Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Android Smart Cloud Storage

Semester Project

Rolf Scheuner

schrolf@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Gino Brunner, Simon Tanner
Prof. Dr. Roger Wattenhofer

July 4, 2017

Acknowledgements

I thank my supervisors Gino Brunner and Simon Tanner for their great support during my semester project. Whenever I stuck with a problem they helped me out with the right idea. In each meeting they shared their motivation and inspiration with me.

Abstract

The most widely used cloud storage providers for private users are Google Drive, Dropbox and Microsoft OneDrive. There are countless ways to access and manage data on each of these providers. From web interfaces over desktop apps to mobile apps. There are even apps supporting different cloud storage providers, but they all treat each cloud storage as single file system.

The aim of this project is to enable the users to handle their data as it was stored in one place while it is actually distributed over multiple cloud storages.

Unified Cloud Storage uses rclone in the background to read/write files from/to different cloud storage providers. It lets the user store their accounts and then define Unified Folders which can spread over multiple cloud drives.

Contents

Chapter 1

Introduction

1.1 Motivation

Cloud storage is the future way of storing data. The key advantage of cloud storage is its availability. The user's data is available on the smartphone, on the PC at work and at home. To share data with friends, it is enough to send them an invitation link to the folder one wants to share. Another big advantage is reliability. Most people do not have a backup solution for their locally stored private data, so if their hard disk fails their data is lost. When they have their data synchronised with a cloud storage provider they have a backup with almost no additional effort. Therefore it is no surprise that more and more people use cloud storage in their everyday life.

Today many cloud storage users use free offers form Dropbox, Google Drive or Microsoft OneDrive. Since these providers offer only a few gigabytes as free capacity, users often use more than one provider. However, when using more than one cloud storage provider, one has to access each cloud storage separately. This is clearly an undesirable additional effort which users have to make.

Therefore UCS (Unified Cloud Storage) aims to let the users access their data independently from where it is stored. It gives the user the usability as if all files were stored on one single cloud drive. Additionally a Unified Folder provides the combined free space of all connected *remotes*.

Features: / Apps:	Number of supported Cloud Storage Providers	Usage	Integrated FileExplorer	Backup Appdata	Sync between folders	Unified Storage	Encryption	Comment
Smart Cloud Storage	1	easy	yes	no	no	yes	no	
MyCLoud	4	easy	yes	no	no	no	no	Loads the webinterfaces
All Cloud Storage	8	easy	no	no	no	no	no	Only 2 Accounts for Free
Synchronize Ultimate	100	hard	yes	no	yes	no	yes	Only 2 Accounts for Free
Otixo	30	easy	yes	no	no	no	yes	
Unclouded	5	easy	yes	no	no	no	no	
CloudDrives	5	easy	no	no	no	no	no	Loads the webinterfaces
Unified Cloud Storage	10	easy	yes	no	yes	yes	yes	

Figure 1.1: Analyzed apps, March 29 2017

1.2 Related Work

In his work about Android Smart Cloud Storage, Marco Studer purposed to use *rclone* as cloud storage interface in order to support different cloud storage providers. [1]

To ensure this project does not just reproduce an existing application, available cloud storage related applications were tested (see figure 1.1). There were various applications supporting multiple cloud storage providers. Some support even up to a hundred different cloud storage providers, but we could not find a single application which enabled the user to combine their cloud storages.

Chapter 2

Unified Cloud Storage

The application developed during this project is named Unified Cloud Storage even if the project's title is Android Smart Cloud Storage. This is because the project was named after it's predecessor project, but the application needs an appropriate name in the Google Play Store.

UCS enables it's users to manage data independently from where the data is stored. The users access their data through Unified Folders which can be spread over different cloud storage providers. A Unified Folder in UCS appears to the users as a normal folder in any other file manager (see figure 2.1). UCS automatically distributes the data in a Unified Folder to the connected cloud storage providers. It uses a compiled binary of *rclone* to connect to the supported cloud storage providers and upload and download files.

2.1 Remote

A *remote* consists of a cloud storage provider and an access token belonging to this provider. The *remotes* are stored in the rclone configuration file, which is never directly accessed by UCS (see section 2.6). This configuration file can also be imported from another installation (e.g., from a desktop installation) of *rclone*. This makes it easy for *rclone* users to start using UCS.

The user has to define at least one *remote* before a Unified Folder can be created and therefore to use UCS. When defining a remote, the user has to authenticate himself and log in on the chosen cloud storage provider via web browser. *Rclone* then stores the generated access token. The user also has the ability to reauthenticate an existing remote (e.g., if a token is no longer valid).

