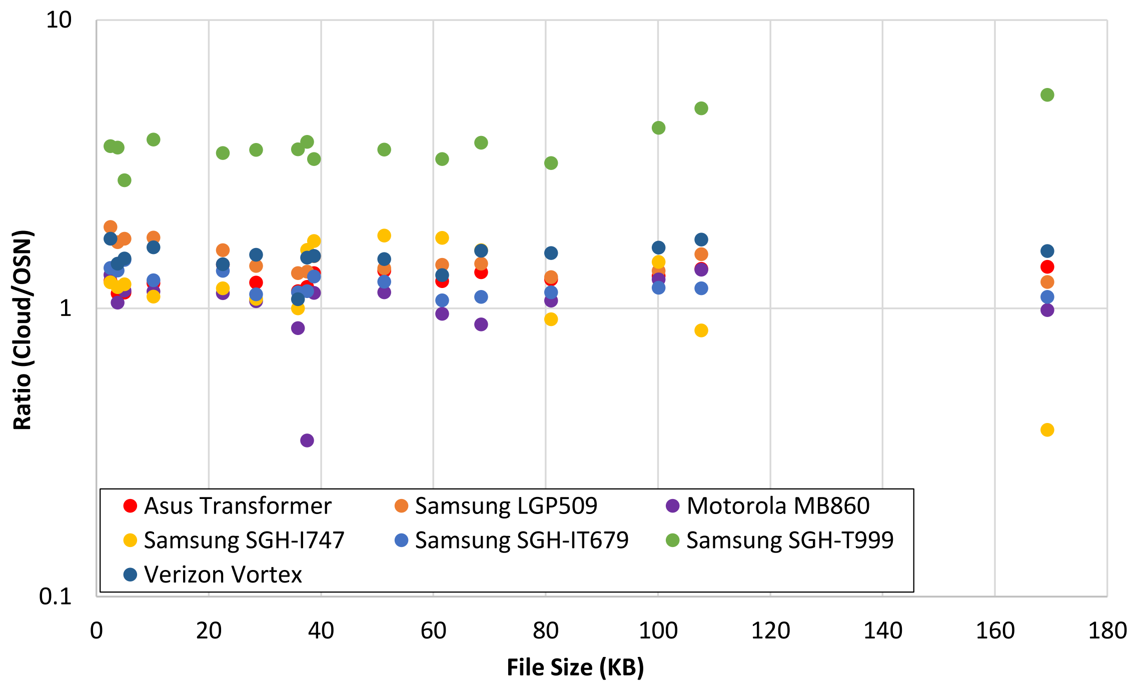


Figure 3.20: 3G/4G download performance for different devices on different cellular providers

3.5 3G/4G Measurements

Additional measurements were carried out to measure the cloud/OSN performance on 3G/4G cellular networks on the West Coast area of the United States[1]. As volunteers were asked to install and run the application that reported measurement results to a server at UNR, seven different devices were used. Figure 3.20 presents the ratio of Dropbox to Facebook performance for downloading different files for the same device. There was a considerable performance variation in data transfers over cellular networks, and this is likely due to the instability in the cellular bandwidth rate. Therefore, the ratios are presented rather than the timing of each device. Each volunteer did not complete the same number of experiments, due to the large amount of data consumption required to perform the measurements. Each point in the figure indicates one pair of measurements for the device at the same location performed in

[1]We would like to thank James Bridgum for developing the application for cellular measurements.

random sequence. If the point has a value above one, this indicates that the Facebook is faster than the Dropbox. Overall, it was observed that Facebook can be faster up to 5.5x or that Dropbox can be up to 9x faster. The measurements showed that Facebook is faster the majority of the time.

3.6 Related Work

Several studies have analyzed cloud storage providers in detail. Drago *et al.* [12] focused on characterizing how Dropbox is utilized by different users and how much data is transferred from a university campus and a residential ISP. They found that the location of the Dropbox server has a major factor in the performance. Zenuni *et al.* [37] compared some of the features different cloud storage providers offer and the performance from a single PC. Drago *et al.* [11] also focused on discovering the different system architectures and capabilities of Amazon Cloud Drive, Dropbox, Google Drive, SkyDrive, and Wuala. The study revealed that protocol design and the client capabilities are very relevant to network performance and reducing bandwidth overhead. Finally, Gracia-Tinedo *et al.* [22] analyzed the transfer speed, variability, and failure rate of Box, Dropbox, and SugarSync. It was found that file size, geographical location, and hour of the day impact performance.

There has been similar studies that analyzed social network user experience. Casas *et al.* [8] studied the quality of experience in Facebook and YouTube usage on 33 different 3G broadband devices. The authors found that YouTube's user experience varies greatly when the network conditions change because of the high volume of content that must be downloaded. Facebook was found to have a more consistent user experience, and it was attributed to the lower amount of traffic. The authors focused on gathering the users' ratings of the performance and collected daily traffic usage for YouTube and Facebook. Other studies focused on different aspects of social networks. Mislove *et al.* [30] analyzed the structure of social networks where the authors analyze Flickr, YouTube, LiveJournal, and Orkut to understand the relationships in the network.

Chapter 4

Personal Online Social Network (POSN) Architecture

In a social networking environment, there are two common types of roles, *data owners* and *friends*, where a user can have one or both types of roles. Data owners are users who have uploaded some type of content (such as photos, links, videos, statuses) into the social network. Friends on the other hand, have some sort of relationship to the user and can access the content of different data owners. A data owner can have any number of friends and be apart of groups that they are interested in. Interactions between the user and their friend circle is a vital part of the social network experience, and must be fully incorporated into a decentralized OSN platform. These interactions need to be carried out as efficiently as possible to reduce the overhead for the users, while still maintaining the user's privacy. The following section discusses the fundamental design of POSN [15], and how the files in the system are organized to provide access control and privacy.

4.1 System Overview

One of the main challenges for decentralized peer-to-peer social networks is the availability of the content. Peer-to-peer systems typically require a peer to be online to exchange data, which can impact the availability of data when the user is offline. Instead, in POSN, we utilize free storage clouds (such as Dropbox, Google Drive, and Microsoft OneDrive) and/or the user's personal computer (PC) to distribute user

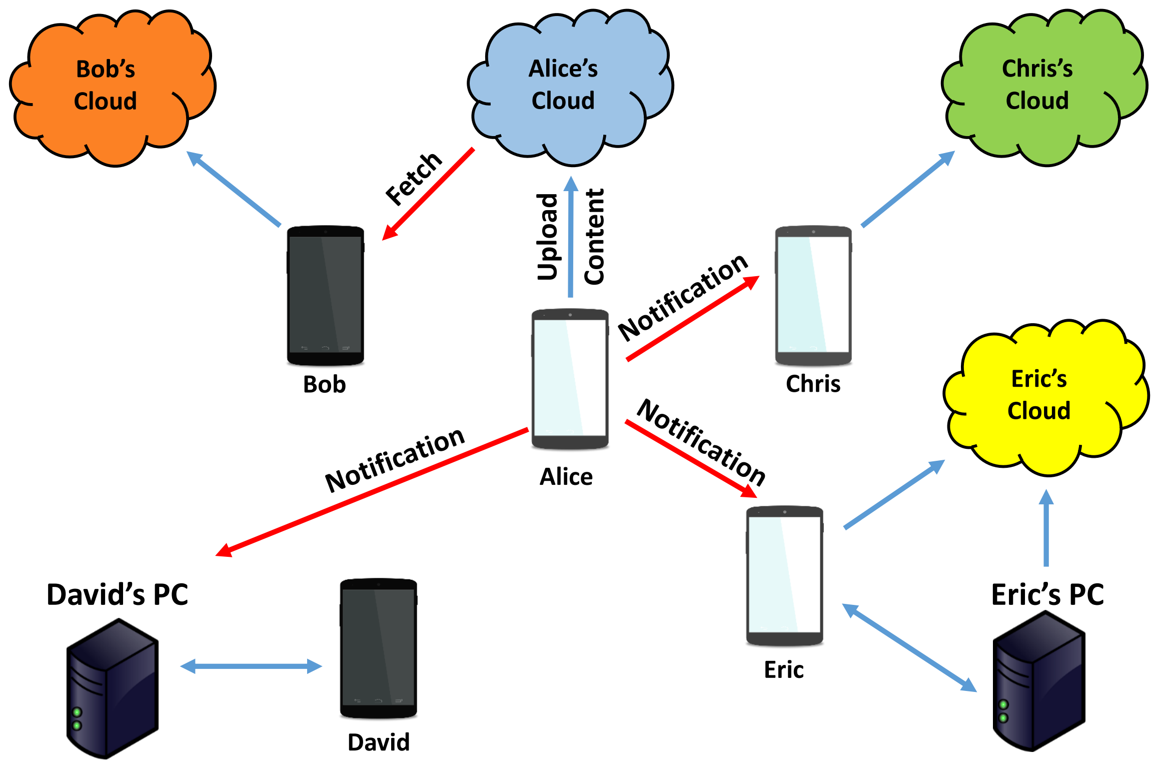


Figure 4.1: Example of how the data owner (such as Alice) uploads data into her cloud and how friend content is fetched from the friend's cloud

content. The cloud is used only to store encrypted content so that cloud provider or adversary has no understanding of the content. It is assumed that the cloud storage cannot perform any computations and hence does not know the plaintext of the stored content. Mobile devices are utilized to provide the required computation power to encrypt and disseminate the data through cloud. A user can host a PC server to perform computing functionality to remedy load on mobile devices.

Many cloud providers provide APIs to seamlessly connect mobile applications directly to their cloud services. Pairing the mobile the device with the cloud and PCs allows for efficient distribution of user content, as well as provides all of the OSN functionality without any additional infrastructure. Each user is required to have either an existing cloud storage account, or to create one when setting up the application. The user is also required to grant access to his or her friends, giving the user full control over what content is shared and who can see the content. POSN requires the cloud provider to allow direct download links to be created in order to provide fine-grained access to user content.

Figure 4.1 shows the decentralized structure in the POSN system. In this example, Alice is considered to be the data owner and she is connected to her friends: Bob, Chris, David, and Eric. When Alice wants to post new content, she will encrypt the data and upload it to her cloud. Alice's online friends: Chris, Eric, and David's PC will receive a notification and can fetch any multimedia from Alice's cloud. Bob was offline when Alice sent out the notifications, so he will have to check Alice's cloud for new content when he comes back online. If Eric posts new content, he can upload to his PC and/or to the cloud. When Alice receives a notification from Eric, Alice can fetch the data from Eric's PC or cloud. A user can also choose to use a PC instead of a cloud, for example David. The PC benefits the system by processing the data to remedy workload on the user's mobile device. One drawback is that the PC should always be online if it is not backed up by cloud.

4.2 File Organization

POSN utilizes different files to hold the data, and is separated into different files using different encryption keys to allow for fine-grain access control. As data is decentralized fine grained access control can not be provided through centralized authorities as in [31] but rather is provided through dissemination of encryption keys. Figure 4.2 shows how the files are organized in the cloud and how they are linked. The following subsections discuss the purpose of each type of file in the system and what information is stored.

4.2.1 Owner Files

The data owner file (such as Alice User File in the figure) is created for every user and holds information that is specific for that user. The file is encrypted with user's own private key and holds the data for the owner's public key, IP address, port number, and online status information. This information allows for friends to send content

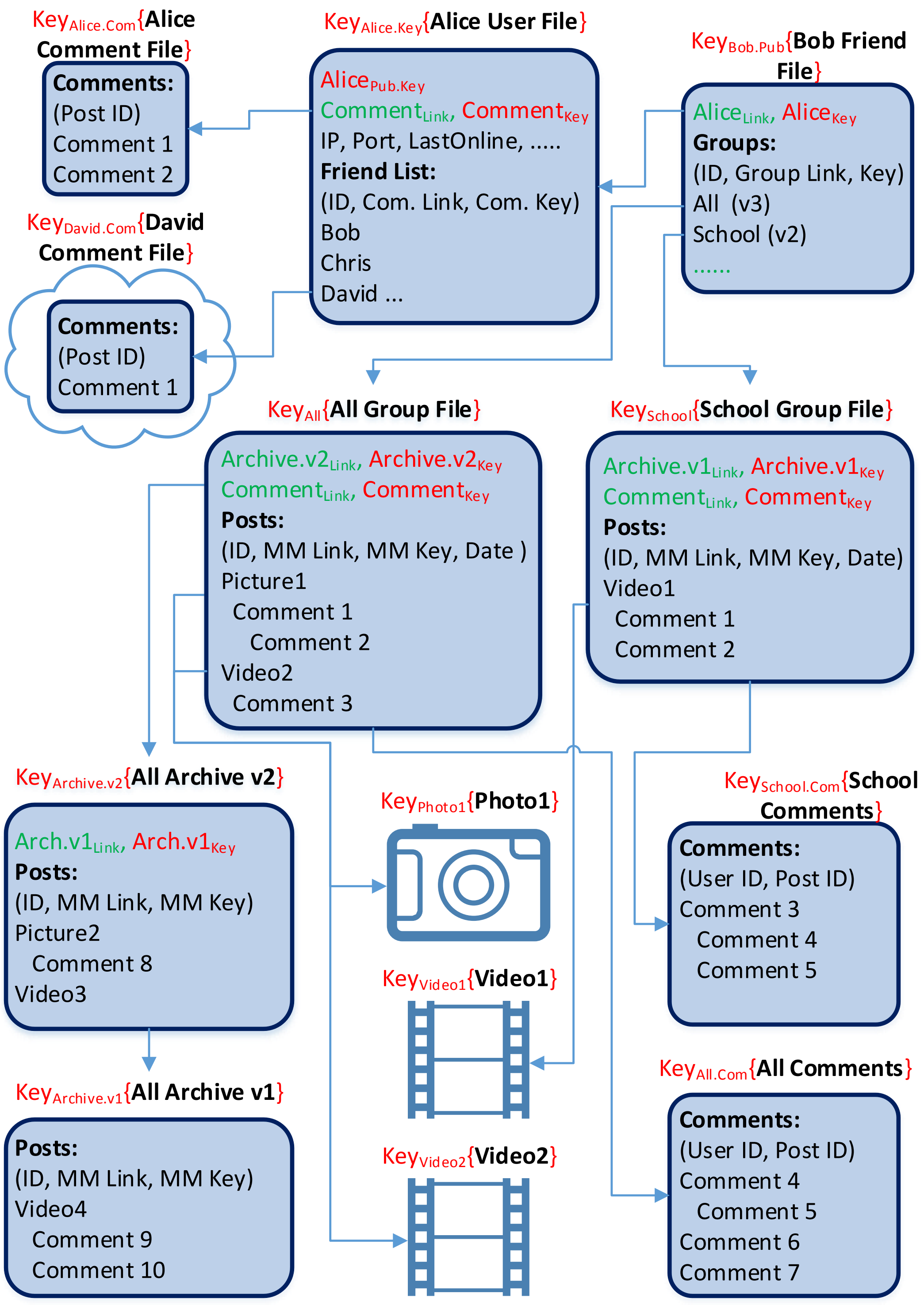


Figure 4.2: Diagram for the POSN file system

to the data owner directly if they are online. The user's friend list is also stored to allow for friends to know mutual friends in order to improve data dissemination. The user's friends can go to the common friends when they become online to get any new posts when the friend was offline. Each friend in the list also has a link and a key for a temporary comment file that is used to store the friend's comments when the data owner is offline and friend uses another cloud.

4.2.2 Friend Files

Friend files are created for each friend and are used to hold the information for the groups the friends can access. Each friend file is encrypted with the friend's public key to guarantee only the friend can access their file. The file contains the list of groups the friend is in, as well as the group ID, URL to the group wall, and the associated wall key. Each group entry will have a version number in order to keep track if the friend needs to fetch any new wall files. A link and key to the data owner file is also included, so that the friend can have access to the friend list and the data owner's online information.

4.2.3 Group Wall Files

The group wall files hold all of the data for the individual posts that the data owner uploads onto the social network. Each post has its own set of associated meta-data as follows: post ID, posting date, content type, user ID, content data, and comments. If the post is a textual status update or a link, then the content is embedded directly with the meta-data. Multimedia posts on the other hand, require additional fields to hold the link to the multimedia file in the cloud and the appropriate symmetric key. The multimedia file has to be uploaded separately into the cloud and cannot be embedded due to the files typically having a large size. Comments are integrated into the walls along with the posts. A comment contains the actual comment, the date/time it was created, and the ID of the user who posted it. The wall file also contains a link and key to the group's archive file, so friends can access older posts.

Each group wall file is encrypted with the group's symmetric key that is stored in the friend file.

4.2.4 Group Archive Files

As the data owner uploads new content to the wall, the group wall files will continuously grow in size, and therefore it becomes slower for the friends to process the new posts. Users typically access a small number of objects among a vast number of posts, with many users accessing only recently posted objects. The purpose of the group archive file is to reduce the overhead of friends by storing older posts into a separate file and minimizing the size of the group wall file. The archive holds the same format for the post content and meta-data as the group wall file does, and holds the link and key to the previous version of the archive file. The archive is encrypted with the appropriate version of the symmetric key, and the latest version is stored in the group wall file.

The archive files create a chain of files from the more recent posts down to the oldest posts. The purpose of the chaining the files is twofold: to organize the post history into different time periods, and to make the friend revoking process more efficient (see Section 5.2.4). The archiving process can be done periodically every day, week, month, or year depending on how often the data owner posts new content. The archive process will also be invoked when the data owner wishes to revoke a user from one or multiple groups. Since reorganizing the archive can require additional processing, the archiving can be done when the user's device is charging and connected to a Wi-Fi connection to minimize the impact on the overall performance of the device.

4.2.5 Temporary Comment Files

When a friend makes a comment on a data owner's post, the comment can be either directly sent to the data owner if they are online, or be placed into a temporary comment file that is stored in the friend's cloud if the data owner is offline. All friends that utilize the same cloud provider as the user can share a single file to append their comments (Bob and Chris in the figure). Other friends (David in the figure) will have their comment files in their own cloud, which will be linked by the user. The data owner is responsible for managing the comments by appending the friends' comments to the post. This scheme was chosen to allow for the support of multiple cloud providers to store the user's data. Because it is not likely that all of the users will have the same cloud provider, a user who does not have an account with the same provider as the data owner cannot directly add their comment to the comment file. When the data owner becomes online, he or she will have to go through their friend's comment files, fetch any comments that belong to their posts, and add them to the specific post comment file.

4.3 Related Work

There have been many studies that implement a decentralized OSN using different storage methods and encryption schemes in order to ensure user privacy. The primary concerns for decentralized OSNs are how the data is stored and how access control is achieved by using encryption or other methods. PeerSon [7] utilizes a Distributed Hash Table (DHT) to look up where the data is stored and uses both asymmetric and symmetric encryption for fine-grained access control. The privacy issues, however, are not well addressed in PeerSon. Safebook [9] also uses a DHT and asymmetric and symmetric encryption, as well as trusted parties that serve as mirrors to help distribute the data. This system ensures anonymity from any observers outside of the friend network by using a multi-hop system, but the users within a specific group do not have anonymity from each other. Vegas [28] utilizes trusted storage servers to increase data availability for the users, and all of the data is encrypted using symmetric keys. LotusNet [1] breaks down the OSN functionality into widgets, and uses a DHT to distribute data to the proper users. The content owners are responsible to grant permission to the users so they can access the data.

Several studies incorporated Attribute-Based Encryption (ABE) into decentralized OSN platforms in order to ensure data confidentiality and fine-grained access control. Cachet [33] improves the architecture of Decent [24], and has the users store data container objects. Each container is acts as part of the access control mechanism by only allowing users with the correct key to decrypt the data. The locations of the different containers are stored in a DHT and multiple nodes can store a container to improve the access time. One issue is that the access policy is defined openly in the container, and allows for the users to see it. Persona [2] uses storage that is considered not trusted and uses ciphertext-policy attribute-based encryption to ensure data confidentiality. Access control lists are defined by each user to determine which friends can have access to the data and the storage enforces them. The user's privacy is not guaranteed because the user's access permissions are stored in the Access Control List (ACL), which is not encrypted so a correspondence can be made.

Bodriagov, Kreitz and Buchegger [4] adapt Predicate Encryption (PE) for a decentralized OSN environment in order to encrypt the data and hide the access control policies. PE is computationally expensive, but the authors created a scheme to construct the access policies, as well as utilize bloom filters to significantly increase the performance. One issue is that the storage complexity is $O(2^g)$, where g is the number of groups a user is a part of. If a user is part of a large number of groups, then the performance of the algorithm is significantly impacted. The author's PE scheme also reveals partial information about the access policy that can be estimated from the size of the key.

Anonymous Broadcast Encryption has been designed into the decentralized OSN architecture by Bosk and Buchegger [5]. The main idea is to distribute keys to users so only those users can decrypt a file, and the users who have the key do not know who else has it. The authors proposed a publisher and subscriber model and were able to achieve almost full privacy, but it requires the entire OSN to be carefully implemented. It was concluded that the model is efficient, but the publisher could have more overhead. The findings are only theoretical estimates and have not been fully implemented.

