snafu design document

Hans Krutzer
Jochem Kooistra
Kash Afzal

May 21, 2013

Contents

1 Introduction

When I first got my smartphone a few months ago, I was very happy with how it integrates and synchronizes with the applications I already use on my computer works. After logging into my Google account, my calendar and contacts were on the phone immediately. After configuring my email accounts, my mailboxes were synchronized, and with Dropbox, I had all my important files in my pocket.

After a few days however, I found out something was still missing. The same problem I have with my desktop and laptop, I now also had with my smartphone: how do I quickly copy something from one device to another? How do I send the image I'm looking at on my computer to my friends on WhatsApp? How do I open a link they sent me on my computer? How do I finish this email on my computer, that I started writing on my smartphone[1]?

It turns out this problem has not been solved. All of these things are doable, but it's not very simple. It takes way too long to open a file on my phone, that I have open on my laptop, which is sitting just 30cm away. We are going to solve this problem with a smartphone application that will copy text, images, and other files from a desktop computer or laptop, to its clipboard or filestorage.

Most people have a smartphone and a computer they use simultaneously. We believe that a lot of these people are not tech-savvy, but still encounter the problem we described. They, however, still managed to synchronize their calendar and email, because Google spent a lot of time on making this simple and easy. To reach these less tech-savvy people, we wish our app to be as simple and easy-to-use as possible. This means no sea of menu's to wade through, and more importantly, little to no configuration.

2 Justification

Of the aforementioned problems, there's not really one of them that has not been solved in some kind of way. You can put images, textfiles and other files in a filehosting service like Google Drive or Dropbox, or transfer them with Bluetooth or through a USB cable. While being the fastest solution, a USB connection requires a cable. Bluetooth connections are not very fast, and often draw a lot of power.

Filehosters like Dropbox and Google Drive require a third-party server to store your data. This means you first have to send your data there, before you can download it again on your mobile device, which is right next to your laptop. Finally, none of these solutions solve the text-sharing problem very well: they all require a file to put your text in, which you then have to open, and select all the text in it and put it in your clipboard.

There are apps will get the text contents of your clipboard from your computer onto your phone, but they are not very easy to use, and require you to enter the IP of your computer. They also lack the ability to transfer anything that isn't text.

[1] The "Drafts" folder used by the Android email client, and many other email clients, turns out to be local, and doesn't save emails using IMAP

3 Requirements

3.1 Functional requirements

We've formalised our functional requirements in a set of rules:

- The user must be able to perform all of the app's abilities listed below in three clicks/taps or less. All actions will be performed from the mobile device, not the computer.
- The user must be able to transfer text from the clipboard of a computer, to the mobile device's clipboard, and vice versa.
- The user must be able to transfer any a file any kind from a computer's clipboard to the mobile device's storage, and vice versa.
- The user must be able to share an image to another application (using Android's share function) on the mobile device, if the app has just transferred one from the computer.
- The user must be able to transfer multiple images at once, selected in the Android gallery, to a computer.
- The user must be able to transfer multiple images at once, on the clipboard of the computer, to a mobile device.
- The mobile application must be able to discover computers on the network, with the only user interaction being the user telling the application to discover computers.
- The application will be an Android widget to be placed on a user's dashboard, for easy access.
- The application will work from the widget, without having to 'launch' the application.

3.2 Non-functional requirements

- The primary (and likely only) method of transferring data will be the (wireless) local network.
- The application can not have a significant battery impact (e.g. more than 1% battery usage after a day of normal use of the phone and the application).
- Transferring files from the computer to the mobile device must be as fast, or at least not noticeably slower, than downloading the file from a local web server running on the computer.

4 Design

4.1 Global design

The application has to live on both the mobile device and the user's computer. Both will however have similar components, these being:

1. The component that loads the data into the application from filestorage or the clipboard
2. The component that transfers the data over the network
3. The discovery-component (the part that makes your computers magically appear in the device's list of computers)

We however don't believe these components can share any actual code, because of the difference in APIs between Android and the 'regular' Java API. Unique to both devices will their GUI, a tray icon and widget for the computer and the Android device respectively, and unique to the Android device will be the Android inter-application sharing option.

4.2 Design in detail

4.2.1 Device discovery

The device discovery will work through UDP messages broadcasted over the local network. The computer will continually broadcast messages in short interval. The device will only listen, or broadcast a single message upon use of the application, after which it will wait for a reply, to save battery. The Android guidelines strongly recommend making sure applications use as little network connectivity as possible, and we can therefore not continuously broadcast, to meet our battery life requirement.

