FileCloud - Cloud Filesystem based Processing Offloading

Tiago Amaral Almeida Lima

Instituto Superior Técnico, Universidade Técnica de Lisboa
Distributed Systems Groups, INESC-ID Lisboa
Lisboa, Portugal
tiago.lima@ist.utl.pt

Abstract

Nowadays, mobile devices, such as smartphones or tablets, have the ability to be connected to the Internet anywhere and anytime with very reasonable speeds. With this feature, users can access the cloud services wherever they are, whenever they want. Furthermore, mobile devices are also becoming very powerful, so that users start doing there demanding tasks, that some years ago only desktop or laptop computers could perform. Although battery and display technologies are improving, battery capacity and display size are still limiting factors for the user to use mobile devices efficiently.

Processing offloading (in order to user remote, but nearby desktop computers) is a viable solution to overcome battery and display limitations, increasing energy efficiency and usability. FileCloud unifies mobile devices with nearby idle desktop computers in the neighborhood, allowing the processing of files on the device with better characteristics. These files can be stored in a Cloud based FileSystem, thus being available to both devices, on can be transferred directly from the mobile device to the desktop computers. With FileCloud, usability issues are reduced since files can be accessed on desktop computers, thus the energy consumption is reduced on the mobile devices since they become idle after the offload of the task.

FileCloud is completely transparent to the user (w.r.t. the discovery and selection of the best host and application to be launched), allows gains in the energy consumption on most tasks (e.g. video visualization) and reduces the time spent by the user (in task such as image editing).

Keywords: Cloud, Offload Computing, Energy Saving, Computing power, Mobile devices

1. Introduction

Nowadays, mobile devices, like smartphones and tablets, have the ability of being connected to the Internet anywhere and anytime with very reasonable speeds by using LTE or 3G network, or even by Wi-Fi connection. With this feature, the users can access the cloud services whenever they want. This is useful because, in some way, cloud resources can be used to reduce hardware limitations of mobile devices [1].

Over the past few years, several cloud services were developed for many different purposes. Storage cloud services are the most used by regular users, for example, Dropbox[1] or Google Drive[2]. With these services, the users can store their files in the cloud and access them anywhere, every time they are connected to the Internet.

Mobile devices are also becoming very powerful. So powerful that they can perform some demanding tasks that some years ago only desktop or laptop computers could do, for example, image and video editing. But running these tasks introduces some problems to the user. The battery technology has improved over the past years, but processors' power has also increased and due to this, the battery continues to drain fast [2]. To address this problem, many systems implemented the concept of Computation Offloading. This concept consists in assigning part, or even all of the workload onto devices in the neighborhood or in the cloud, freeing the device of this work and saving battery power.

But there is one specific problem that has never been addressed by these kinds of systems: how to take advantage of nearby idle computers to complement and augment mobile devices. It would be very useful for the users to take advantage of desktop applications, which provides a much richer user experience. For example, currently, the applications available for video or word document editing aren't even comparable to the desktop ones. They lack many features that the user may want to use.

Nowadays, users working on a smartphone or tablet can't, in an easy and straightforward manner, take advantage of nearby computers to execute tasks that are slow or cumbersome to execute on the mobile device (such as image

[1]Dropbox, https://www.dropbox.com/
[2]Google Drive, https://drive.google.com

edition of word processing: user must to walk to his desktop computer or laptop, search again for the file and open the appropriate application. All this process is a burden for the user.

This document presents FileCloud, a system that unifies the following resources: mobile devices, cloud storage and nearby desktop PCs. With FileCloud, the user can take advantage of idle desktop computer to execute some actions that are not suited to run on a mobile device. Also, the better processor on the desktop computer helps a lot doing these tasks, because the majority of the mobile devices still have very weak processors, and lack a large display and physical keyboard.

With FileCloud, the user opens a file on the mobile device, but transparently FileCloud searches for a suitable device to open this file, and transfers the processing (or visualization) to such computer. When the user reaches the computer, the application was already launched and ready for the user to do what they want. FileCloud also allows the user to take advantage of the better user interface of the desktop application, and at the same time take advantage of a bigger display.