There are several other proposed decentralized OSNs that utilize a hybrid of client-server and peer-to-peer architectures. Vis-a-Vis [34] assigns each user their own personal virtual server (VIS) to store data. Each VIS is treated as a member in an overlay network that represents different social groups. Confidant [26] stores user data without encryption on secure devices. It also relies on trust relationships with friends so they can replicate the data to have better availability. Polaris [36] takes a different approach and has OSNs store and manage the user data. The user can use different providers to have different OSN functionality and any sensitive data is stored on the user's mobile device. Each provider gets only a subset of the entire personal data, instead of a single provider having the entire data set.

Chapter 5

POSN Application Implementation

This chapter describes the implementation of the POSN application and details the functionality used to carry out common social networking tasks. The POSN application currently supports the most popular cloud providers: Dropbox, Google Drive, and OneDrive and is implemented using the APIs described in Chapter 2. All of the code is written in Java and used the Android Studio development environment to create the application.

5.1 Local Application Files

The POSN application uses a set of local files to manage the data that is shared through the application. As a single user can belong to a large number of friend groups, the constant downloading of content files from the friend's clouds can be slow and not efficient for a responsive application. As a result, there is a need to store the previously fetched data, so it can be displayed in the application immediately while the new data is fetched from the clouds.

Files are created for groups the user created, friends list, notifications, and the newsfeed. These files store the information as arrays of JSON objects to allow for fast saving and parsing of the data. When the data is read in from the files, each file's information is stored in a HashMap, where the key is the appropriate ID. This data structure was chosen because it has a constant time complexity for looking up data, which is ideal when checking for redundancies. Some of these files, such as the newsfeed, can grow to be large, therefore a mechanism will need to be in place to remove older information, once a size threshold has been met.

5.2 POSN Functionality

This section describes the implementation details of the social network functionality. Android's primary execution thread is used to handle the user interface and touch responses, therefore code that requires large amounts of execution time need to be carried out on a separate thread. POSN uses AsyncTasks to execute the functionality described in this section, due to the need to execute file I/O and network communications with the cloud. AsyncTasks provide flexibility to execute code off the main thread and then provide a method to update the user interface with the results from that execution.

5.2.1 Initial Setup

When the application is launched for the first time, the user will need to provide their email address and a password. The password is used to authenticate the user every time the the application is opened, and it is used to create a symmetric key to encrypt/decrypt files that are stored on the device. It is important to note that the password itself is never stored on the device or in the cloud, so if a user forgets their password, the content will have to be recreated. In order to recover the password, we could utilize challenge questions. The user can be given a fixed number trials before the challenge is erased to prevent brute force attacks. The user will also create a public/private keypair that is randomly generated from a seed that is obtained from a box that the user draws in for a period of time.

The user is also prompted to initialize a cloud provider in POSN, and is required to either create a new account with the cloud provider, or to sign in with an existing account. After the user is logged in, a screen showing the application's permissions is displayed, and the user is required to accept them. Once the permissions are accepted, all of the initial folder structures in the cloud and on the device are created.

The final initialization step is to create different friend groups and send out initial friend requests. The user can organize their friends into multiple groups, where each friend can be in one or more groups. Each friend group is given a unique symmetric key, and this key is used to encrypt the group's wall post file.

5.2.2 Unique ID Generation

In order for efficient content access, we need to create some unique IDs for friends, groups, posts, and comments. Our system uses the SHA-256 hashing algorithm along with different input strings to generate the IDs. The input strings for different IDs are as follows:

- *User ID* - email address and a random salt
- *Group ID* - name of the group and email address
- *Post ID* - user ID and generation time
- *Comment ID* - post ID and generation time

5.2.3 Friendship Establishment

As there is no central database, we can not utilize friend recommendation systems such as [32]. Establishing friendship in a decentralized system requires methods to find friends and ensure their identities. If the user is using a mobile device, then the phone's contacts can be used to add new friends within POSN. Currently, an email message is sent to the desired users, along with a Uniform Resource Identifier (URI), that contains relevant information to initiate the request. SMS messaging is another method that can be used to send friend requests.

The friendship establishment process has three phases and is shown in Figure 5.1. The requesting user (such as Alice) first sends an email to the new friend (such as Bob) that contains a URI with the following: a user ID, public key, temporal cloud URL, and nonce. The temporal cloud URL is used to ensure that a man-in-the-middle

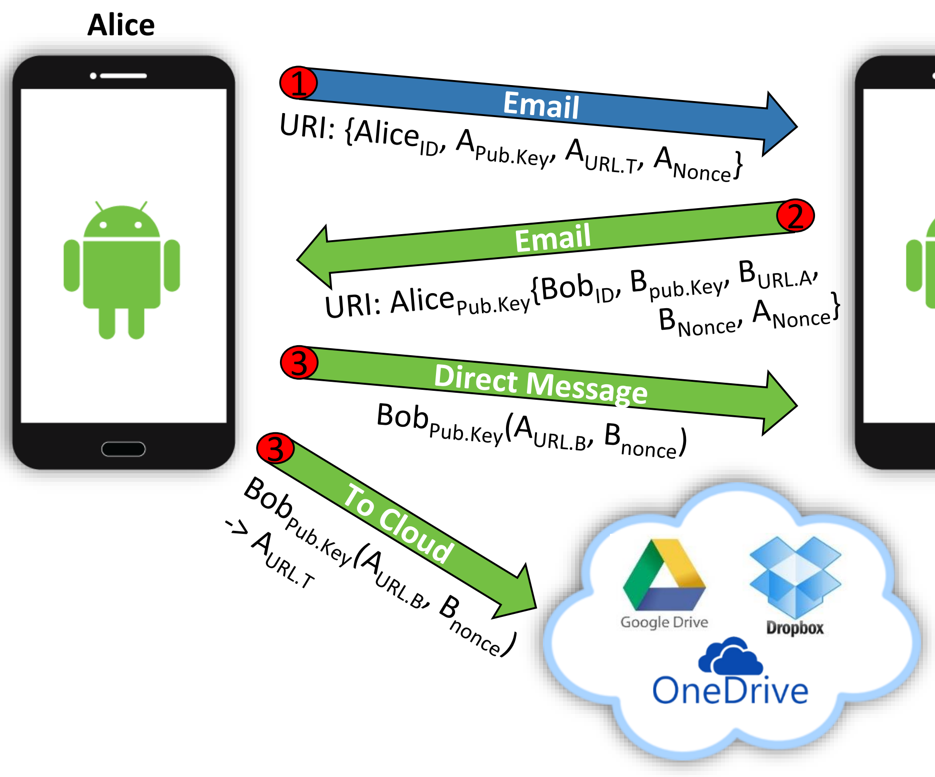

Alice
Bob
1
Email
URI: {Alice_ID, A_Pub.Key, A_URL.T, A_Nonce}
2
Email
URI: Alice_Pub.Key{Bob_ID, B_pub.Key, B_URL.A, B_Nonce, A_Nonce}
3
Direct Message
Bob_Pub.Key(A_URL.B, B_nonce)
3
To Cloud
Bob_Pub.Key(A_URL.B, B_nonce) -> A_URL.T
Google Drive
Dropbox
OneDrive

Figure 5.1: Friendship establishment process between two users

attack is avoided, where the adversary could intercept the initial email and responds back to Alice with their information instead of Bob.

When the friend clicks the URI in the email, the POSN application is opened (or download page is opened). In the application, Bob can respond in either two ways: accept or decline the friend request. If the friend request is accepted, then an email is sent back to Alice with the following friend information: user ID, public key, cloud URL, the requesting nonce, and a new nonce. It is important to note that the information sent in the Bob's URI is encrypted with Alice's public key to ensure that only Alice can read. Bob will select which group Alice will be part of and create a new friend file for Alice and add the appropriate group information.

Once Alice receives Bob's email of accepting her request, Alice creates a new friend file for Bob and adds the appropriate information to the file. The direct link to the friend file is encrypted with Bob's public key and sent as a direct message to Bob's device. The new encrypted link to the friend file is also uploaded to the temporal file in the cloud. Even if an adversary knew about the temporal file, the encryption on the data prevents an adversary from viewing the data inside the temporal file without knowing Bob's private key, which is never shared by anyone. Once Bob becomes online, he will notify Alice that the temporal file has been fetched and Alice's application can delete the file.

5.2.4 Access Control

With all social interactions, there may be a time where a user wishes to change what a friend can access, or even wish to no longer be connected with a friend and remove them from their network completely. Adding a friend to a new group is very straightforward, where the new group information and key is appended to the friend's friend file. Since all of the group archives and individual posts are accessible through the group file, no additional overhead is required.

On the other hand, revoking a friend's access can be a challenge, due to changing the keys and possibly re-encrypting the files to prevent the user from accessing the content. This re-encryption process would add a significant amount of overhead to the system, and in POSN, this additional overhead is avoided. Instead, POSN essentially cuts off a revoked user by encrypting any new group content with a new key after removing a friend. A new symmetric key for each group that the revoked friend was in will be generated, and only the friend files of the friends in the same group will be updated with the newest wall link and key. This adds the benefit of not having to re-encrypt the old data, and this scheme has significantly less files to process.

The content that is currently in the group's wall will be archived by moving the wall file to a new location in the cloud. The revoked friend will only be able to access any content that was created before the revoking process. This design was chosen based on the fact that the revoked friend has already seen the older content. The data does not need to be re-encrypted because the user is likely has a local copy of that group's content on their device, and this would add unnecessary overhead.

This also adds the benefit that the revoked user will not know that they have been removed, and he or she will just think their friend is no longer posting new content. A notification will be sent out to the online friends if their friend file was updated due to the revoking process.

If the data owner wishes to completely remove a friend from their network, the data owner can remove the friend from all the groups in the friend file from the cloud and carry out the same process to update the group keys. By deleting the friend file, the friend would no longer be able to get any new key updates, or access any content from the cloud. The revoked friend may have a local copy of the data, but the group wall links will no longer be valid as the walls will be moved to a new location in the cloud, therefore preventing the user from accessing even the old content.

5.2.5 Content Sharing

Uploading and sharing content, whether it is a status update, picture, video, or a link, is a fundamental aspect of online social networking. When a user creates a new post, the user must select which group to share the post with. If the new post contains any multimedia, then a new symmetric key is generated to encrypt the multimedia file. The multimedia file is then uploaded into the cloud and the direct download link for the file is fetched. The multimedia key and the direct download link are embedded with the new post information. The appropriate group wall file is then updated by appending the new post to the wall files. The updated wall file is then uploaded into the cloud, and any online friends in the group will be notified of the new post. When the friends get the notification, they can fetch the wall file and can add any new posts to their newsfeed.

5.2.6 Content Access

Accessing different friend data is another fundamental part of social networking, and needs to be done as efficiently as possible. When a user becomes online, different wall files from all the friends need to be fetched and decrypted using the appropriate symmetric keys. The post ID is used to determine which posts need to be added to the newsfeed, and if any multimedia content needs to be fetched from the clouds. If multimedia content is needed, then the direct download link and the multimedia symmetric key is used to fetch and view the content.

If a user cannot decrypt the group wall file, then there are two possibilities: either the user's friendship has been revoked, and they no longer have access, or the key was updated and the newer version needs to be fetched. For the first possibility, the user will attempt to update the key by fetching their friend file and find that there is no update. When the decryption is attempted again, it will fail a second time, and the group file will be ignored. In the second possibility, the new key will be fetched and the decryption will succeed, and the algorithm can continue on processing the other wall posts.

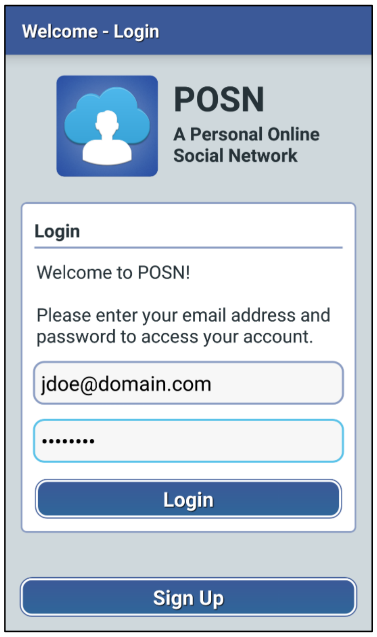
5.2.7 Friend File Update

In some situations, the user's friend file for each friend will need to be updated. The update function will be called to update the user's local copy of the friend file. The friend file is fetched from the friends' cloud, and the user will decrypt it using their private key. This allows for any updated group keys and any changes to the groups to be automatically fetched. This function should be called when a notification is received from a friend that alerts the user that an update to the friend file has been made.

5.3 Application Interface

This section displays a series of figures showing the interface of the POSN application. All parts of the interface were designed to be intuitive and user-friendly and were implemented using XML files. The components and layouts used are native to Android, but some have been modified to meet the functionality required by POSN.

Figure 5.2 presents the initial login screen where the user can supply their credentials to be authenticated or set up a new account. To create a new account, the



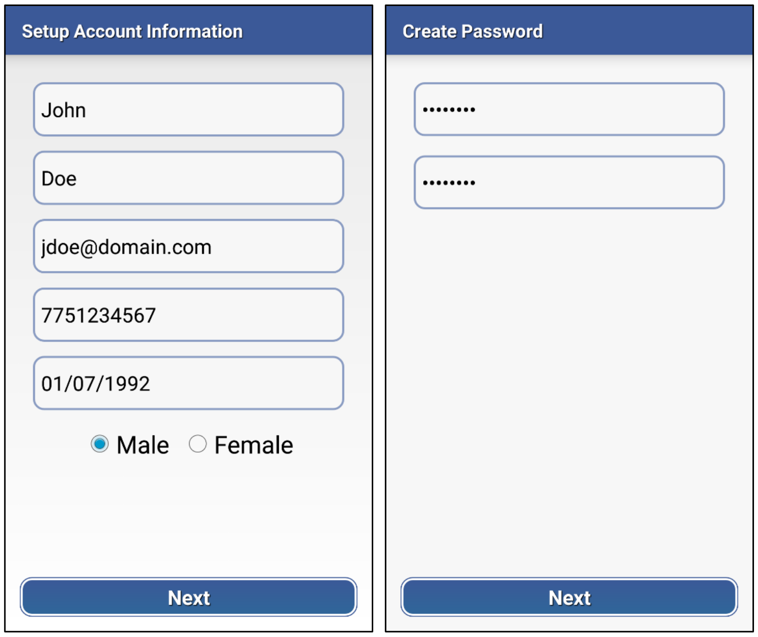
Welcome to POSN!

Please enter your email address and password to access your account.

jdoe@domain.com

••••••••

Figure 5.2: Initial login screen



John

Doe

jdoe@domain.com

7751234567

01/07/1992

[✓ Male] Female

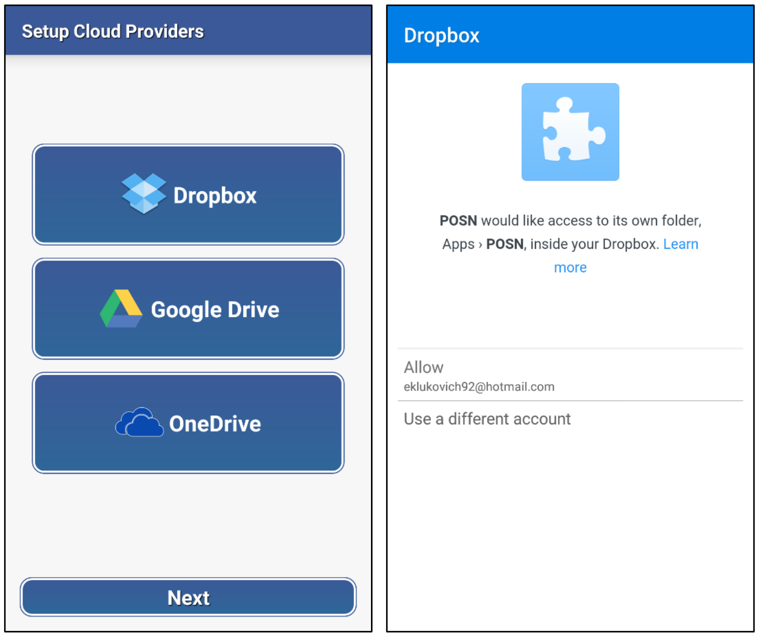
••••••••

••••••••

Figure 5.3: Account creation

user first has to supply some basic information about himself or herself and create a new password as shown in Figure 5.3.

The user can choose a cloud provider and he or she can sign in to their account. A screen will ask to grant permission to the application to use the account as in

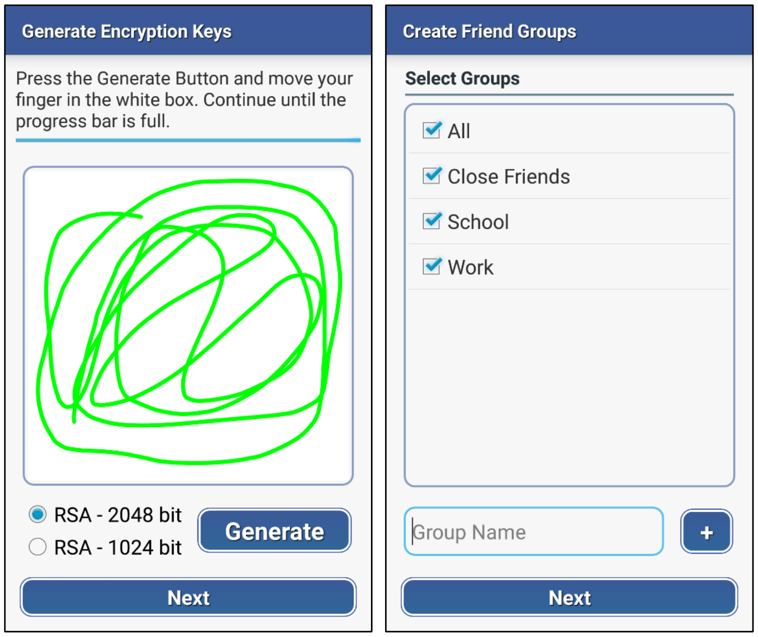


POSN would like access to its own folder, Apps › **POSN**, inside your Dropbox. Learn more

Allow
eklukovich92@hotmail.com

Use a different account

Figure 5.4: Cloud access



Press the Generate Button and move your finger in the white box. Continue until the progress bar is full.

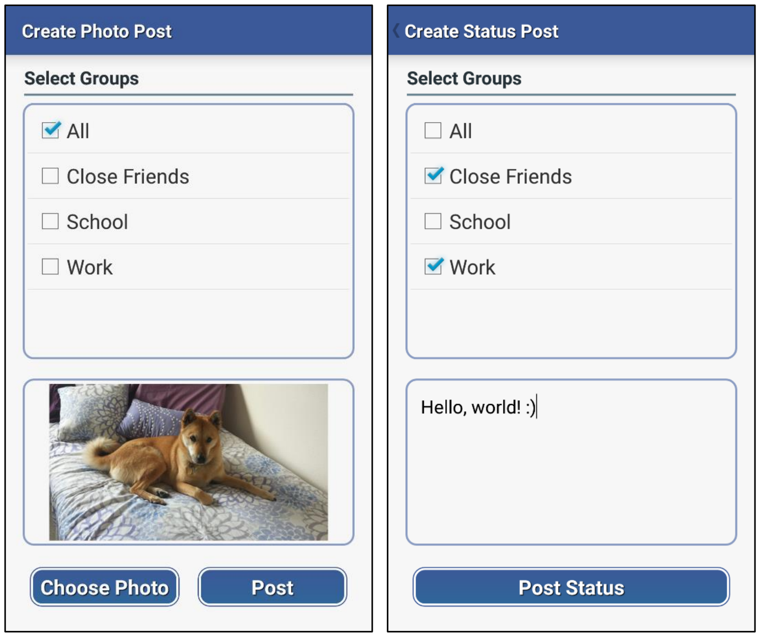
[● RSA - 2048 bit] ○ RSA - 1024 bit

- ✓ All
- ✓ Close Friends
- ✓ School
- ✓ Work

Figure 5.5: Key generation and group creation

Figure 5.4. Encryption keys are generated based on the seed from the drawing in the box as in Figure 5.5. Friend groups are then created to organize friends and control who views which content.

The user has the option to create a new status or wall post as shown in Figure 5.6.