The user can also create an encrypted *remote*. An encrypted *remote* uses a normal *remote* to access the cloud storage but all files are encrypted before uploading and decrypted after downloading. This helps the user to keep private data private independently from the terms of use of the specific cloud storage provider.

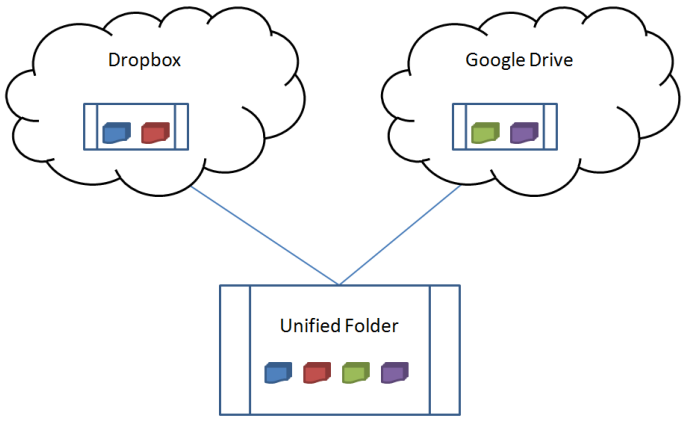

Figure 2.1: Schematic drawing of a Unified Folder spread over Dropbox and Google Drive

The user can create *remotes* with following providers (see also figure 2.2):

- Google Drive
- Dropbox
- Microsoft OneDrive
- Amazon Cloud Drive
- Hubic
- Yandex

Further *remotes* from following additional providers can be imported with the *rclone configuration import* function:

- Amazon S3
- Swift / Rackspace Cloudfiles / Memset Memstore
- Google Cloud Storage
- Backblaze B2

This second group of cloud storage providers can not be supported by UCS, because they have a highly specific authentication procedure in *rclone*.

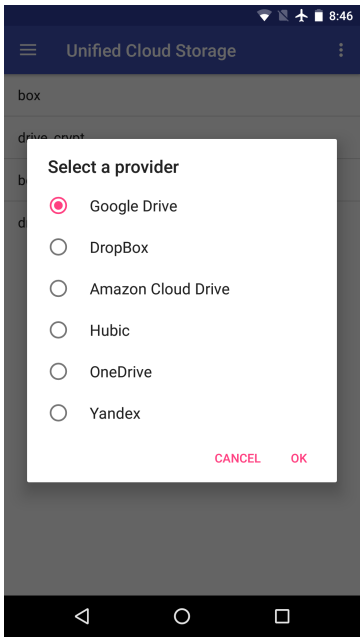

Figure 2.2: Available cloud storage providers

2.2 Unified Folder

Unified Folders have one or more *remotes* to which they distribute the contained data. On each *remote* a Unified Folder has its own subfolder. The data is distributed similarly as in a RAID 0: on upload UCS chooses the remote with the least data stored so far to upload the new file.

To define a Unified Folder the user has to pick at least one of the previously defined *remotes*. On creation on each selected *remote* a subfolder belonging to the created Unified Folder is created.

Rclone needs several seconds to list the files and subfolders of a folder. Therefore the folder structure of a Unified Folder is cached locally and only updated when the user tells UCS to reload the folder structure. It is also possible to only reload a subfolder of the Unified Folder. An update is necessary if data in the Unified Folder was changed through another access than UCS or when a Unified Folder is defined in UCS which has already data available in its subfolder on at least one remote.

When a Unified Folder is deleted in UCS, the corresponding subfolders on the *remotes* are not deleted. When the user recreates the Unified Folder with the same name, the data will still be available.

2.3 Upload Files

There are two ways to upload files to a Unified Folder:

First, while browsing through the folder structure in a Unified Folder, the user can select *upload* from the actions menu. Then a file-choose intent is sent and the user can choose a file to be uploaded. This requires a file explorer on Android and notifies the user if no file explorer is found on the system. Since Android Marshmallow there is a standard Android file explorer which can be accessed through *Settings* - *Storage* - *Explore* in stock Android.

Second, UCS registers itself on the share intent of Android, so a file can be sent to UCS which opens an activity to choose the path to upload the file. The URI (Uniform Resource Identifier) has different forms depending on the application which sent the intent. It is therefore not trivial to get the absolute file path (needed for *rclone*) from the URI.