FileCloud implements a client-server architecture: a client installed on a mobile device and servers on neighbor desktop computers. On the client, after the user selects a file, stored in the cloud, FileCloud scans the neighborhood for available computers and communicates, in order to know their specifications, together with the actions that can be made for that file type, depending on the installed applications. When the action is selected, the client sends the information needed for the server, in order to automatically select the appropriate application and launch it for the user. In sections 3.1 and 3.4 the architecture and execution flow are explained in more detail.

Section 2 addresses the current state of the art, describing some systems currently available related to FileCloud. Section 3 describes FileCloud in more detail. It presents the architecture, execution flow and its implementation. On Section 4 the evaluation results are presented and discussed. Then, on Section 5 is given some ideas for the future work and, finally, on Section 6 is presented the conclusions of this project.

2. Related Work

In this section is described some system that are related to FileCloud. The systems and architectures here described fall in on of the following categories: processing offloading, Mobile Cloud, UI mobility and Home Entertainment Systems. Although the systems try to solve the same problem as FileCloud, none of them use the existing Cloud file systems as a data communication to layer and use neighbour computer to provide a better user experience.

2.1. Processing Offloading

Processing Offloading is an efficient mechanism to take advantage of remote computers in order to complement the capabilities of a device and optimize resource consumption. Related to mobile computing and environments, two examples of systems that empower Processing offloading are SPADE and Cuckoo. SPADE [3] is a scheduler for parallel and distributed execution. SPADE can accelerate the manipulation of a batch of files using idle remote computer through jobs. These jobs that run on remote computers are composed of simpler tasks. Using a simple user interface, a user with minimum computer knowledge can define the jobs, what is usable and how they can possibly be divided. Then SPADE transparently selects a computer on the LAN and automatically uploads the files to the remote computer, executes the application defined by the user and downloads the resulting files back to the mobile device. The Cuckoo framework [4] simplifies the development of smartphone applications that benefit from computation offloading and provides a dynamic runtime system that decides, at runtime, if a part of an application is going to be executed locally or remotely. This framework can be used to easily write and efficiently run applications that can offload computation. Cuckoo offers a programming model that is prepared for mobile environments. This programming model offers the developers an interface of the system, so it should be easy to use and understand. It must support both local and remote execution since when the mobile devices are not connected to the Internet, the cloud resources are unreachable. It must also allow the remote method implementation to be different from the local implementation. For example, the remote method could be parallelized in order to get full performance from the remote resources.

These systems have different goals from FileCloud: SPADE is to be used when executing batches of tasks in the mobile device and Cuckoo provides a programming model and environment to the development of applications that benefit from computation offloading. Interactive applications can take advantage of SPADE, while, the use of preexisting applications is not possible with Cuckoo.

2.2. Mobile Cloud

CloneCloud [5] is a system that automatically transforms mobile applications to benefit from the cloud, by optimizing the execution time and the energy used in the mobile device. CloneCloud transforms a monolithic application into a distributed one, by rewriting an unmodified application executable. While the modified executable is running, at automatically chosen points, individual threads migrate from the mobile device to a device clone in a cloud. The remaining functionality on the mobile device keeps executing, but blocks if it attempts to access migrated state. The migrated thread executes on the clone, possibly accessing native features of the hosting platform such as the fast CPU, network, hardware accelerators, storage, etc. A mathematical optimizer chooses migration points that optimize objective (such as total execution time or mobile-device energy consumption) given the application and the cost model. Finally, the run-time system chooses what partition to use.

By offloading the right portion of the applications execution onto the device clones that are operating in the cloud, CloneCloud can *boost* these applications since they can take advantage, for example, of a better processing power of the remote computer reducing the execution time. But there are other "variables" that are analysed before starting partitioning the execution, for example, if the execution on the clone is more reliable or more secure it may be worth it to take advantage of that. The partitioning of the application depends on the expected workload, such as CPU speed or network performance. It may be more fine-grained or not depending on these factors. This means that the same application run on different times may be partitioned differently.