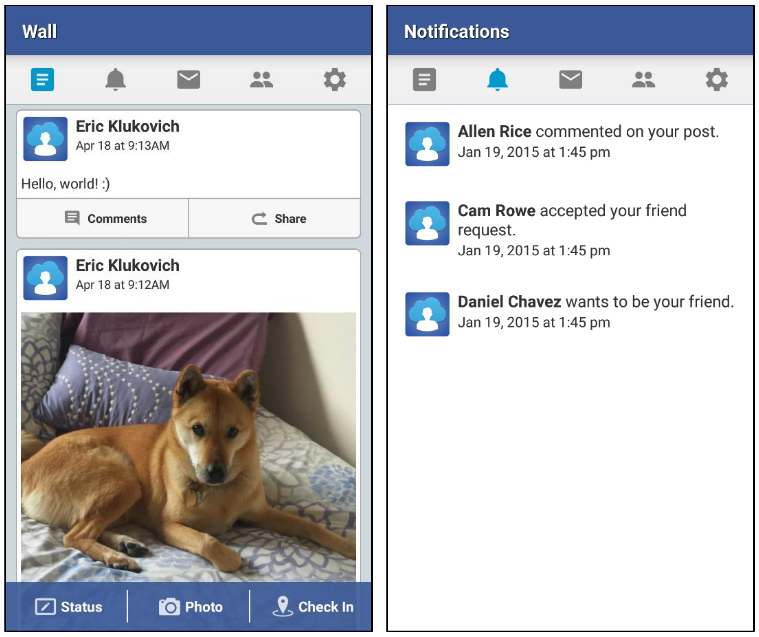


[✓ All]
Close Friends
School
Work

All
[✓ Close Friends]
School
[✓ Work]

Hello, world! :)

Figure 5.6: Posting new content



Eric Klukovich
Apr 18 at 9:13AM
Hello, world! :)

Eric Klukovich
Apr 18 at 9:12AM

Allen Rice commented on your post.
Jan 19, 2015 at 1:45 pm

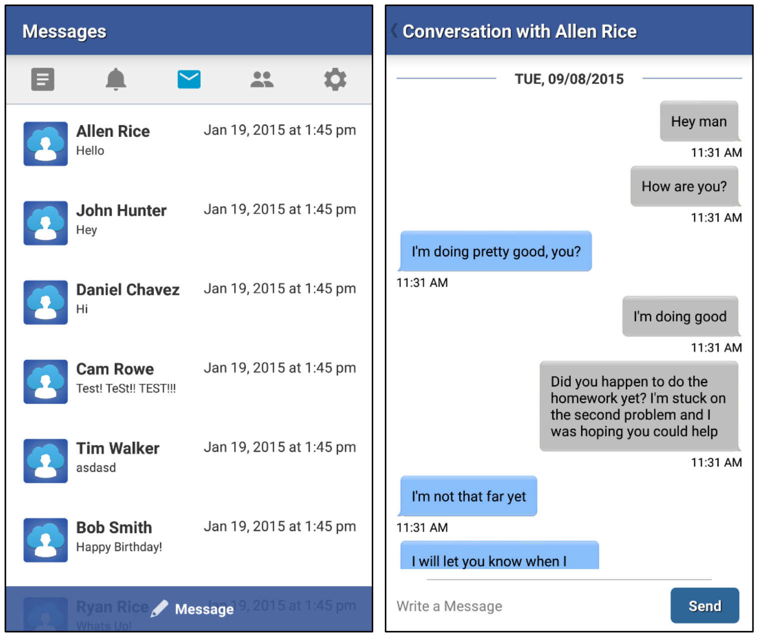
Cam Rowe accepted your friend request.
Jan 19, 2015 at 1:45 pm

Daniel Chavez wants to be your friend.
Jan 19, 2015 at 1:45 pm

Figure 5.7: Wall and notification tabs

The user can select which group to share the new post with, and add text or a photo. The first tab displays the user's wall with all of friends' posts as shown in Figure 5.7. Any new notifications are displayed in the second tab.

The third tab displays the conversations the user has with his or her friends


Messages

Allen Rice Jan 19, 2015 at 1:45 pm
Hello

John Hunter Jan 19, 2015 at 1:45 pm
Hey

Daniel Chavez Jan 19, 2015 at 1:45 pm
Hi

Cam Rowe Jan 19, 2015 at 1:45 pm
Test! TeSt!! TEST!!!

Tim Walker Jan 19, 2015 at 1:45 pm
asdasd

Bob Smith Jan 19, 2015 at 1:45 pm
Happy Birthday!

Ryan Rice 19, 2015 at 1:45 pm
Whats Up!

Conversation with Allen Rice

TUE, 09/08/2015

Hey man
11:31 AM

How are you?
11:31 AM

I'm doing pretty good, you?
11:31 AM

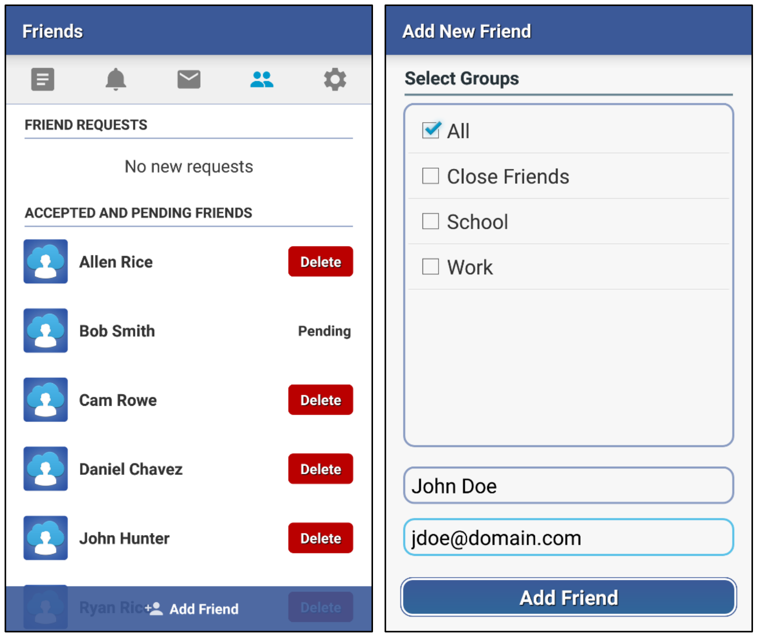
I'm doing good
11:31 AM

Did you happen to do the homework yet? I'm stuck on the second problem and I was hoping you could help
11:31 AM

I'm not that far yet
11:31 AM

I will let you know when I

Figure 5.8: Messaging tab


Friends

FRIEND REQUESTS

No new requests

ACCEPTED AND PENDING FRIENDS

Allen Rice	Delete
Bob Smith	Pending
Cam Rowe	Delete
Daniel Chavez	Delete
John Hunter	Delete
Ryan Rice	Delete

Add New Friend

Select Groups

[✓] All
[ ] Close Friends
[ ] School
[ ] Work

John Doe
jdoe@domain.com

Figure 5.9: Friend list tab

as shown in Figure 5.8. When the conversation is selected, the chat conversation is displayed. The fourth tab displays the user's current friends and new friend requests as shown in Figure 5.9. A friend can be added by entering a name and email, and selecting which groups they will be in.

Chapter 6

Conclusions

Social networking has gained a great importance to interact with other people in our daily lives. Popular social networks use a centralized architecture, which can lead to privacy issues as the data is controlled by the provider. A decentralized architecture can be utilized to distribute the data back to the users, where they will have more control of what content is being shared and with whom. In this thesis, we present POSN as a new OSN platform that is centered around mobile devices to process data and support the OSN functionalities. As data storage and availability is a challenge in a decentralized system, free cloud storage is used to supplement the user devices, while guaranteeing that the data is always available. We detail the design and implementation of the POSN architecture and application that supports many of the common social networking functionalities.

Overall, POSN is a decentralized social network that has distinct advantages over the centralized counterparts by ensuring that user content and privacy are protected. The way POSN uses encrypted data and how the files are structured provide data confidentiality and access control. The friend file provides the friend with their specific access rights, and the file can only be modified by the data owner and viewed by the data owner and the specific friend. The encryption keys are transferred between users directly where the keys are encrypted with a user's public key, or are placed in different friend, group, or user files where the friend is given specific access link and key. As a result, the access control scheme and encryption keys are hidden from other friends and adversaries who do not have access, guaranteeing the user's total privacy. The platform also provides the data owner full control of which users gets to view their content, and ensures that only those users can view it.

We also performed a feasibility study to determine if cloud providers could be used to deliver content to the users in a timely manner. The download and upload performance of two popular social networks and five cloud providers was measured on a separate Android application. Overall, the social network providers had a slightly better performance when delivering small files, but the cloud providers had better performance for the larger files. It was found that the design of the API to interact with the services have a major role on the efficiency. Locations with public Wi-Fi access showed the worst performance due to the limited bandwidth, but has consistent performance across all providers. The performance to deliver content with social networks and cloud providers was found to be very comparable, even though their primary targets of content delivery differ.

Chapter 7

Future Work

In this chapter, we discuss dome of the possible future directions for the work.

7.1 Cloud Measurement Study

In order to provide a more complete and unbiased measurement, the geographical location of the measurements could be considered. A user campaign can be designed to have volunteers perform the measurements at a variety of locations. A challenge that would arise is how to ensure that all the tests are performed without other biases getting introduced, such as the application is running in the foreground, other applications being closed, and the tests being performed during the non-peak hours. It was observed from the few volunteers from the 3G measurements, that these factors were difficult to control, and some results had to be disregarded due to various factors. The measurement application can be further developed to reduce these factors as much as possible.

Other tests and measurements can be implemented to determine the network impacts on the current measurements. Measurements such as cellular 3G/4G vs. Wi-Fi, congestion of the location, and device performance should also be carried out in future studies.

7.2 POSN Security Enhancements

Currently in POSN, the method of establishing friendship between two users can be susceptible to malicious email provider or client as a man-in-the-middle. If the user who initiates the friend request, and is unknowingly using a malicious email server, then the initial friend request email could be sent to the adversary. In turn, the adversary can then spoof an email and provide their information to obtain the initiating user's information to access content that was never intended to be shared with them. A solution to this issue would be to use two factor verification mechanism, such as an additional email provider or an SMS message to confirm the user. While both of these possible solutions are feasible, other options for verification or to remove this issue from the friendship establishment mechanism could be explored.

Another security concern is that POSN is susceptible to profiling attacks, where an adversary can monitor that traffic going to and from the cloud servers. While the adversary will not be able to access the file's contents due to encryption, they can approximate the user's identity based on which files the user is requesting and receiving. This issue can be resolved by using an anonymizer mechanism such as Tor or Invisible Internet Project (I2P) [17, 25]. This mechanism helps users maintain their privacy by allowing for resources to be requested without the adversary being able to link the usage to a specific user. The mechanism also prevents the adversary from determining who sent the initial request for data by observing other users in the network. The directories and file names stored in the cloud could also be randomized to further protect the users' privacy.

7.3 POSN Application

In the current implementation, when users go online, they are required to check all friends' clouds to get updated content. This method is satisfactory if the user has a small number of friends, but will be very slow as their friend list grows. Often times, a user's friends will have mutual friends with the user, and they can assist in updating the user. When the user becomes online, then he or she can go to a online friends who have the most common friends to get any updated content. This updating method will save the user a large number of connections to different clouds to fetch data. Other metrics such as friends with longest online duration or randomly chosen friends can be used as well. The implementation of optimizing the update process and a study to find the most ideal selection method should be carried out.

Commenting in POSN currently has the data owner manage and centralize all the comments for their own data posts. This method is straight-forward and easy to implement, but poses a challenge if the data owner is offline very often. If two of the data owner's friends are commenting on the same post, it is not guaranteed that the friends will see the other comments. Instead, a new scheme could be designed to allow for the data owner to treat a set of their friends as delegates to help propagate comments to mutual friends. The friends could have either the largest amount of mutual friends, or use the same cloud provider so they can directly add comments in the comment file.

Another important features in OSNs is having the ability to easily search for content. In POSN, a search mechanism is not currently implemented. Each user can create an index file for each group and when the user posts new content, tags can be added to describe the content. The the index files can be distributed to each of the user's friends to assist with content searching.

Appendix A

Complete Cloud Measurment Results

A.1 Different File Size Results Per Location

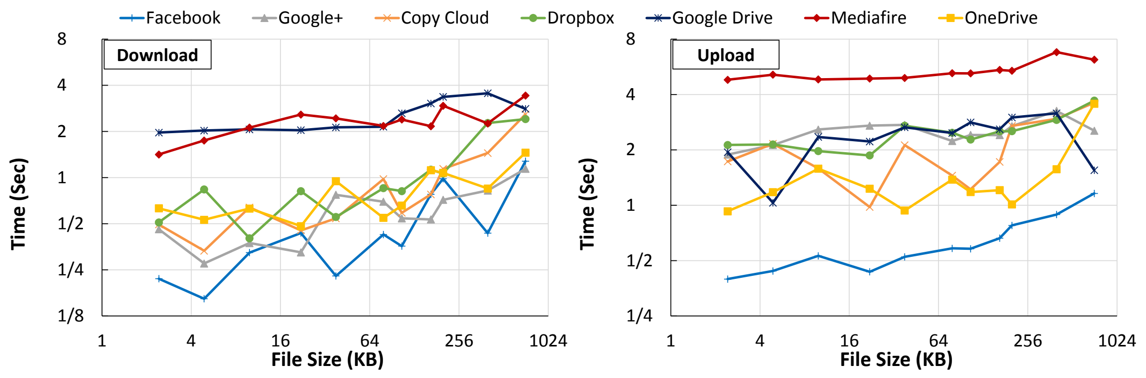


Figure A.1: Download and upload time for different file sizes at the airport (log-log scale)

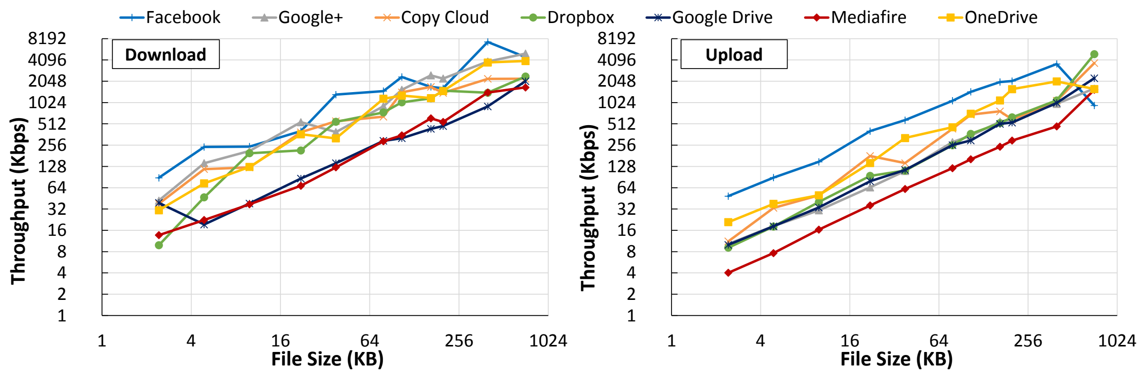


Figure A.2: Download and upload throughput for different file sizes at the airport (log-log scale)

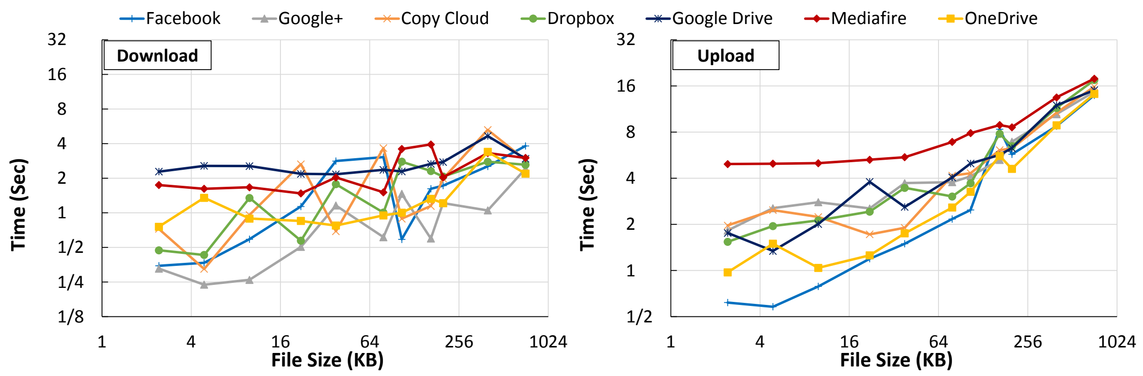


Figure A.3: Download and upload time for different file sizes at the bookstore (log-log scale)

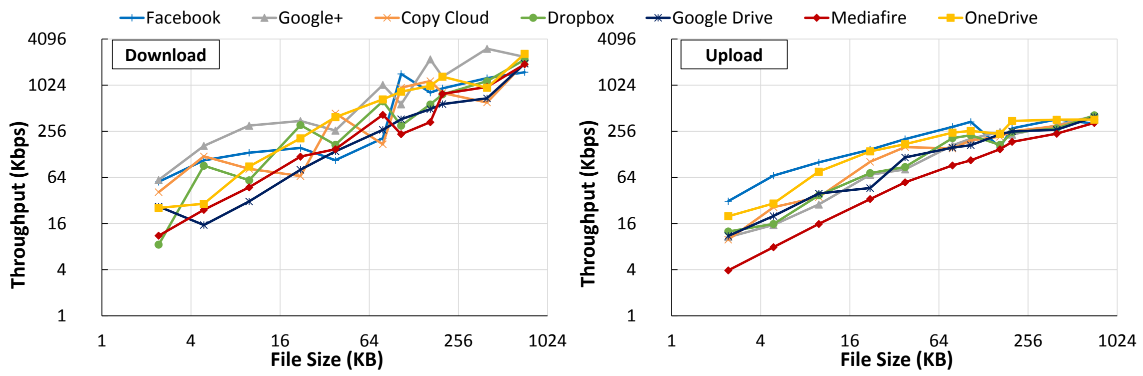


Figure A.4: Download and upload throughput for different file sizes at the bookstore (log-log scale)

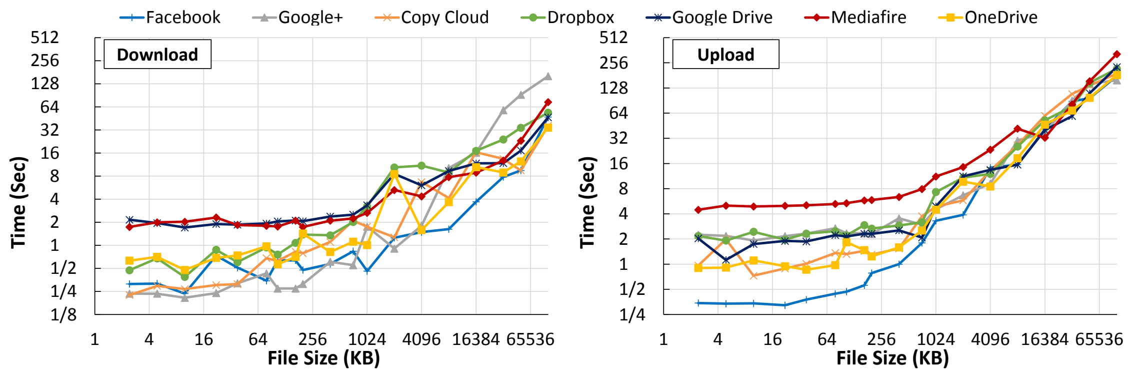


Figure A.5: Download and upload time for different file sizes at the cafe (log-log scale)

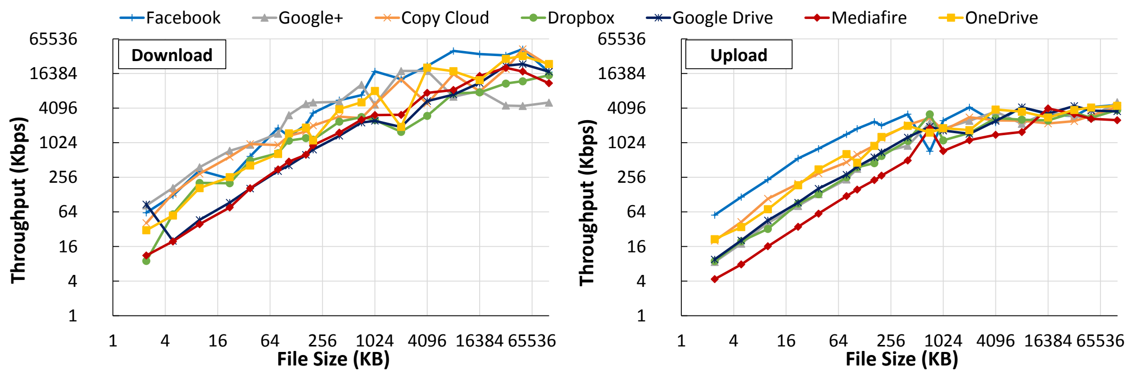


Figure A.6: Download and upload throughput for different file sizes at the cafe (log-log scale)