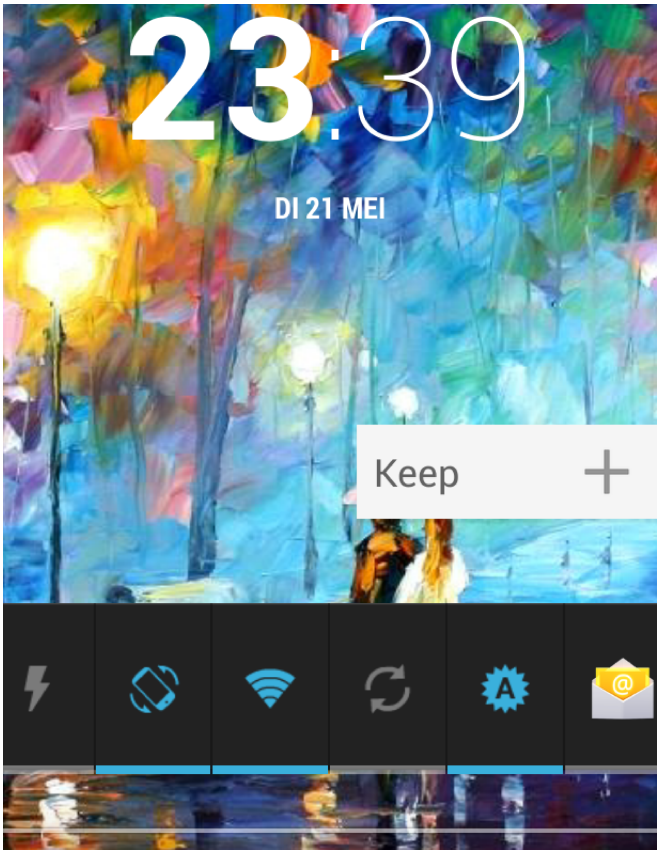
4.2.2 Data transfer

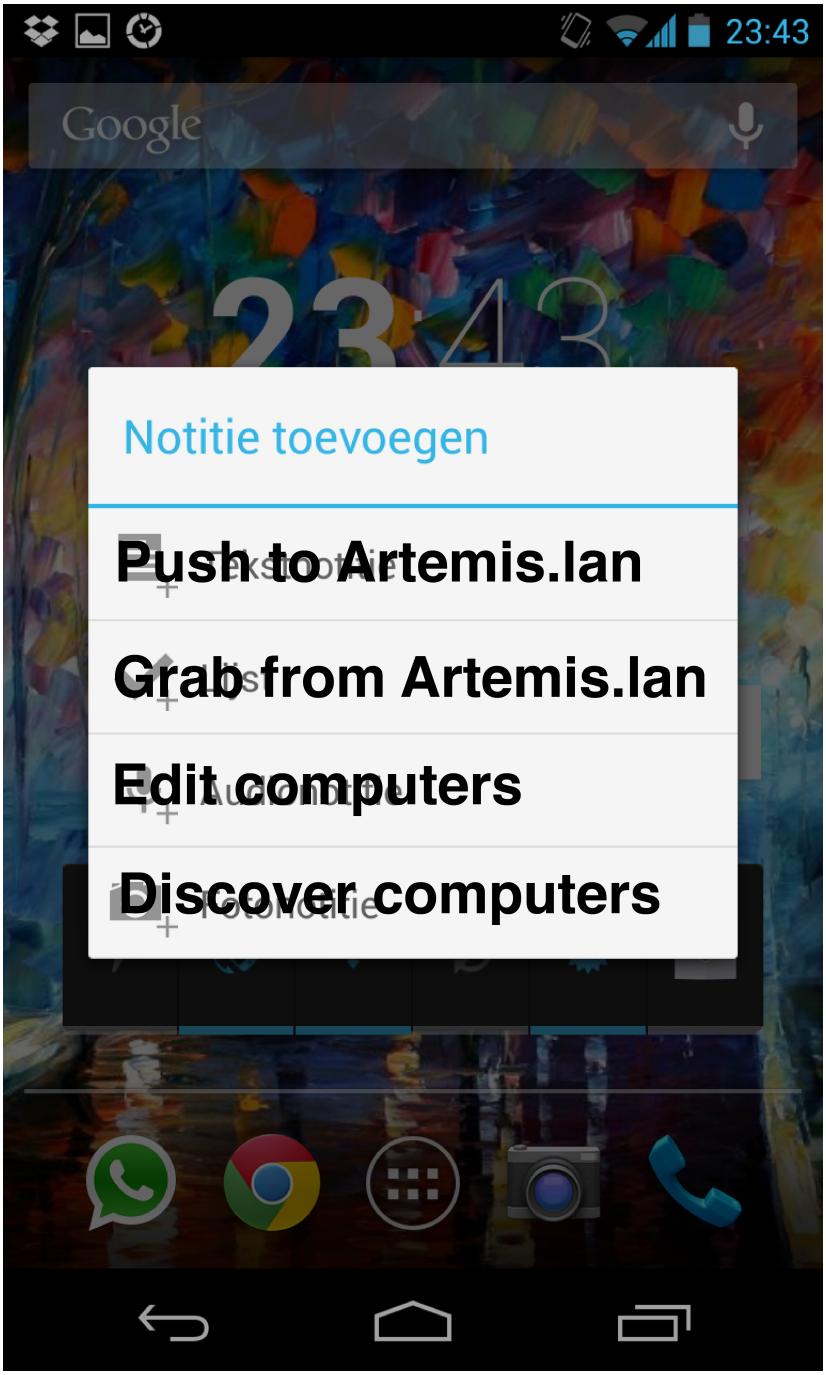
The data will not use a custom protocol. Instead, it will use the most commonly used protocol in Android and probably on the internet: HTTP. HTTP is a tried and tested protocol, and provides many, if not all, of the options we need for data transfer. It can also be secured with HTTPS if necessary. HTTP has these important features built-in: statelessness, data compression, and MIME-types. MIME-types will enable us to have a single URL on the server, with which the application on the mobile device can discover the file type. Furthermore, because HTTP is so common, there are many great libraries for both the client- and serverside.

HTTP requires, of course, a server, which will live on the user's computer. The device will use one URL to get files or text, and one URL to send files or text to, using HTTP GET and HTTP POST respectively.

4.3 User interface design

The user interface will model the Google Keep application's widget. A widget can be placed anywhere on the dashboard. After placing the widget on the dashboard, the user can tap it, and a menu will appear, allowing user to either "grab" from $computer_name, or "push" to it. At the bottom of the menu, there will be an option to discover devices, and remove items from the list.




23:43
Google
Notitie toevoegen
Push to Artemis.lan
Grab from Artemis.lan
Edit computers
Discover computers

5 Planning

Deliverable	Date
First version serverside program	May 24
First version clientside app (copies text, discovers devices)	May 26
Second version app (copies text and images)	May 29
Third version app (copies any file)	May 31