Jupiter [6] is a system to augment the capabilities of smartphones transparently with the support of cloud computing. One or more cloud servers can be employed as the extension of the storage and computing capabilities of smartphones. The cloud provides storage of applications and user data, and also provides virtual machines to execute large applications that cannot be on smartphones due to resource constraints. Jupiter aims to achieve three primary goals. There are two ways to load an application. The first one is choosing an application in the Android Launcher. Jupiter maintains a list of applications, which are stored in the AppLib and the required application is loaded through a click. The second way is "on-demand", in which an application is loaded through another, for example, when a user wants to view an attachment file in an email client, Jupiter loads the corresponding viewer automatically from the AppLib based on the file. Desktop application can also be loaded similarly through the above two ways.

CloneCloud takes advantage of the resources in the remote computer, but partitioning applications is not a good solution for our execution environment. The main feature of FileCloud is to take advantage of neighbour computers' larger screens and better processors in the use of applications with a graphical user interface. Partitioning a user interface is very difficult, but does not free the mobile device from complex computations. Jupiter introduces some concepts that are wanted for this project, such as executing desktop application or loading an application based on the file, but there are some differences. Jupiter does not take advantage of bigger screens on the neighbour when loading a desktop application. A bigger screen is very useful when editing an image or a video.

2.3. UI mobility

With respect to mobility, one possible way is to separate UI from computation, allowing their execution on different devices. VNC [7] is an ultra-thin client system based on a simple platform independent display protocol that frees the user of the burden of carrying around the hardware all the time. In VNC, applications run on a centralized server, and the client only presents application UI, interacting in a client-server fashion. This allows the development of simple and efficient displays and their mobility.

Distributed User Interfaces (DUIs) [8] become a new field of research and development in Human-Computer Interaction (HCI). DUIs are those interfaces whose different parts can be distributed in time and space on different monitors, screens, and computing platforms. In order to understand and classify existing approaches for DUIs, as well as to identify under-explored situations, a reference model was introduced. This reference model examines DUIs according to the 4C [9] dimensions: i) Computation (What is distributed), ii) Communication (When is it distributed), iii) Coordination (Who is distributed) and iv) Configuration (From where and to where is the distribution operated)

In order to obtain a flexible user interface distribution across multiple devices a model-based user interface language was extended in order to address the specification of distribution at various user interface granularities [10]. The language was extended in order to be able to specify DUIs and was also designed a solution to generate implementations of such DUIs. These DUIs can dynamically change how the user interface elements are distributed according to user requests or other events.

VNC does not solve the problem that FileCloud aims to solve, because the goal of FileCloud is not to access the mobile device's entire environment on remote computers. The goal is to take advantage of the bigger display and of the better user interface of desktop applications for certain types of tasks.

DUIs do not support the offloading of computation: optimizes user interaction, but power consumption remains high. FileCloud differentiates from DUI's because the goal is not to distribute the user interface of the mobile application. The goal is to use the user interface of an application installed on a neighbour desktop computer.

2.4. Home Entertainment Systems

Home Entertainment Systems are usually composed of a client server architecture where one system stores the multimedia contents (Media Server) and the other presents them to the user (Media center).

Media Centers can be implemented off the shelf Hadware sucha as HTPC running extended versions of commodity OS, such as XBMC (http://xbmc.org/) or Windows Media Center (http://windows.microsoft.com/en-US/windows/products/windows-media-center/). These system, by means of a simple UI, allow browsing and accessing media available on remote sources.

The access to that media can be accomplished by means of (for instance) Digital Living Network Alliance (DLNA, http://www.dlna.org/). DNLA allows the access to media and provides mechanism to identify and match the characteristics of the media being played and the system playing it. Universal Plug and Play (UPnP) [11] allows the discovery of the resources and defines the type of device that DLNA supports and the mechanisms for accessing media over a network.

DNLA can now be executed on mobile devices allowing the files stored there to be access from Media Center. This way of functioning is close to what FileCloud aims, but the media control and access is started on the Media Center, not on the mobile device.