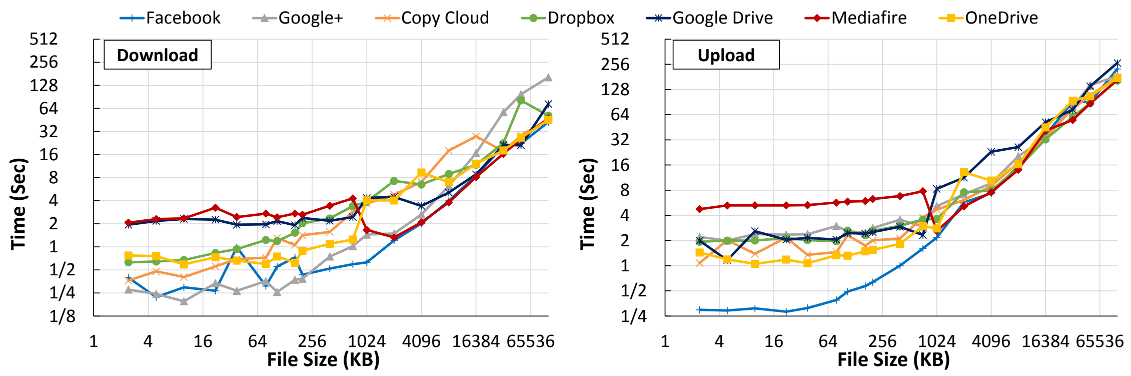


Figure A.7: Download and upload time for different file sizes at the library (log-log scale)

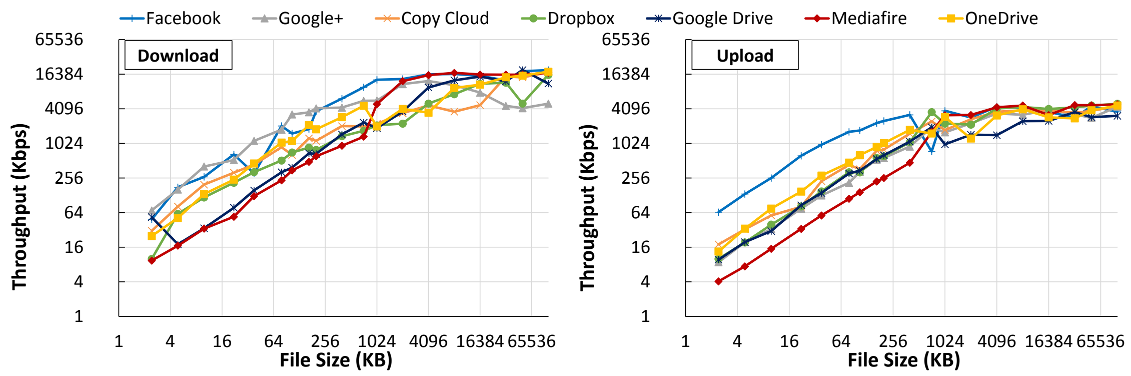


Figure A.8: Download and upload throughput for different file sizes at the library (log-log scale)

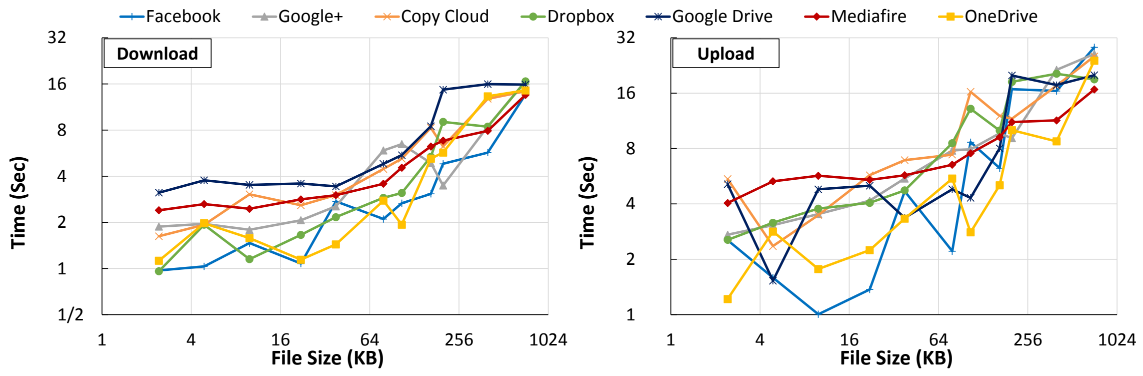


Figure A.9: Download and upload time for different file sizes at the mall (log-log scale)

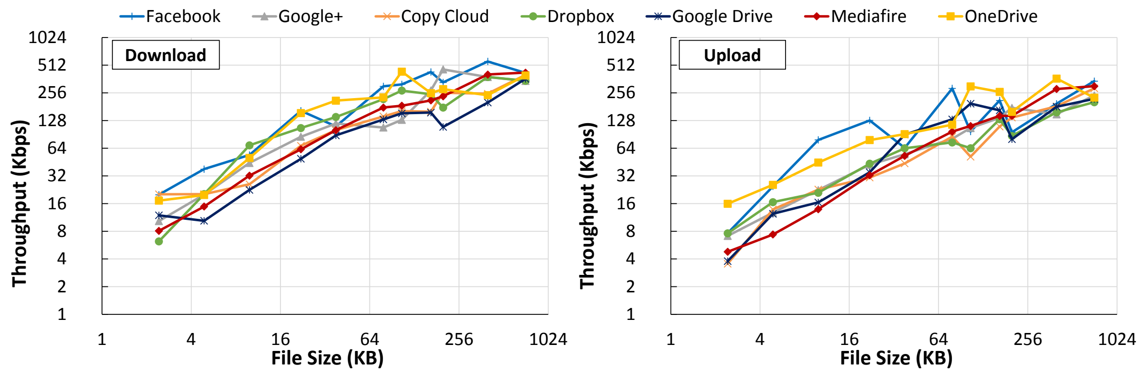


Figure A.10: Download and upload throughput for different file sizes at the mall (log-log scale)

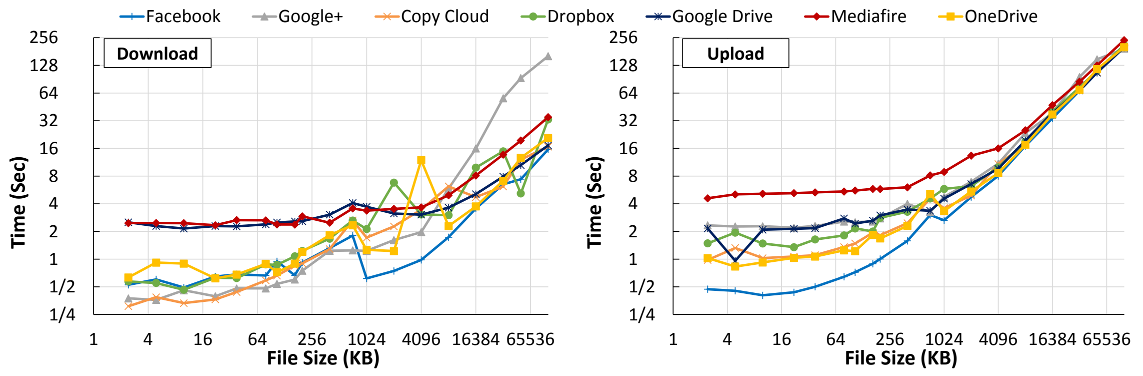


Figure A.11: Download and upload time for different file sizes for ISP 1 (log-log scale)

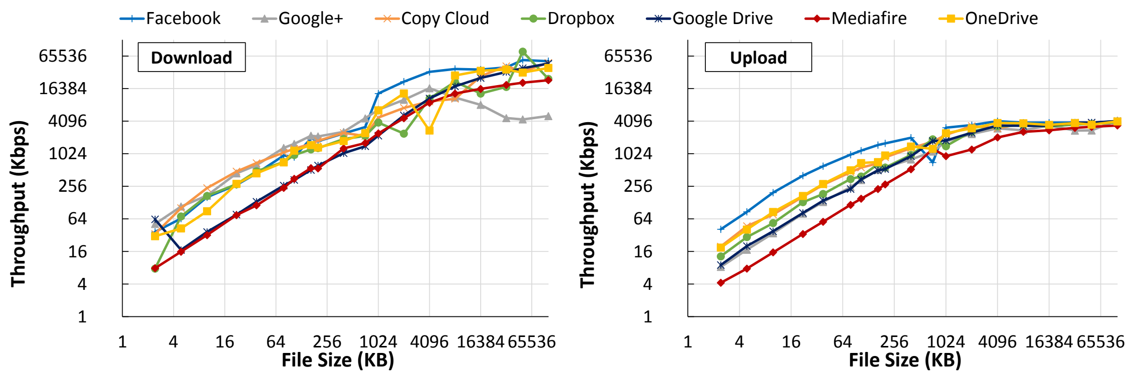


Figure A.12: Download and upload throughput for different file sizes for ISP 1 (log-log scale)

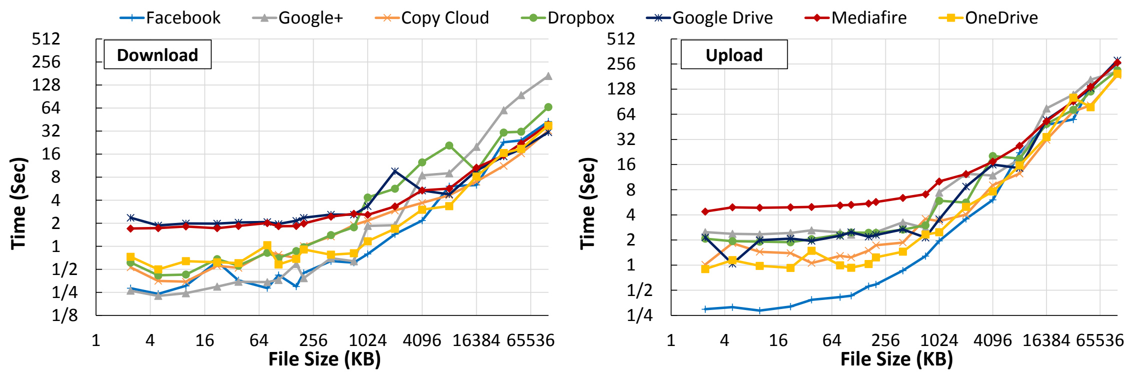


Figure A.13: Download and upload time for different file sizes for ISP 2 (log-log scale)

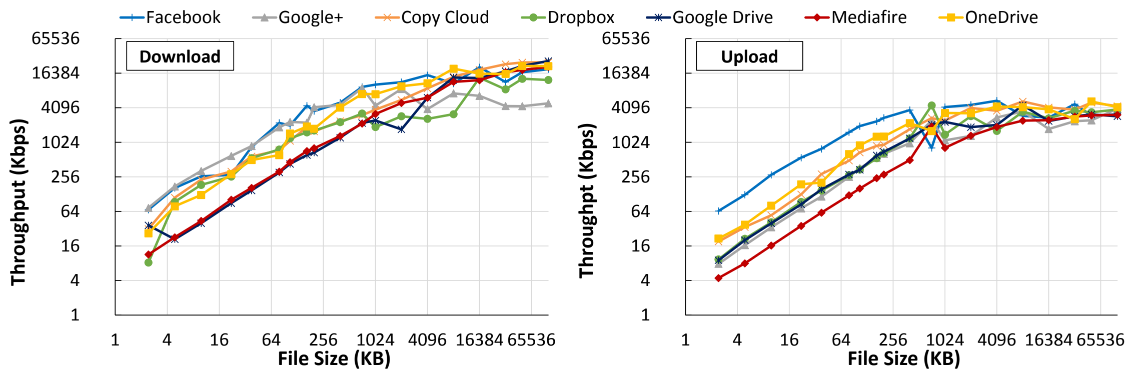


Figure A.14: Download and upload throughput for different file sizes for ISP 2 (log-log scale)

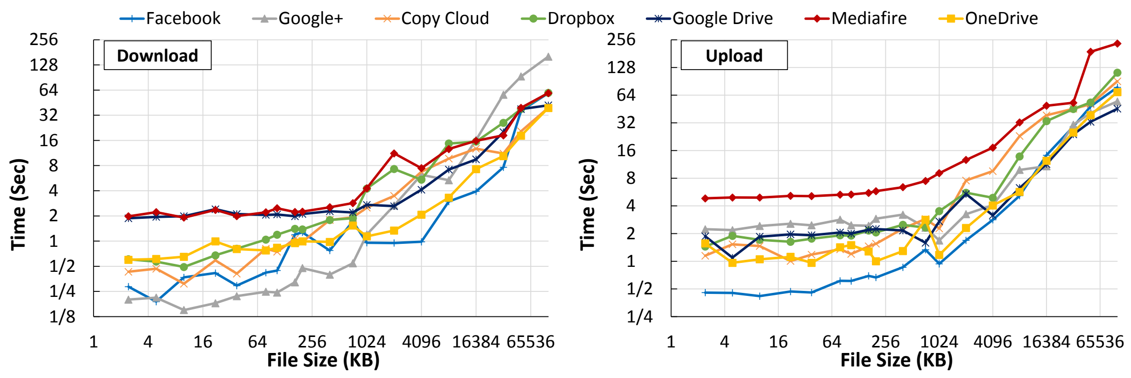


Figure A.15: Download and upload time for different file sizes for ISP 3 (log-log scale)

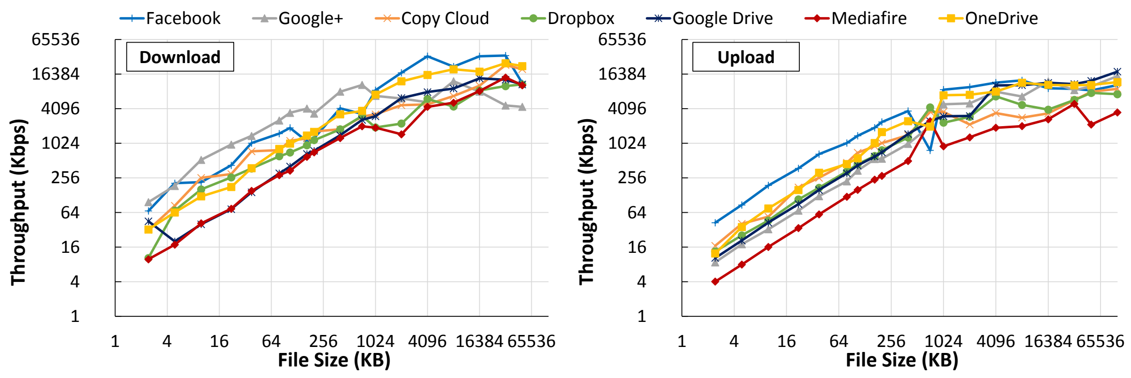


Figure A.16: Download and upload throughput for different file sizes for ISP 3 (log-log scale)

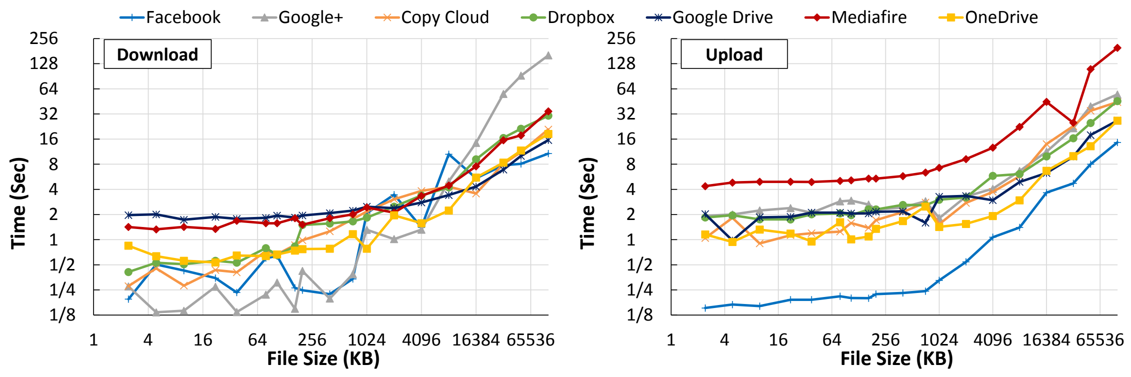


Figure A.17: Download and upload time for different file sizes at the university (log-log scale)

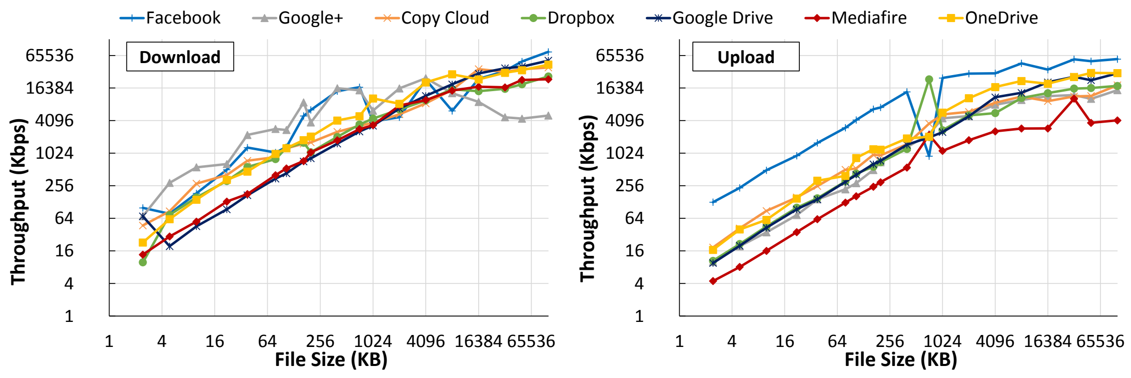


Figure A.18: Download and upload throughput for different file sizes at the university (log-log scale)

A.2 Different Location Results Per File Size

A.2.1 Photo Files

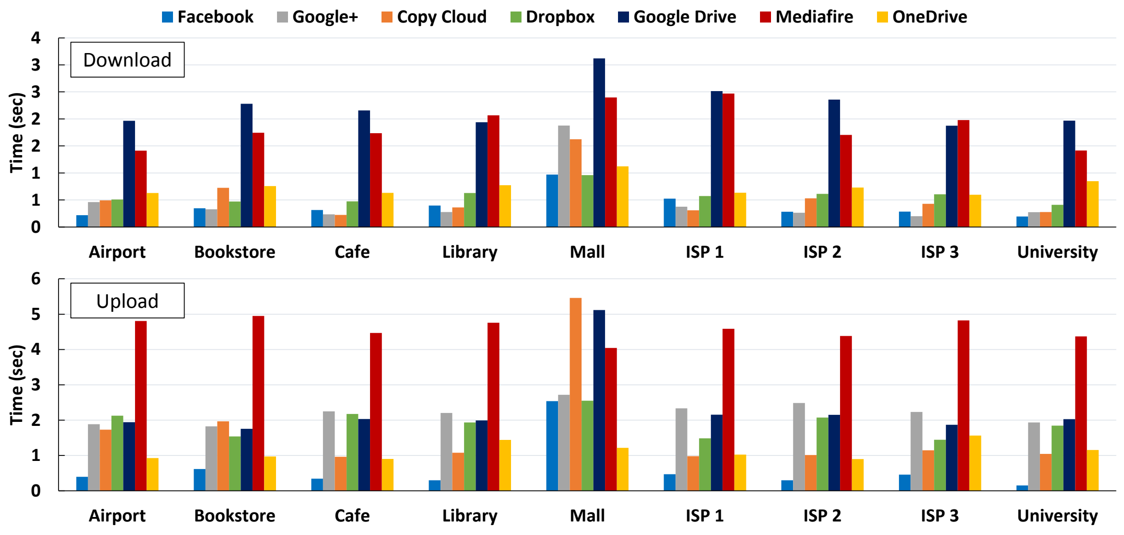


Figure A.19: Download and upload time for various locations with a 2.42 KB photo (log scale)

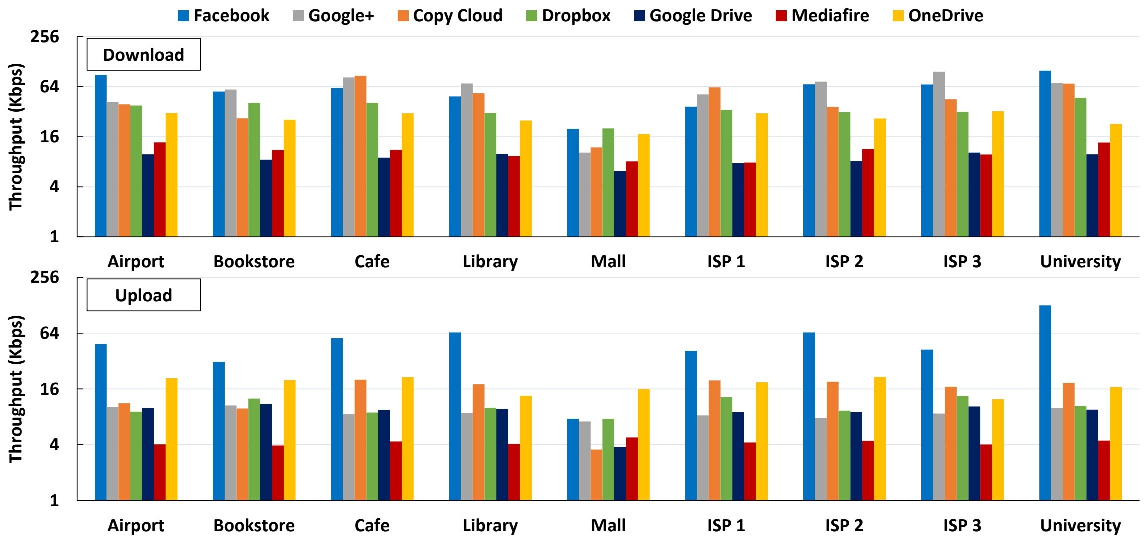


Figure A.20: Download and upload throughput for various locations with a 2.42 KB photo (log-log scale)