While uploading UCS shows a notification, which disappears as soon as the upload has finished.

2.4 Download Files

When the user clicks a file it is downloaded to the standard download folder and opened as soon as the download is completed. When no application to open the file was found, the user gets a notification.

To download a file or folder without opening afterwards, the user selects the desired files and folders and clicks download from the options menu (see figure 2.3). The file or folder is then downloaded to the standard download folder of the Android device.

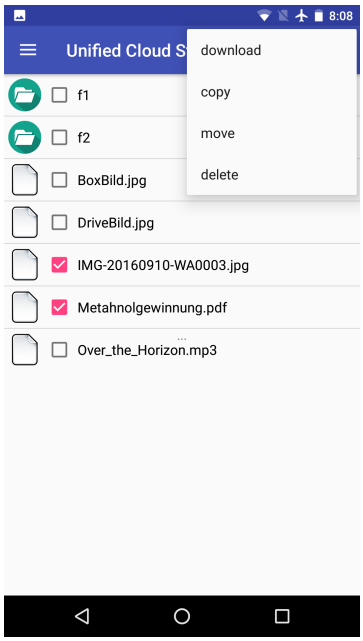

Figure 2.3: Download multiple files

2.5 Network Handling

The user can configure whether he wants UCS to use only WiFi or also mobile data connection. To keep track of the network state, UCS registers a NetworkChangeListener and stores a boolean as SharedPreference if currently network connections are allowed or not:

$$networkAllowed = wifiConnected \vee (mobileConnected \wedge not\_wifiOnly)$$

Before an action using Internet is performed, UCS checks the boolean in the SharedPreference if it is allowed.

2.6 Using a compiled Binary

UCS uses a precompiled binary of *rclone* to connect to the supported cloud storage providers.

Rclone is a command line tool that enables the user to copy, move and sync files from and to various cloud storage providers. It is developed and published by Nick Craig-Wood as freeware for personal and commercial use.[2]

Rclone is programmed in Go Programming Language and therefore can not be imported as library in an Android Project. But since Android is a regular Linux system one can compile *rclone* on Android and then include it as asset in the application.[3]

UCS executes *rclone* the following way:

```
Runtime.getRuntime().exec(String[] commands)
```

This returns a Process object on which one can observe if the command was already executed. To get results from the command, the InputStream and ErrorStream of the Process is read. For interactive subprocesses further commands are written to the OutoutStream of the Process.

Because *rclone* commands which move or copy data take several seconds to execute, these commands are executed in separate threads.

Chapter 3

Conclusion and Future Work

UCS lets the users access their files as if they were stored on one single cloud drive while they are actually distributed over multiple independent cloud storages. Thanks to the caching, browsing through a Unified Folder works smooth and fast (see section 2.2).

The choice of *rclone* as cloud API was necessary since it enabled us to support the most common cloud storage providers without including a bunch of different APIs in the application. However, it makes UCS dependent of *rclone*. That means each time a cloud storage provider changes it's interface, we have to wait for *rclone* to be updated. Then the *rclone* library has to be updated and then UCS can be updated.

3.1 Future Work

It would be easier for the developer if the Java interface for *rclone* would not be part of the source code of UCS. It should be contained in a separate Java library. Like this it would be reusable and could be developed independently from UCS. The implementation of the *rclone* interface contained in the application as it is now is not complete. It only implements the functionality used for the application. A more general implementation packed into a library would not only be useful for the further development of UCS, but also for other applications (e.g., a cloud file browser for desktop usage).

A feature which would be useful is the usage of different cloud storages as drives for a RAID. Unified Folders represent a RAID 0 configuration. It would be nice if users could also define RAID 1 or RAID 5 configurations with different cloud storages. The key question before implementing this is how to define the stripe-size of the RAID. One idea is to use files as stripes, this is the simplest solution to implement, but it will give bad performance with large files. Another approach would be to split files into stripes of predefined size. This is as it is done by conventional file systems. But it would give additional computation effort on the mobile device to split and recompose the files on upload and download. It would also cause the files to be unreadable for any other access than with UCS.

Bibliography

[1] Studer, M.: Android smart cloud storage. Semester project, ETH Zurich (December 2016)

[2] Craig-Wood, N.: Rclone. https://rclone.org/ Accessed: 2017-07-03.

[3] Park, S.: Develop android app using golang. https://de.slideshare.net/SeongJaePark1/hello-androidgo (April 2015) Accessed: 2017-07-03.