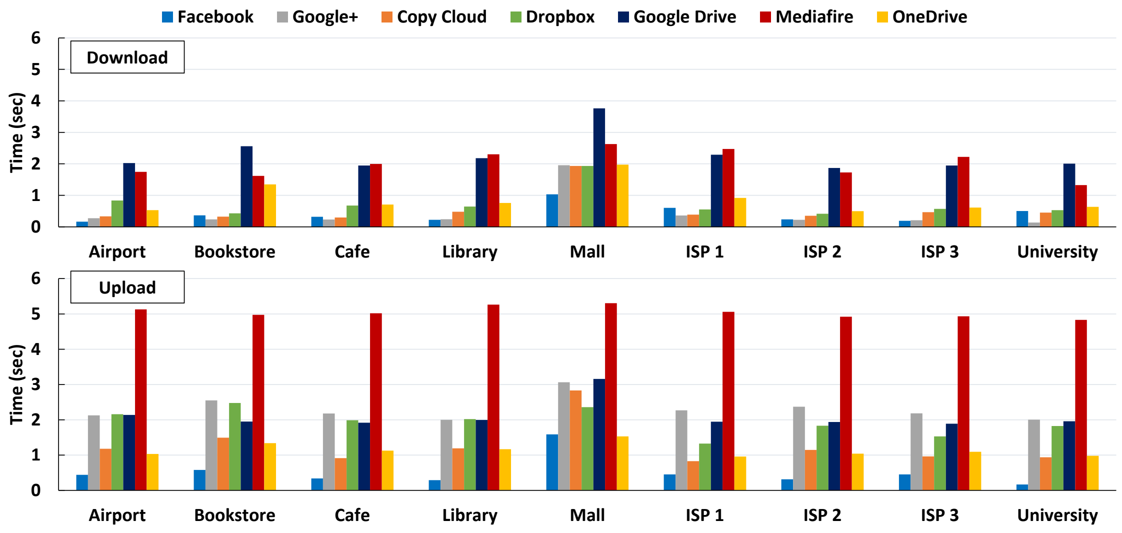


Figure A.21: Download and upload time for various locations with a 4.89 KB photo (log scale)

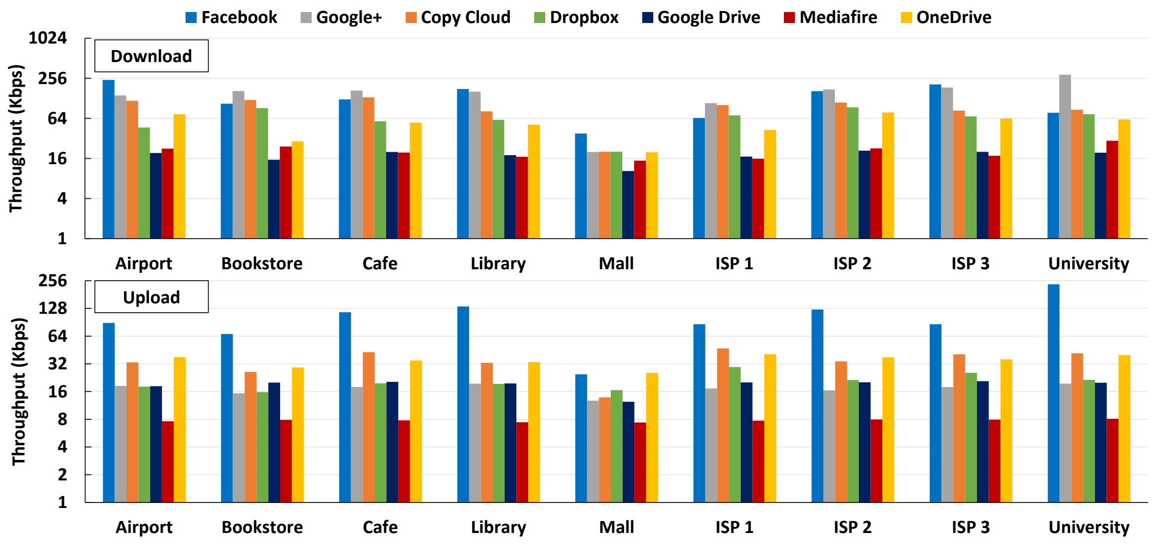


Figure A.22: Download and upload throughput for various locations with a 4.89 KB photo (log-log scale)

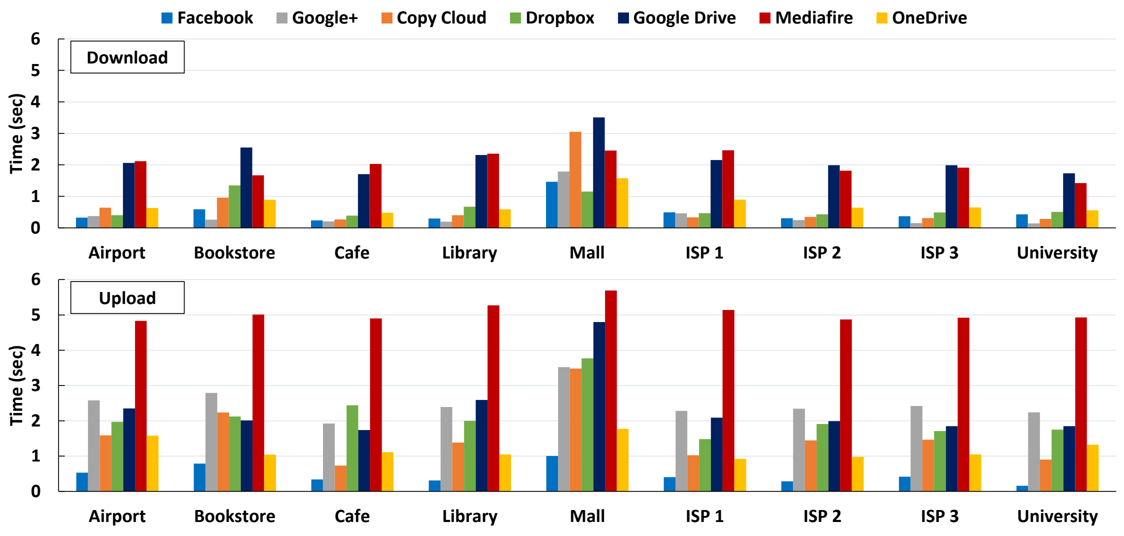


Figure A.23: Download and upload time for various locations with a 9.89 KB photo (log scale)

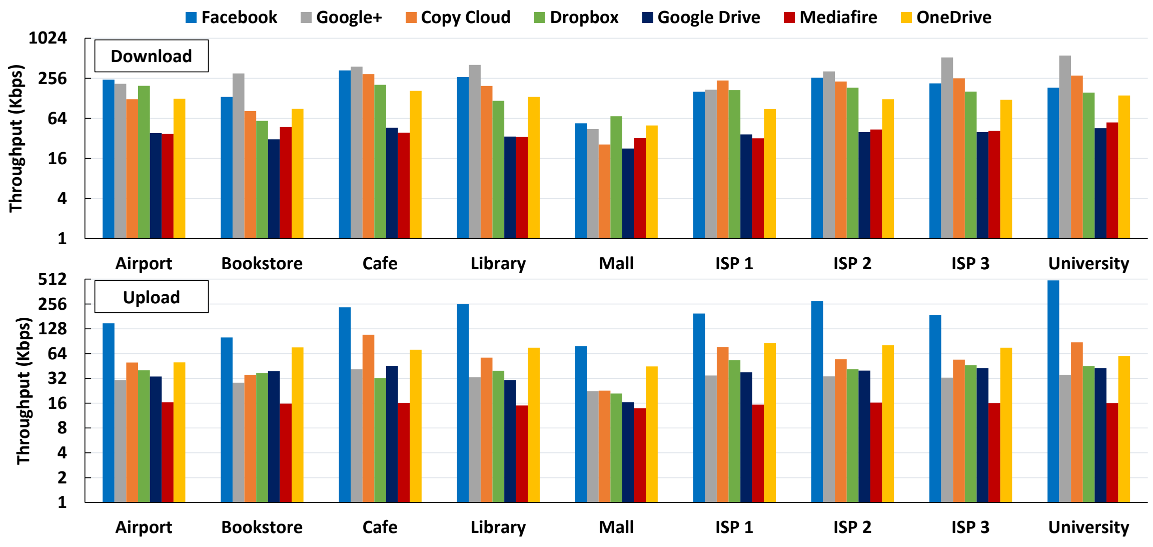


Figure A.24: Download and upload throughput for various locations with a 9.89 KB photo (log-log scale)

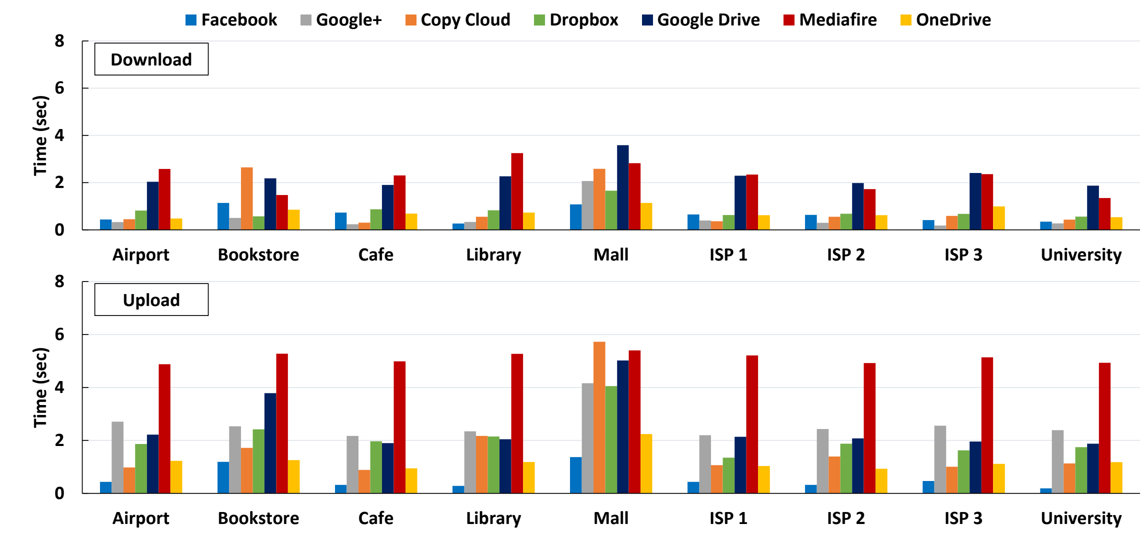


Figure A.25: Download and upload time for various locations with a 21.98 KB photo (log scale)

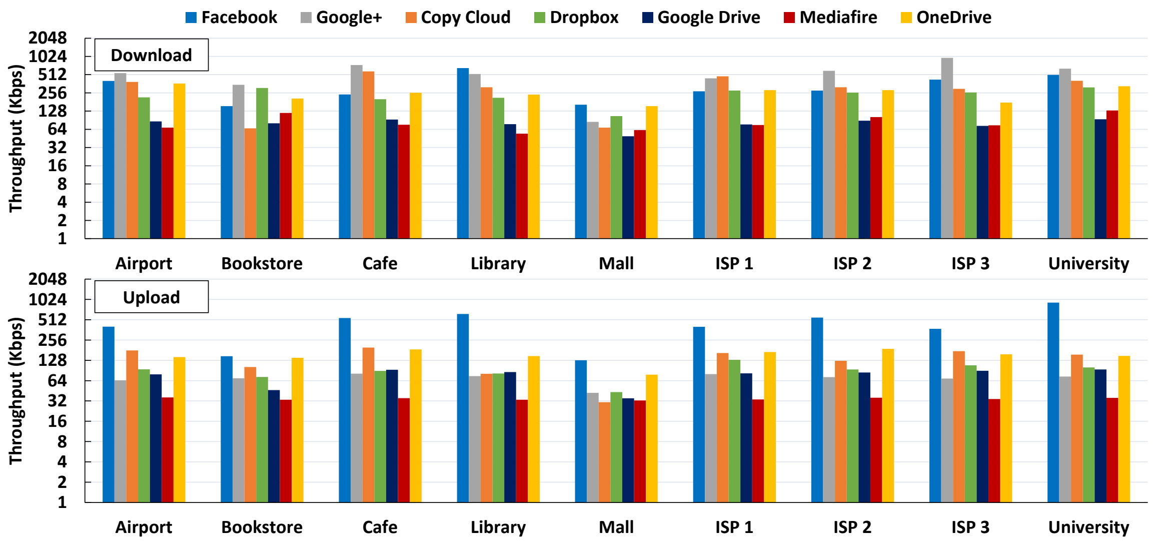


Figure A.26: Download and upload throughput for various locations with a 21.98 KB photo (log-log scale)

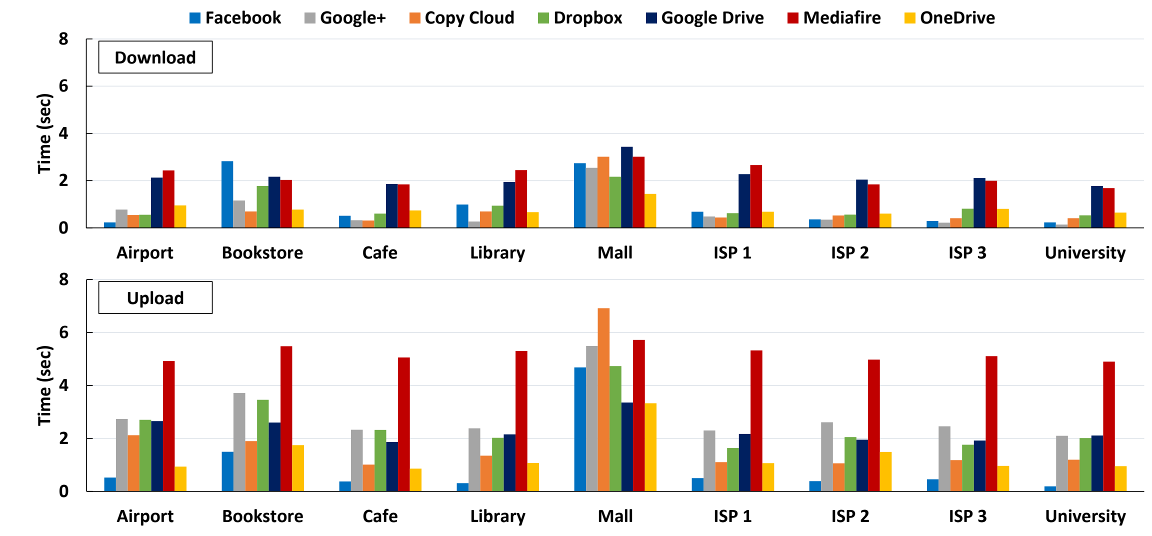


Figure A.27: Download and upload time for various locations with a 37.81 KB photo (log scale)

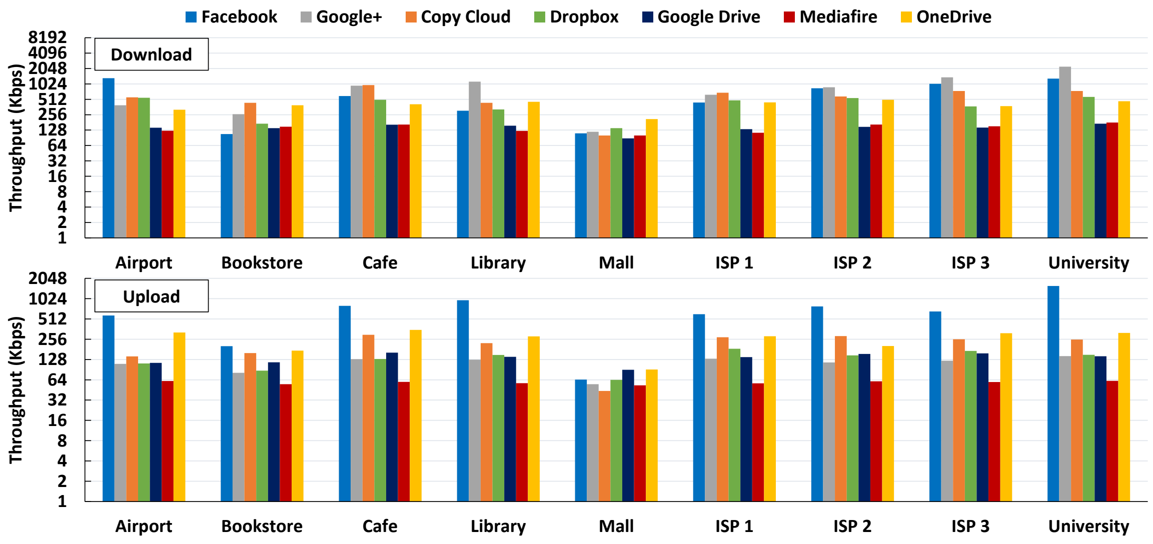


Figure A.28: Download and upload throughput for various locations with a 37.81 KB photo (log-log scale)

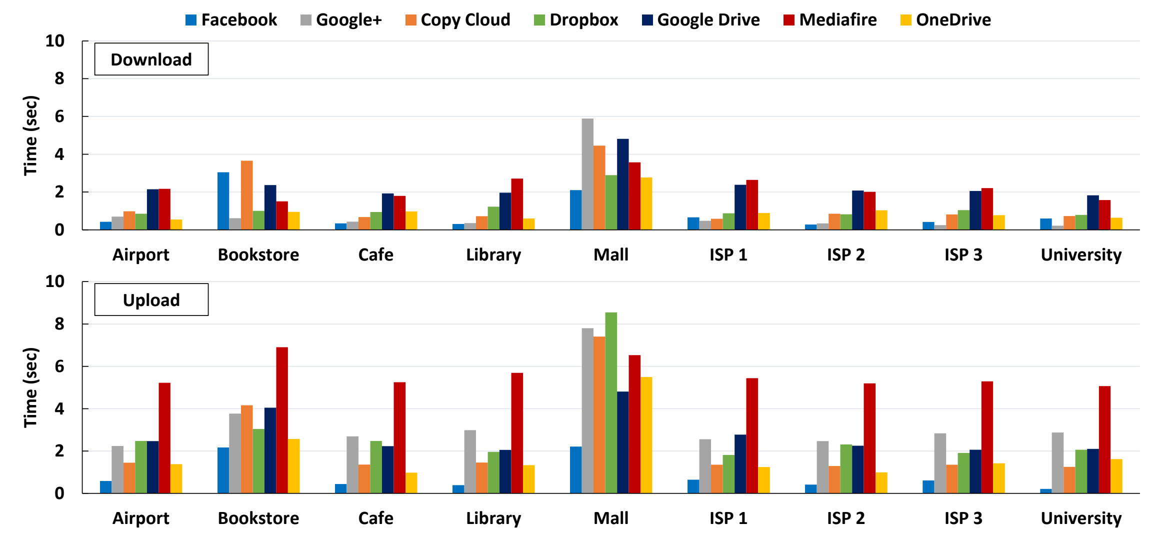


Figure A.29: Download and upload time for various locations with a 79.08 KB photo (log scale)

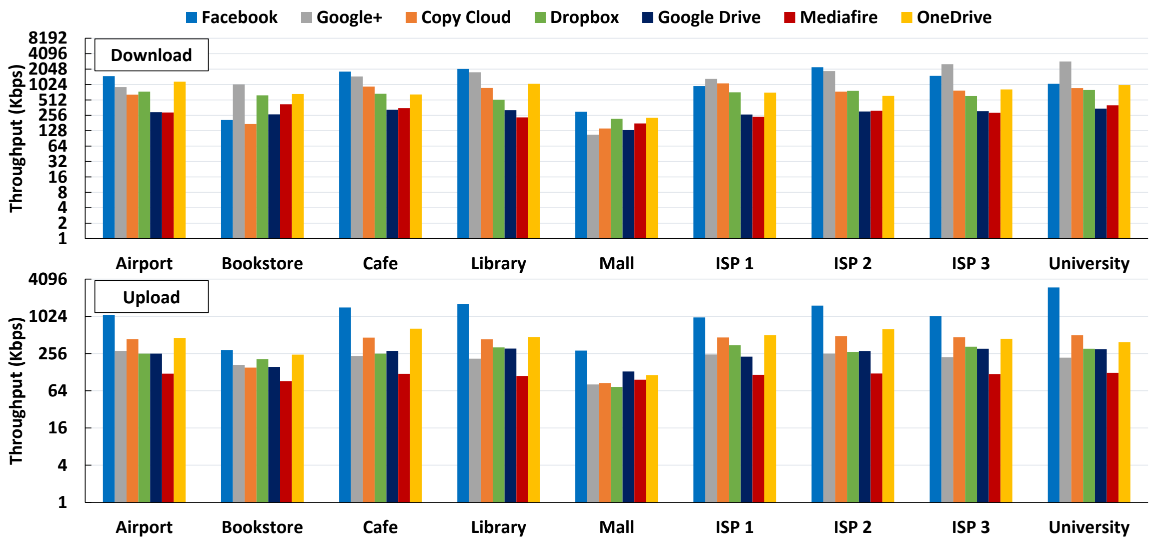


Figure A.30: Download and upload throughput for various locations with a 79.08 KB photo (log-log scale)

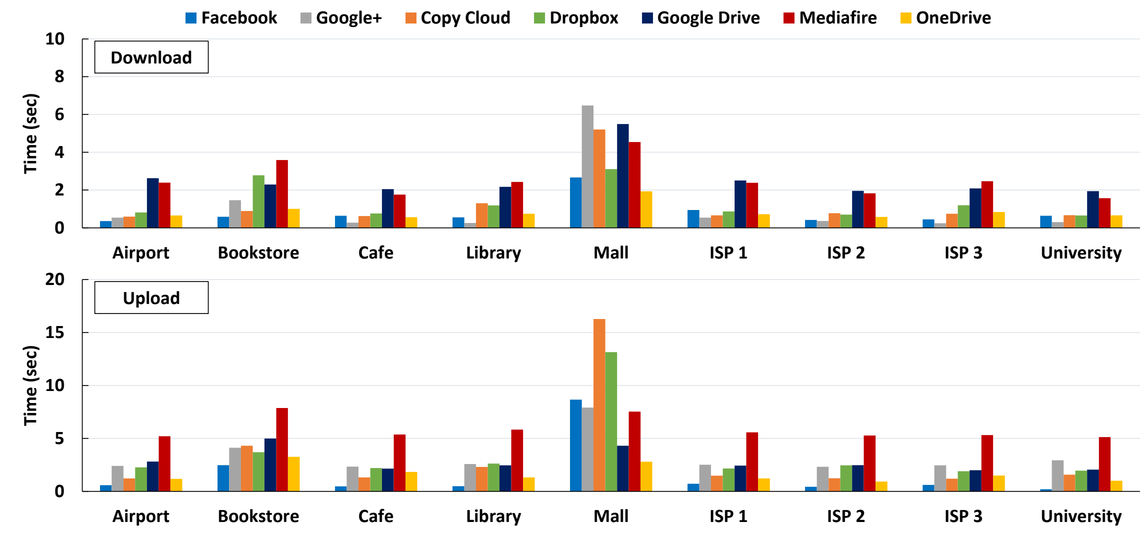


Figure A.31: Download and upload time for various locations with a 105.19 KB photo (log scale)

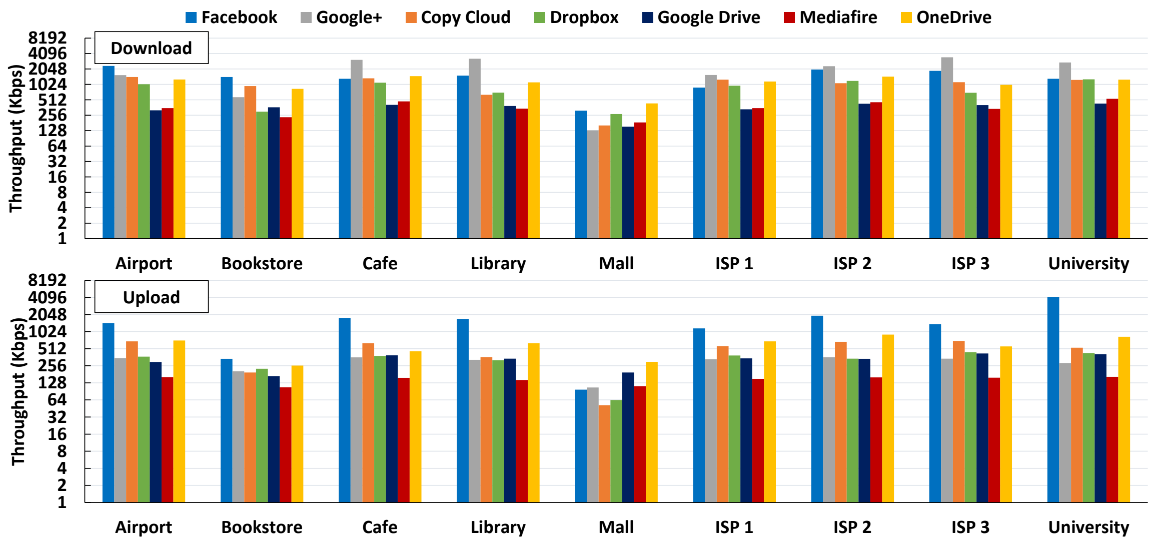


Figure A.32: Download and upload throughput for various locations with a 105.19 KB photo (log-log scale)

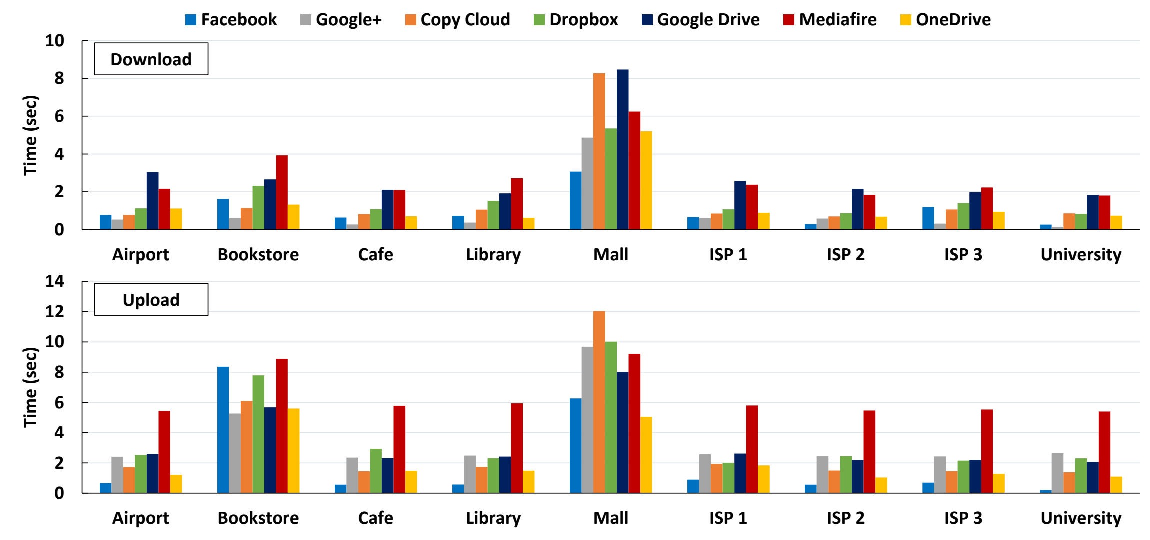


Figure A.33: Download and upload time for various locations with a 165.38 KB photo (log scale)

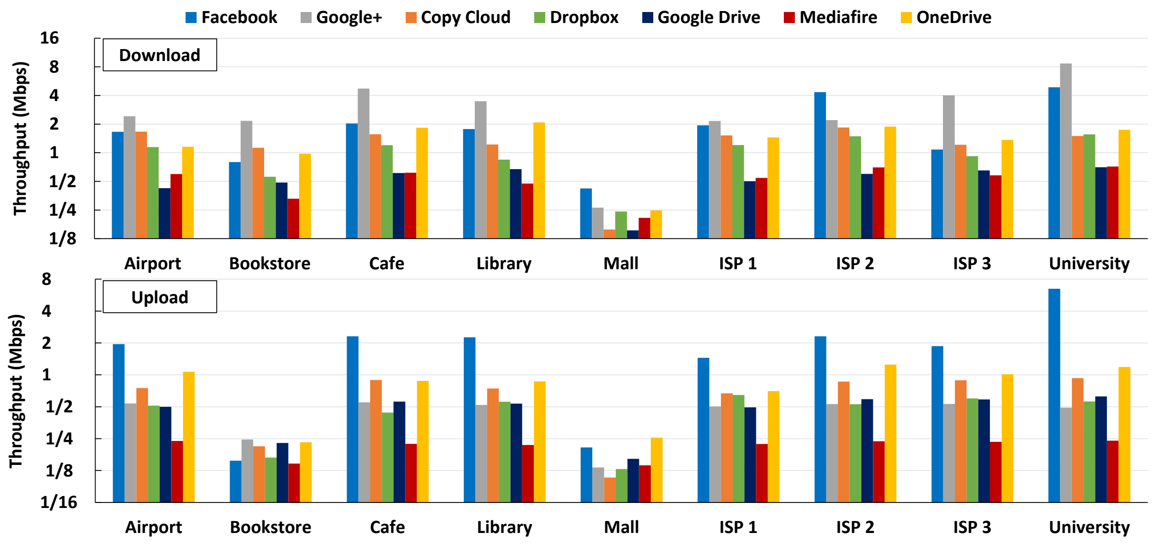


Figure A.34: Download and upload throughput for various locations with a 165.38 KB photo (log-log scale)

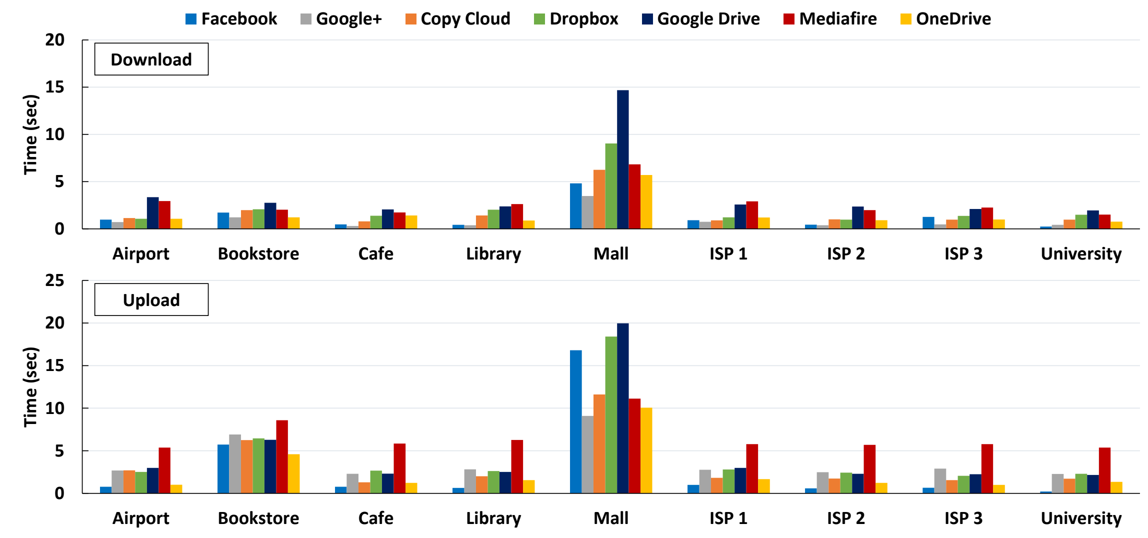


Figure A.35: Download and upload time for various locations with a 200 KB photo (log scale)

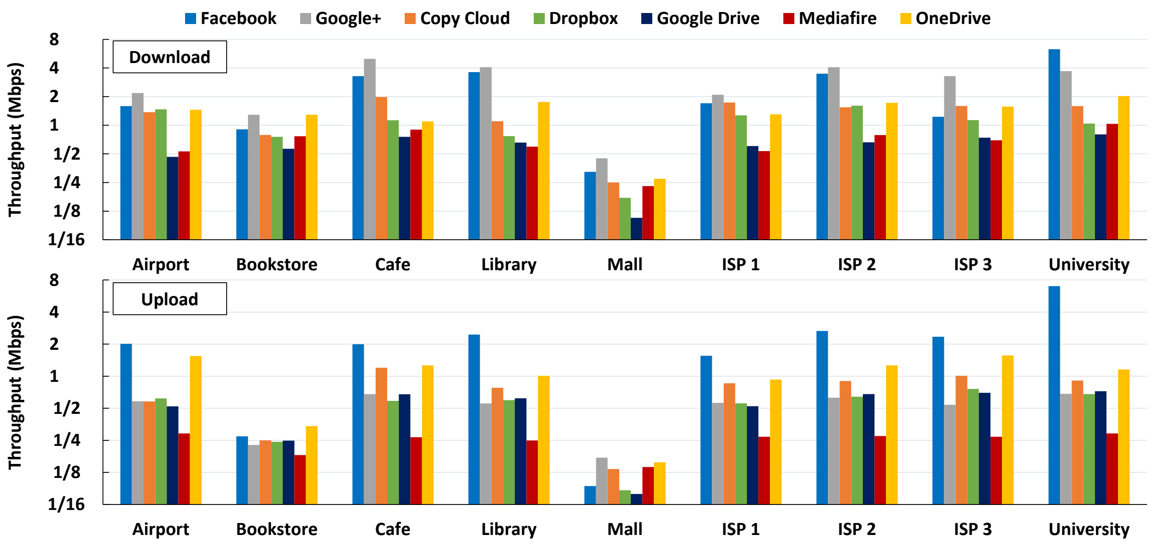


Figure A.36: Download and upload throughput for various locations with a 200 KB photo (log-log scale)

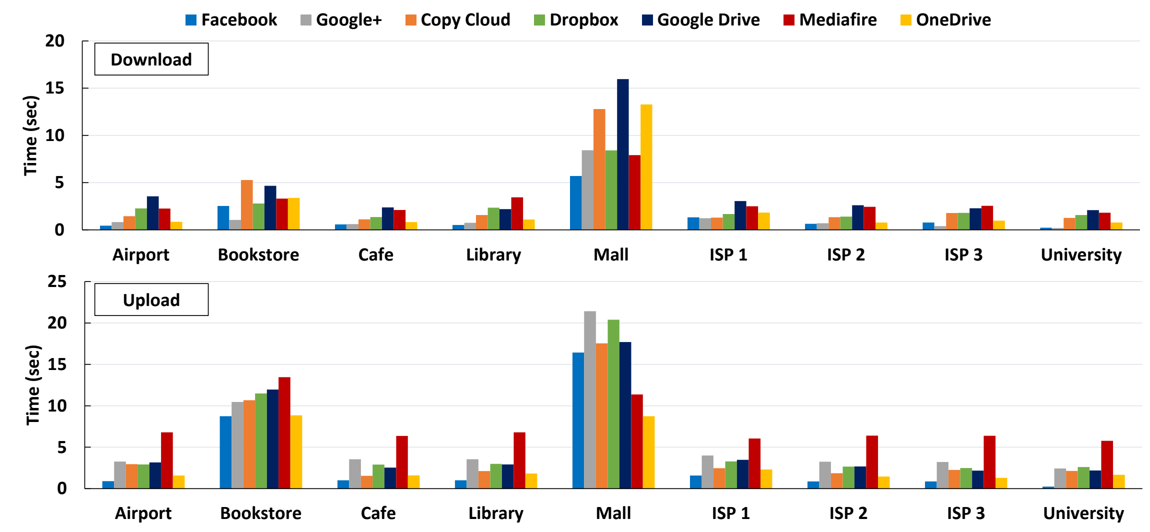


Figure A.37: Download and upload time for various locations with a 400 KB photo (log scale)

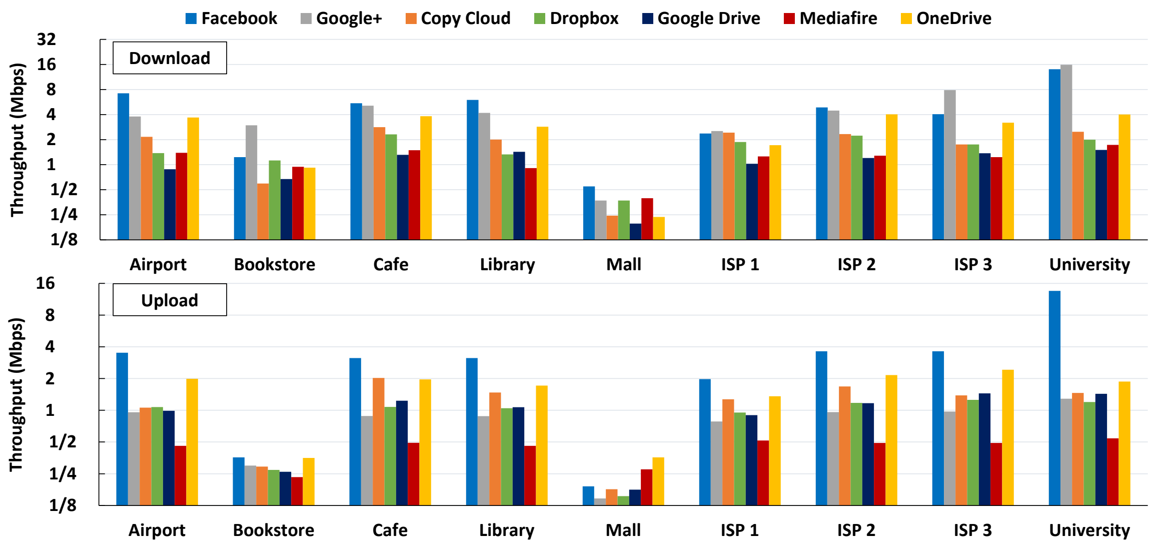


Figure A.38: Download and upload throughput for various locations with a 400 KB photo (log-log scale)

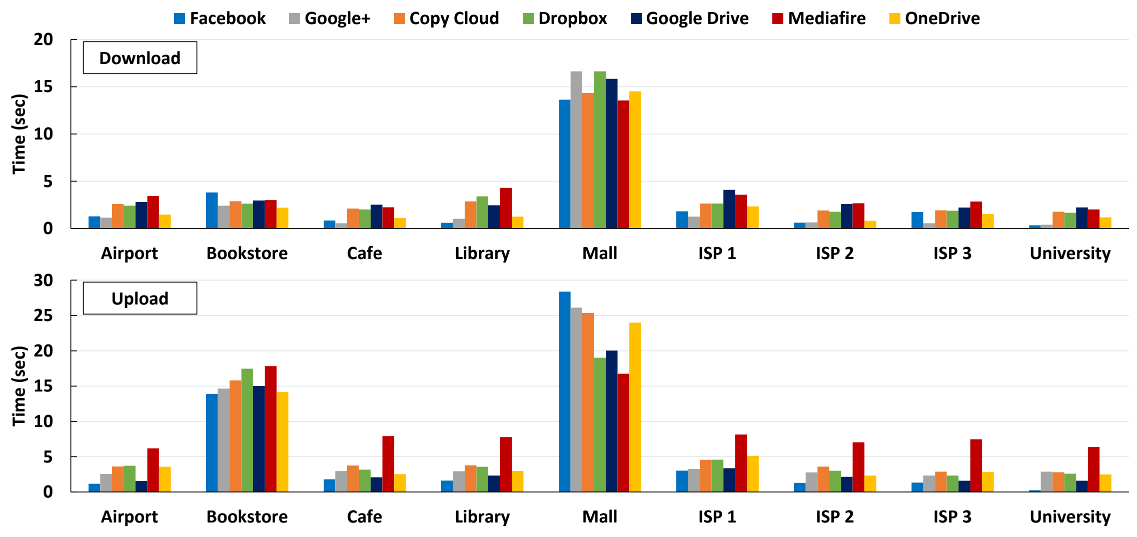


Figure A.39: Download and upload time for various locations with a 718.43 KB photo (log scale)

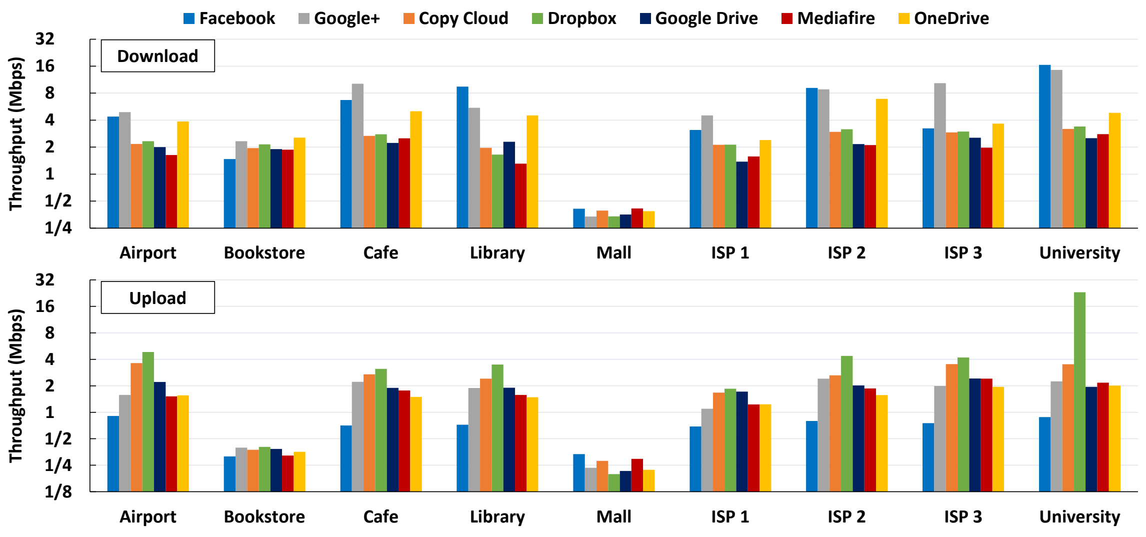


Figure A.40: Download and upload throughput for various locations with a 718.43 KB photo (log-log scale)

A.2.2 Video Files

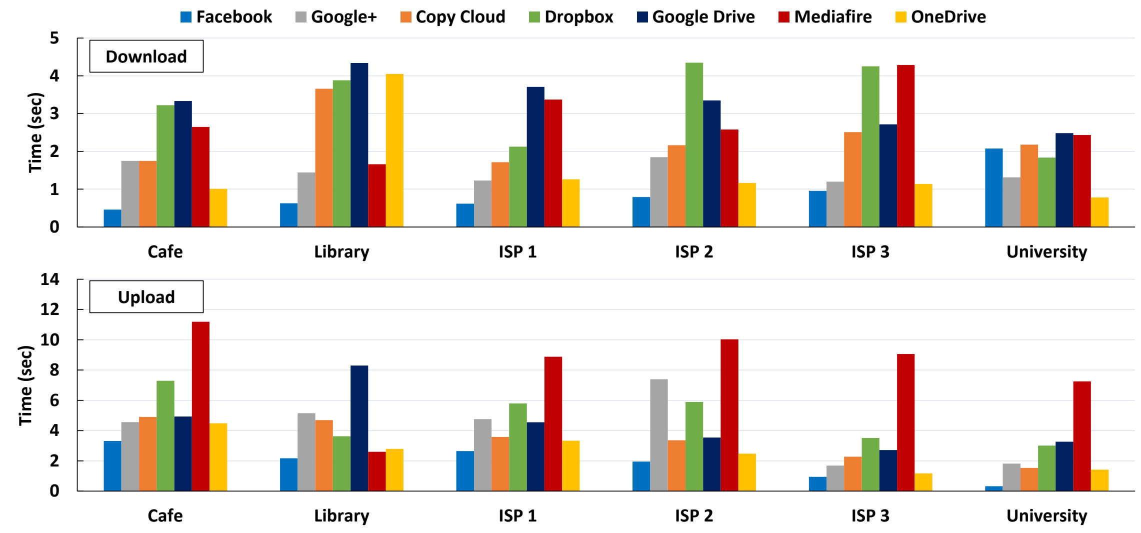


Figure A.41: Download and upload time for various locations with a 1 MB video (log scale)

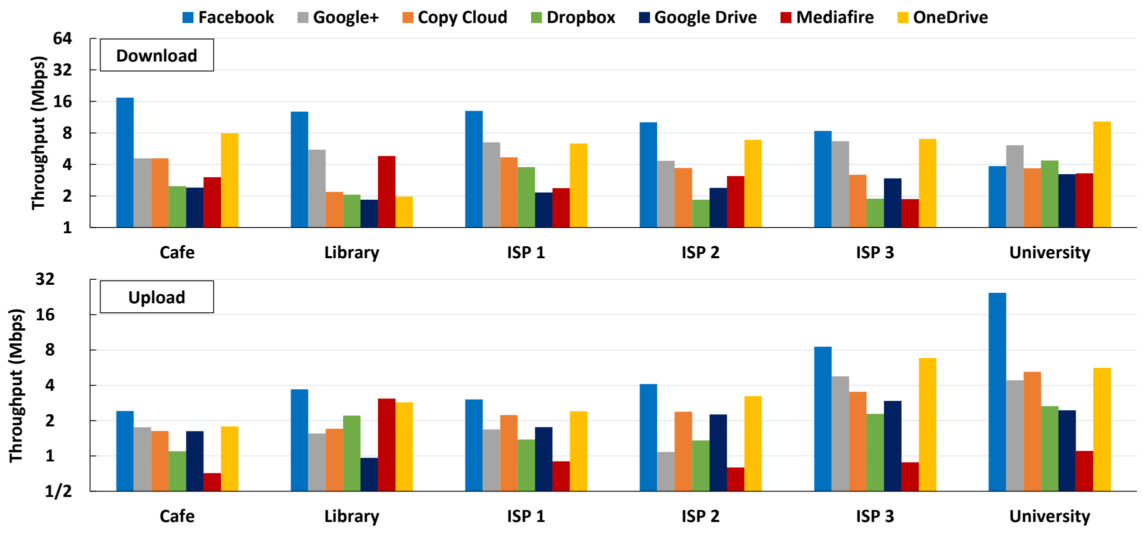


Figure A.42: Download and upload throughput for various locations with a 1 MB video (log-log scale)

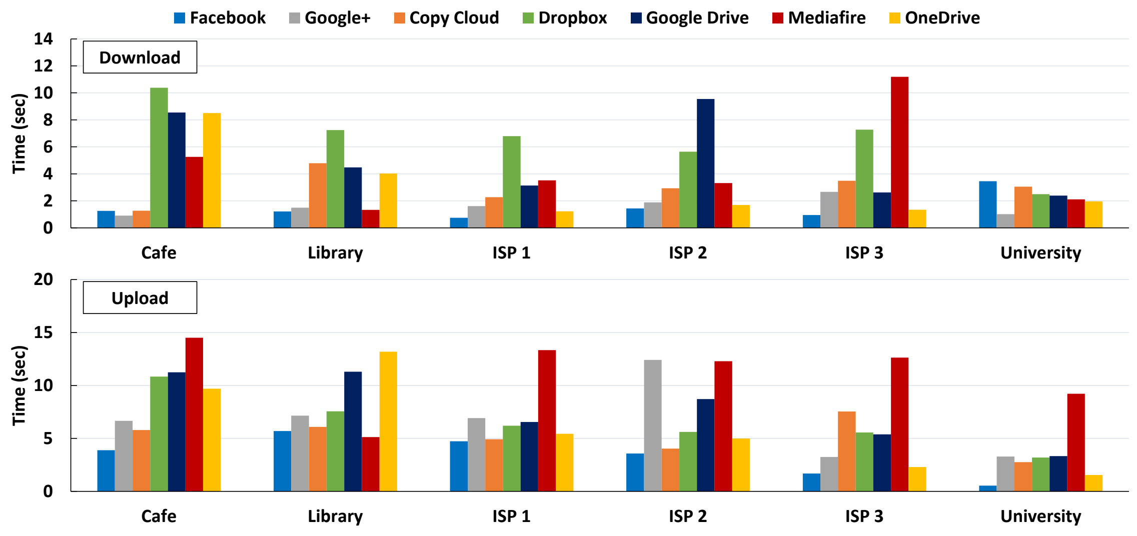


Figure A.43: Download and upload time for various locations with a 2 MB video (log scale)

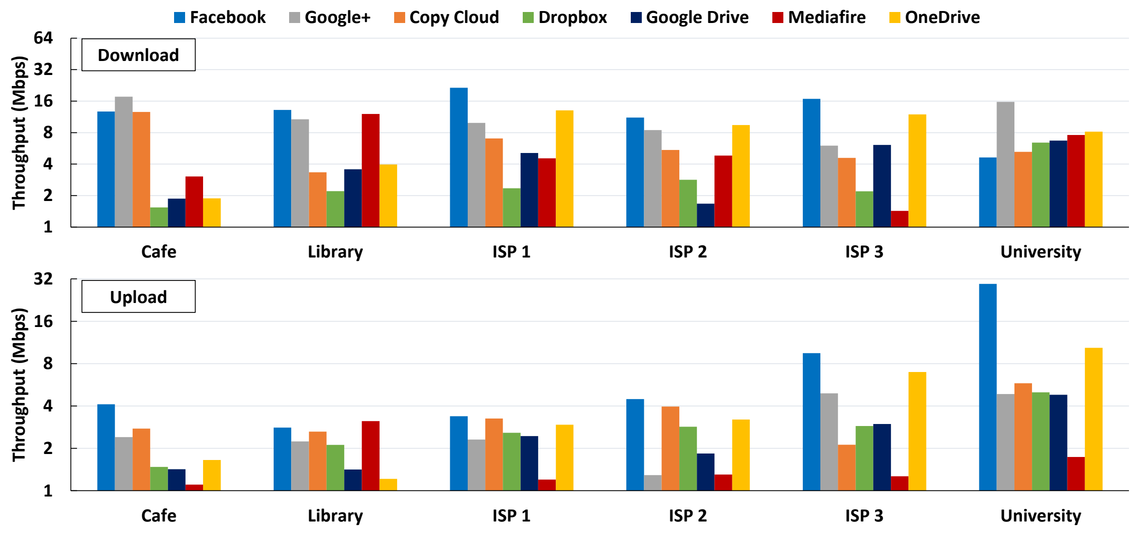


Figure A.44: Download and upload throughput for various locations with a 2 MB video (log-log scale)

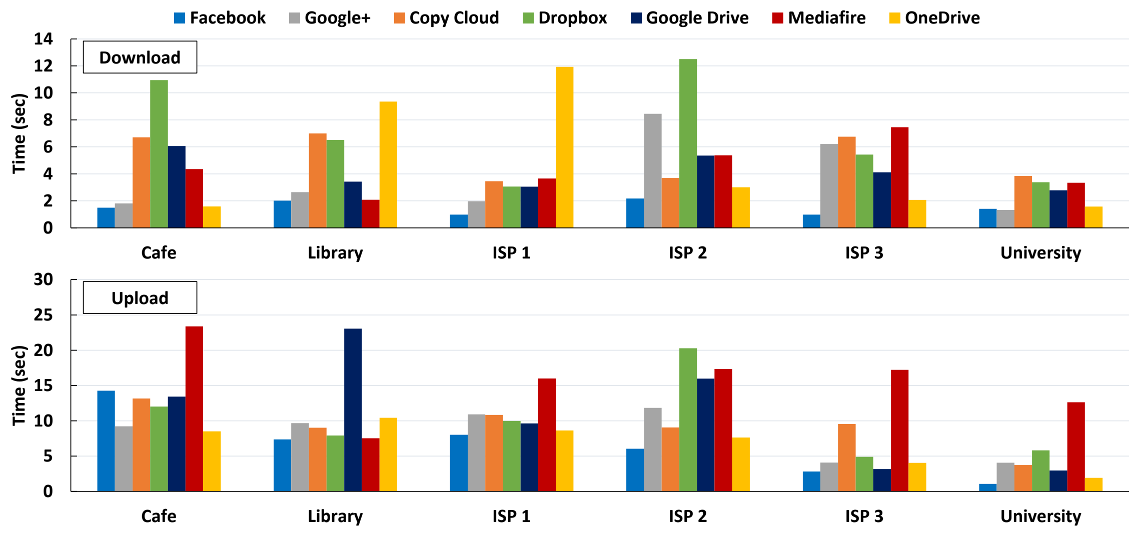


Figure A.45: Download and upload time for various locations with a 4 MB video (log scale)

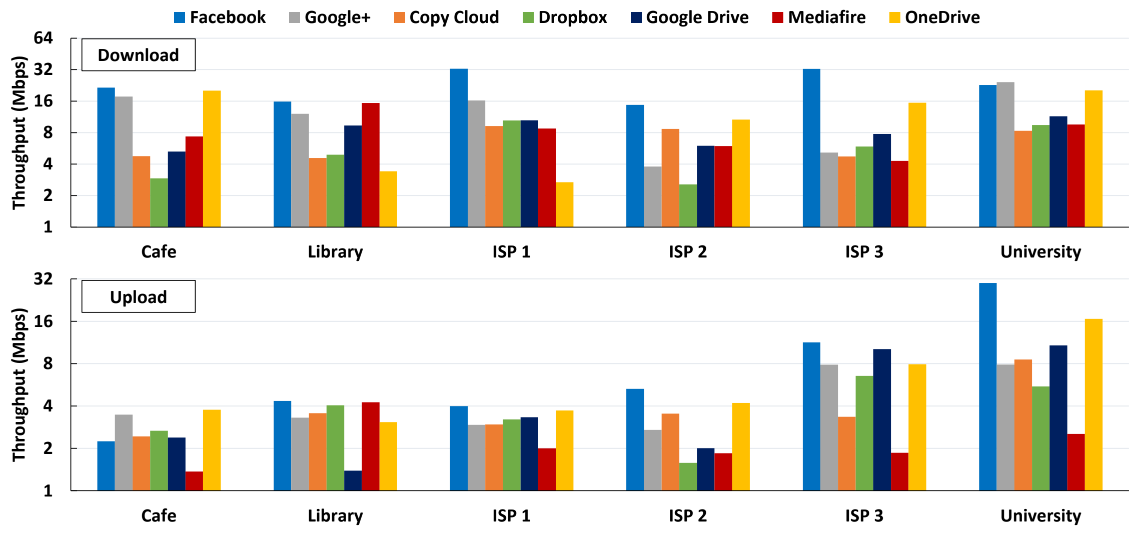


Figure A.46: Download and upload throughput for various locations with a 4 MB video (log-log scale)

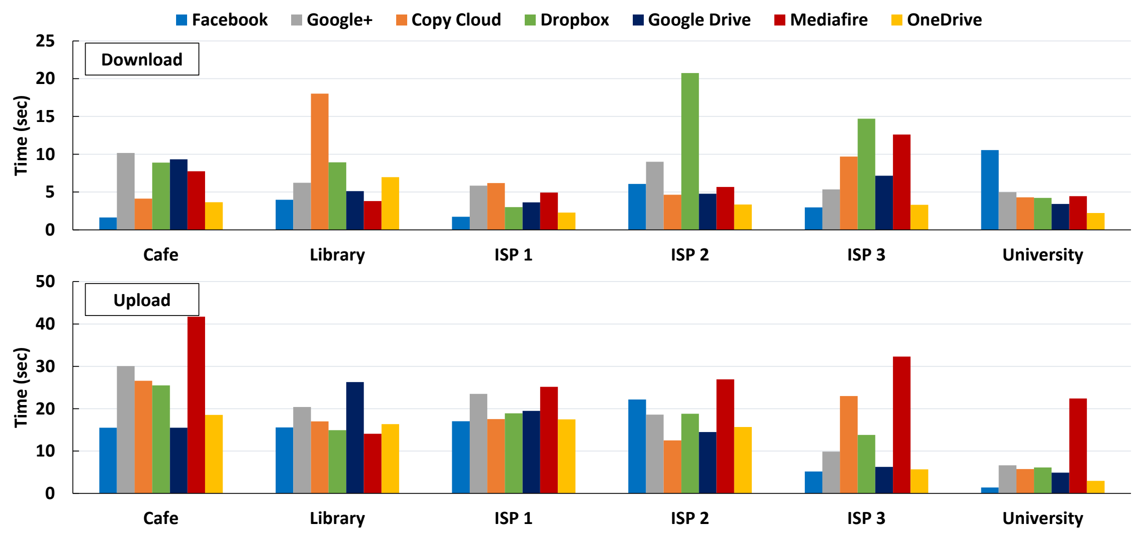


Figure A.47: Download and upload time for various locations with a 8 MB video (log scale)

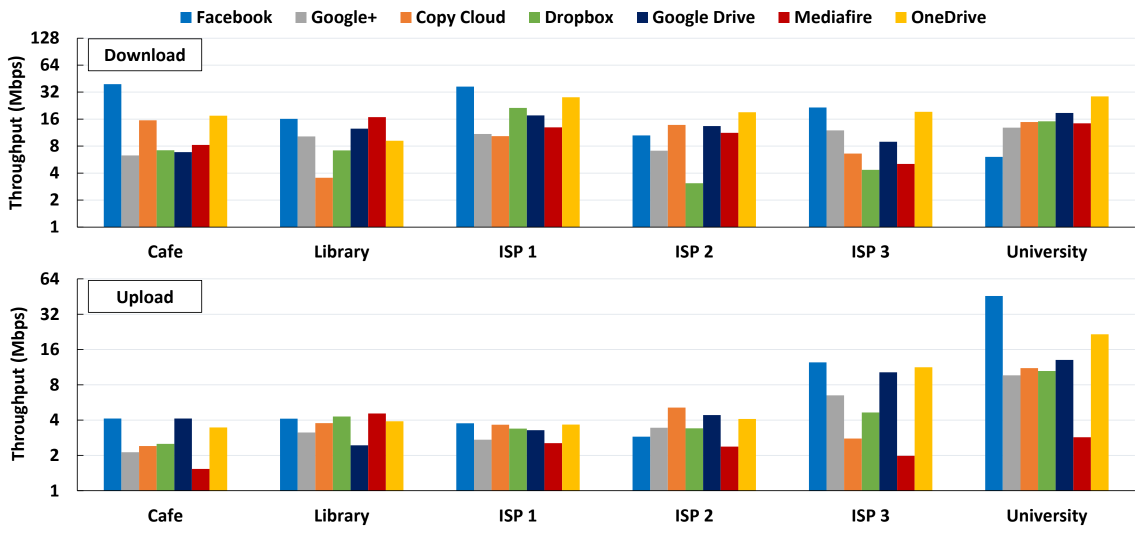


Figure A.48: Download and upload throughput for various locations with a 8 MB video (log-log scale)

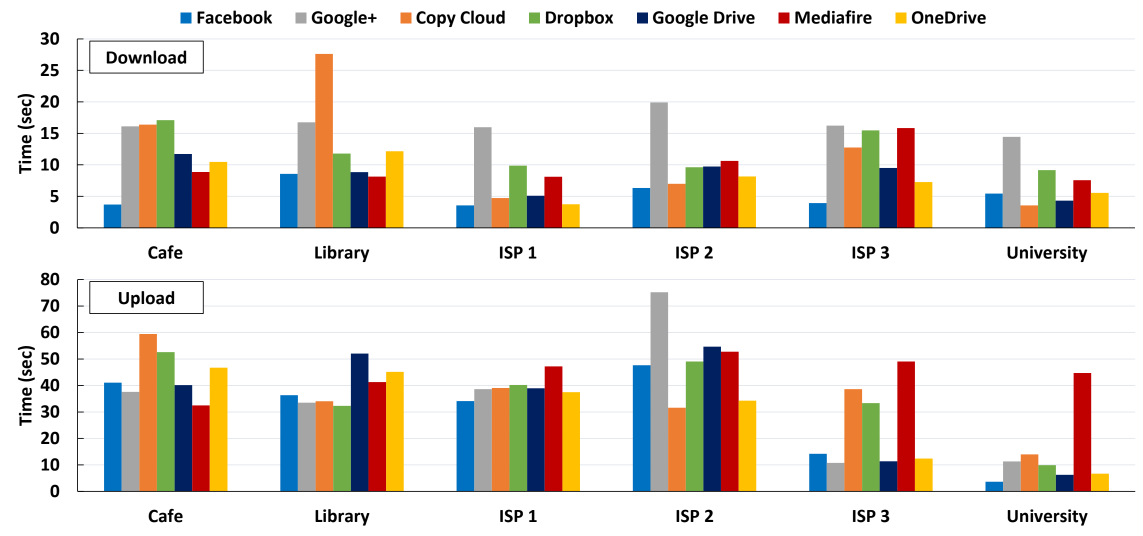


Figure A.49: Download and upload time for various locations with a 16 MB video (log-log scale)

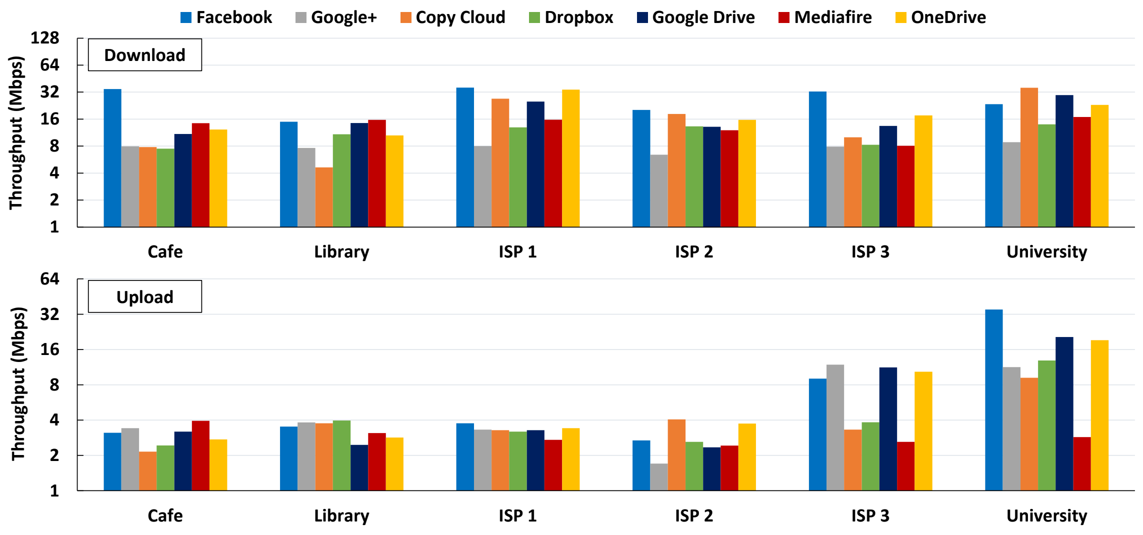


Figure A.50: Download and upload throughput for various locations with a 16 MB video (log-log scale)

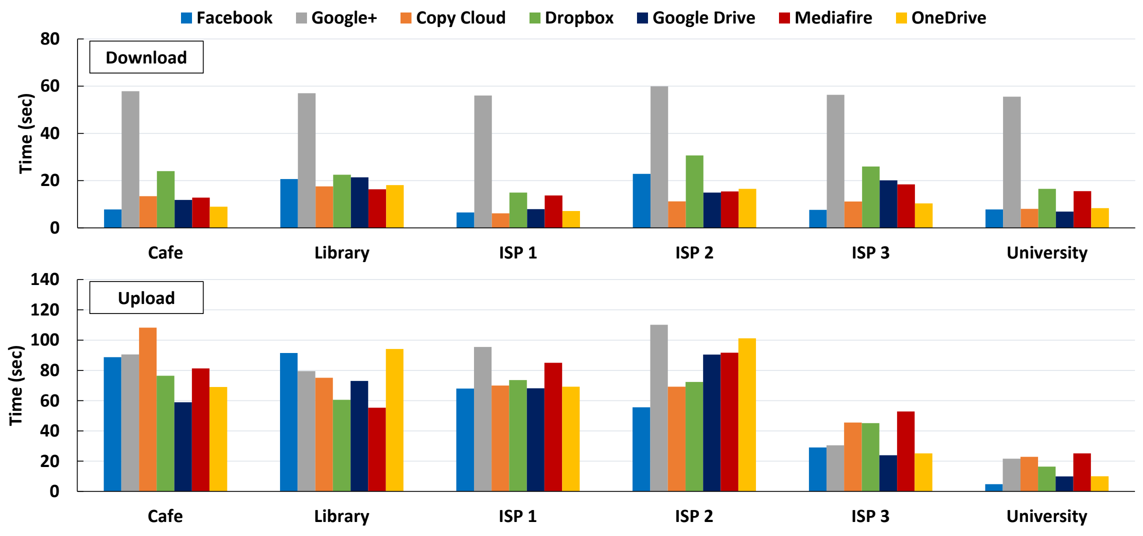


Figure A.51: Download and upload time for various locations with a 32 MB video (log scale)

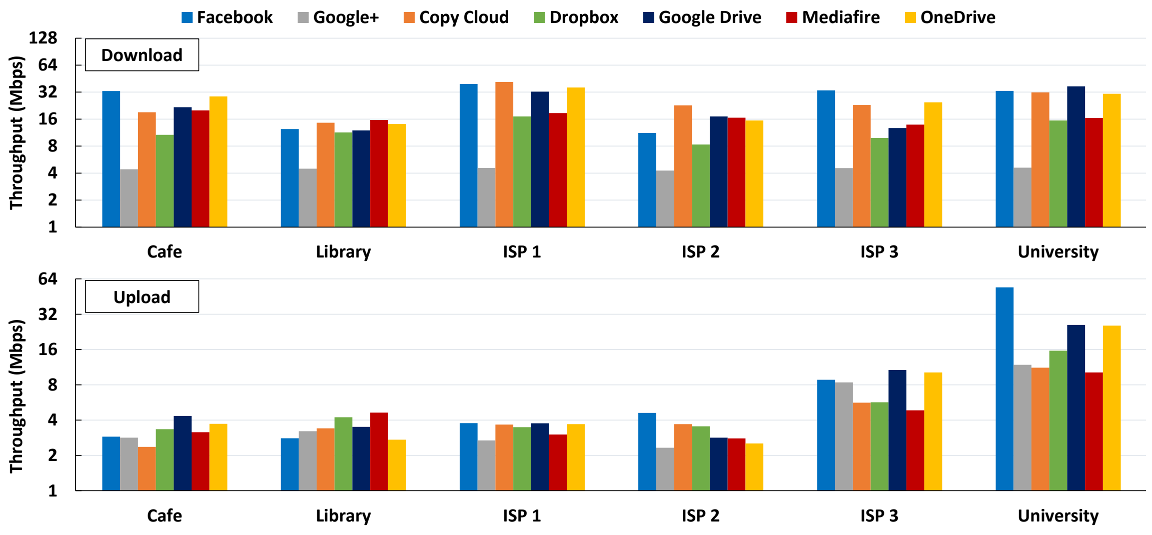


Figure A.52: Download and upload throughput for various locations with a 32 MB video (log-log scale)

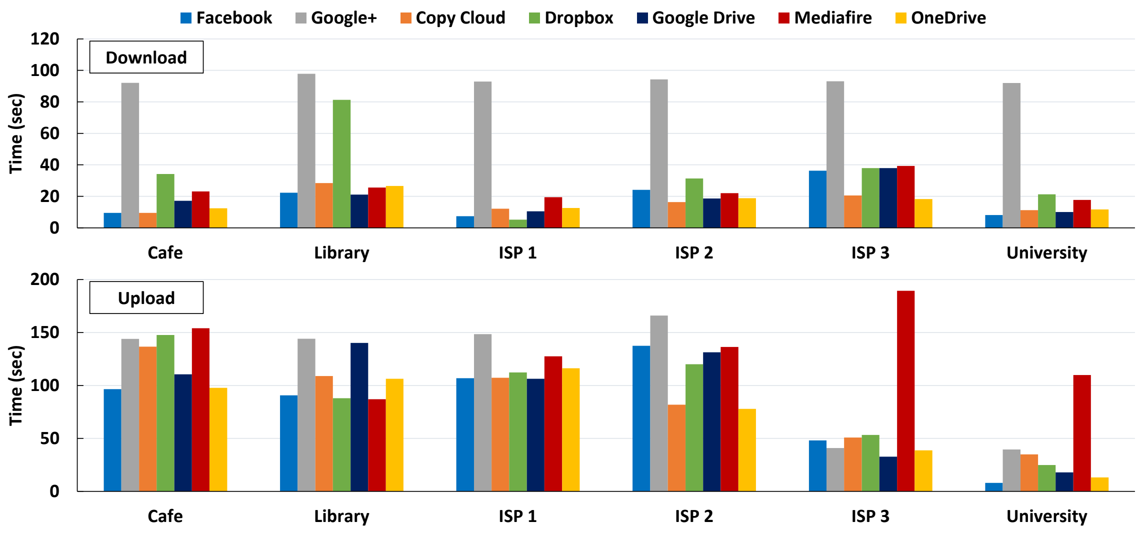


Figure A.53: Download and upload time for various locations with a 50 MB video (log scale)

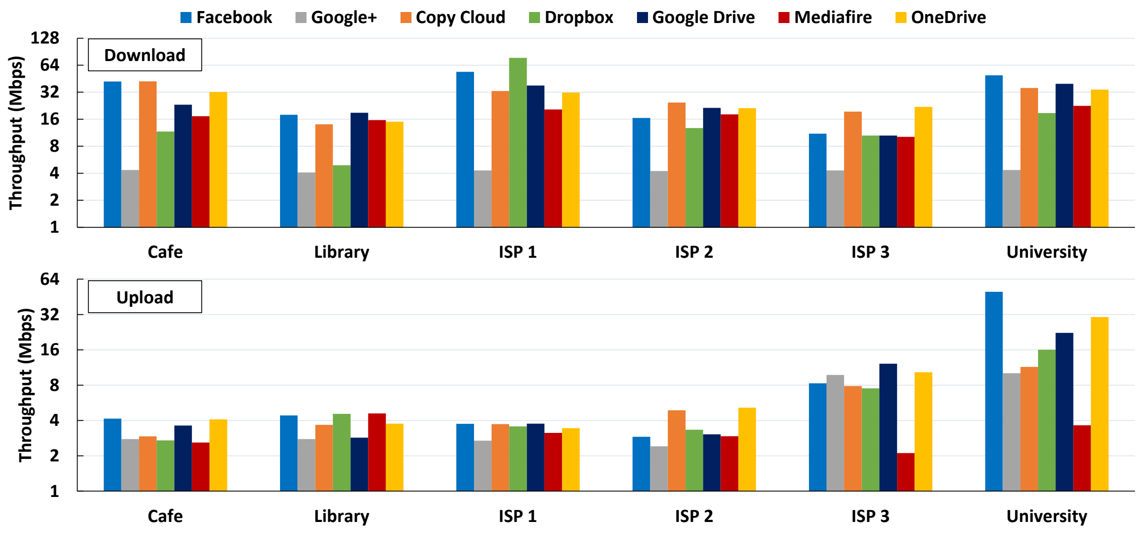


Figure A.54: Download and upload throughput for various locations with a 50 MB video (log-log scale)

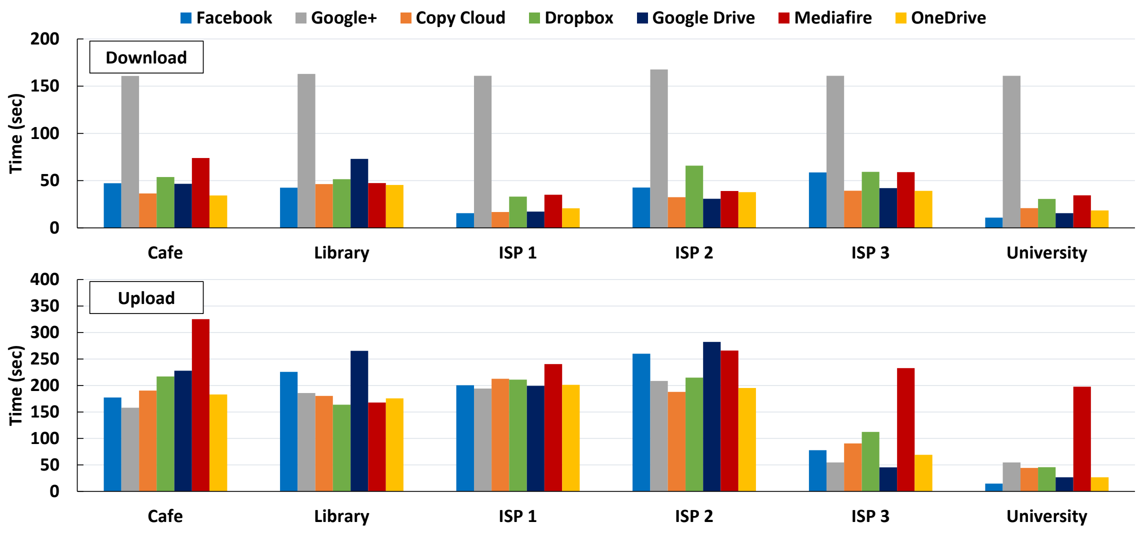


Figure A.55: Download and upload time for various locations with a 100 MB video (log scale)

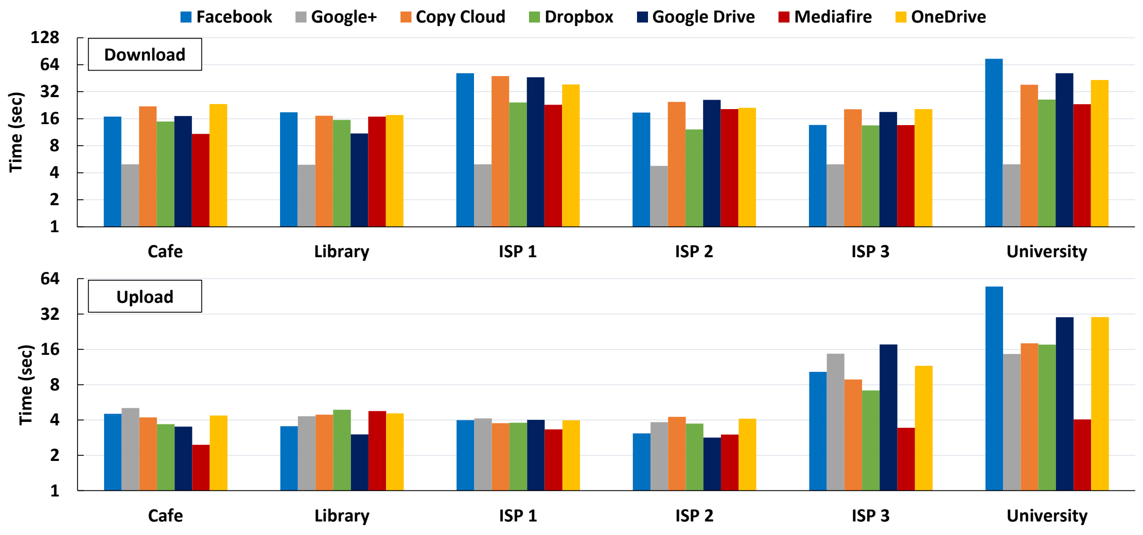


Figure A.56: Download and upload throughput for various locations with a 100 MB video (log-log scale)

Bibliography

[1] Luca Maria Aiello and Giancarlo Ruffo. Lotusnet: Tunable privacy for distributed online social network services. *Comput. Commun.*, 35(1):75–88, January 2012.

[2] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: An online social network with user-defined privacy. ACM SIGCOMM '09.

[3] Barracuda Networks, Inc. Copy cloud rest api. `https://developers.copy.com/documentation`. Accessed: 2016-03-24.

[4] Oleksandr Bodriagov, Gunnar Kreitz, and Sonja Buchegger. Access control in decentralized online social networks: Applying a policy-hiding cryptographic scheme and evaluating its performance. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom 2014 Workshops, Budapest, Hungary, March 24-28, 2014*, pages 622–628, 2014.

[5] Daniel Bosk and Sonja Buchegger. Privacy-preserving access control in decentralized storage for online social networks. In *10th International IFIP Summer School on Privacy and Identity Management*, 2015.

[6] Tim Bray. The javascript object notation (json) data interchange format, 2014.

[7] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: Early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, SNS '09, pages 46–52, New York, NY, USA, 2009. ACM.

[8] Pedro Casas, Andreas Sackl, Sebastian Egger, and Raimund Schatz. Youtube amp; facebook quality of experience in mobile broadband networks. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 1269–1274, Dec 2012.

[9] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communications Magazine*, 47(12):94–101, Dec 2009.

[10] Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. Decentralized online social networks. In *Handbook of Social Network Technologies and Applications*, pages 349–378. Springer, 2010.

[11] Idilio Drago, Enrico Bocchi, Marco Mellia, Herman Slatman, and Aiko Pras. Benchmarking personal cloud storage. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 205–212, New York, NY, USA, 2013. ACM.

[12] Idilio Drago, Marco Mellia, Maurizio M. Munafo, Anna Sperotto, Ramin Sadre, and Aiko Pras. Inside dropbox: Understanding personal cloud storage services. In *Proceedings of the 2012 ACM Conference on Internet Measurement*, IMC '12, pages 481–494, New York, NY, USA, 2012. ACM.

[13] Dropbox, Inc. Dropbox android sdk. `https://www.dropbox.com/developers/core/sdks/android`.

[14] Dropbox, Inc. Dropbox oauth guide. `https://www.dropbox.com/developers/reference/oauth-guide`. Accessed: 2016-03-24.

[15] Esra Erdin, Eric Klukovich, Gurhan Gunduz, and Mehmet Hadi Gunes. Posn: A personal online social network. In *ICT Systems Security and Privacy Protection*, pages 51–66. Springer International Publishing, 2015.

[16] Esra Erdin, Eric Klukovich, and Mehmet Hadi Gunes. An analysis of friend circles of facebook users. In *The 9th IEEE Workshop on Network Measurements (WNM)*, Clearwater Beach, FL, 26 Oct 2015 2015.

[17] Esra Erdin, Chris Zachor, and Mehmet Hadi Gunes. How to find hidden users: A survey of attacks on anonymity networks. *IEEE Communications Surveys Tutorials*, 17(4):2296–2316, Fourthquarter 2015.

[18] Facebook, Inc. Facebook android sdk. `https://developers.facebook.com/docs/android`. Accessed: 2016-03-24.

[19] Google, Inc. Android development. `http://developer.android.com/develop/index.html`. Accessed: 2016-03-24.

[20] Google, Inc. Google drive api. `https://developers.google.com/drive/android/`.

[21] Google, Inc. Picasa android sdk. `https://developers.google.com/picasa-web/docs/2.0/developers_guide_java#Audience`. Accessed: 2016-03-24.

[22] Raul Gracia-Tinedo, Marc Sanchez Artigas, Adrian Moreno-Martinez, Cristian Cotes, and Pedro Garcia Lopez. Actively measuring personal cloud storage. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 301–308, June 2013.

[23] Josh Halliday. Facebook urged to tighten privacy settings after harvest of user data, August 2015. http://www.theguardian.com/technology/2015/aug/09/facebook-privacy-settings-users-mobile-phone-number.

[24] Sonia Jahid, Shirin Nilizadeh, Prateek Mittal, Nikita Borisov, and Apu Kapadia. Decent: A decentralized architecture for enforcing privacy in online social networks. In *PerCom Workshops*, pages 326–332. IEEE Computer Society, 2012.

[25] Bingdong Li, Esra Erdin, Mehmet Hadi Gunes, George Bebis, and Todd Shipley. An overview of anonymity technology usage. *Computer Communications*, 36(12):1269 – 1283, 2013.

[26] Dongtao Liu, Amre Shakimov, Ramón Cáceres, Alexander Varshavsky, and Landon P. Cox. Confidant: Protecting osn data without locking it up. In *Proceedings of the 12th ACM/IFIP/USENIX International Conference on Middleware*, Middleware'11, pages 61–80, Berlin, Heidelberg, 2011. Springer-Verlag.

[27] Mediafire, Inc. Mediafire rest api. `http://www.mediafire.com/developers/core_api/1.3/getting_started/`. Accessed: 2016-03-24.

[28] Marco Maier Michael Durr and Florian Dorfmeister. Vegas – a secure and privacy-preserving peer-to-peer online social network. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 868–874, Sept 2012.

[29] Microsoft, Inc. Microsoft live android sdk. `https://msdn.microsoft.com/en-us/library/office/dn631814.aspx`. Accessed: 2016-03-24.

[30] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[31] Hema A. J. Narayanan and Mehmet Hadi Gunes. Ensuring access control in cloud provisioned healthcare systems. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 247–251, Jan 2011.

[32] Jeff Naruchitparames, MH Gunes, and Sushil J Louis. Friend recommendations in social networks using genetic algorithms and network topology. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2207–2214. IEEE, 2011.

[33] Shirin Nilizadeh, Sonia Jahid, Prateek Mittal, Nikita Borisov, and Apu Kapadia. Cachet: A decentralized architecture for privacy preserving social networking with caching. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 337–348, New York, NY, USA, 2012. ACM.

[34] Amre Shakimov, Harold Lim, Ramon Caceres, Landon P. Cox, Kevin Li, Dongtao Liu, and Alexander Varshavsky. Vis-a-vis: Privacy-preserving online social networking via virtual individual servers. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–10, Jan 2011.

[35] James Smith. Android asynchronous http client. `http://loopj.com/android-async-http/`. Accessed: 2016-03-24.

[36] Christo Wilson, Troy Steinbauer, Gang Wang, Alessandra Sala, Haitao Zheng, and Ben Y. Zhao. Privacy, availability and economics in the polaris mobile social network. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, HotMobile '11, pages 42–47, 2011.

[37] Xhemal Zenuni, Jaumin Ajdari, Florije Ismaili, and Bujar Raufi. Cloud storage providers: A comparison review and evaluation. In *Proceedings of the 15th International Conference on Computer Systems and Technologies*, CompSysTech '14, pages 272–277, New York, NY, USA, 2014. ACM.