3. FileCloud

FileCloud tries to unify the following three resources: mobile devices, use of idle desktop computers in the neighborhood and the access to files stored in the cloud. Systems like Dropbox or Google Drive allows the access to the user's personal files any-time, anywhere. With FileCloud the user can take advantage of idle desktop computer to execute some actions that are not suited to run on a mobile device for several reasons, such as the excessive battery drain. Also, the better processor on the desktop computer helps a lot, because the majority of the mobile devices still have very weak processors.

Services to offload tasks [12] already exist, but these services are too complex to configure by the average mobile device user. FileCloud aims to optimize this through a transparent execution on a remote computer and automatically providing the following usual tasks: i) File open, ii) File Edit, and iii) File Print.

3.1. Architecture

In order for the system to run, it needs two daemons: one installed on the mobile device and another on at least one PC in the neighborhood. As illustrated in Figure 1, the architecture of both daemons are very similar. Both have three layers: the User Interfaces which the user directly interacts with, the Core and a layer with the communication modules. This architecture is described in more detail in the following sections.

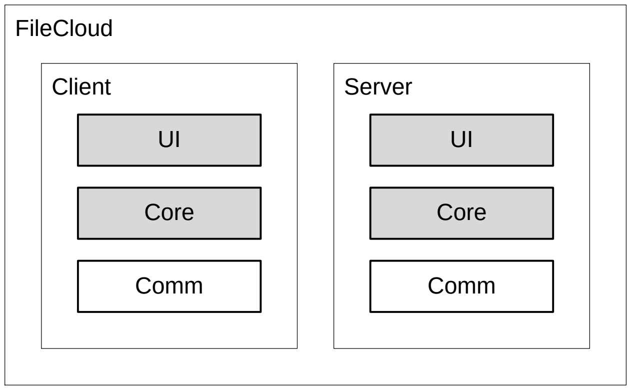

Figure 1. Architecture overview.

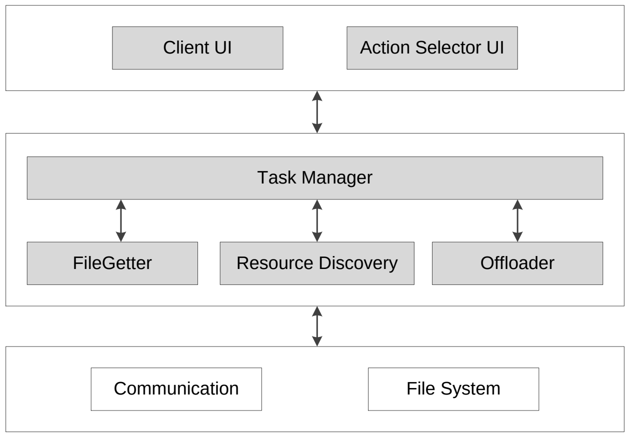

Figure 2. Client architecture. In gray are the modules implemented during the project and in white are the already existing technologies, which FileCloud will use.

3.2 Client Architecture

As aforementioned, the client is composed by three layers. The upper layer is the one the user directly interacts. This layer is implemented by two modules. The **Client UI** is the main interface of the client. It is responsible to show all the file information of the user's Cloud-based File System account and is where the user can select the wanted file. We also have the **Action Selector UI**. This interface is where the user can select the action he wants to perform on the file selected.

The middle layer is the core of the client. This is where all the information is processed in order to offload the task. The **Task Manager** module is responsible to coordinate the execution flow, by calling the other modules whenever they are needed. The **FileGetter** module communicates with the cloud-based File System and gets all the file information needed from it. The **Resource Discovery** module searches for available PCs and communicates with them in order to obtain information about them. The **Offloader** is the module responsible to assemble all the information the user inserted and start the offload of the task.

Finally, we have the bottom layer. This layer is implemented by two modules: the **Communication** that offers the mechanisms to communicate with the PCs and the **File System**, which is where all the file meta-data is stored. These modules are implemented by existing technologies that will be used by FileCloud.

3.3 Server Architecture

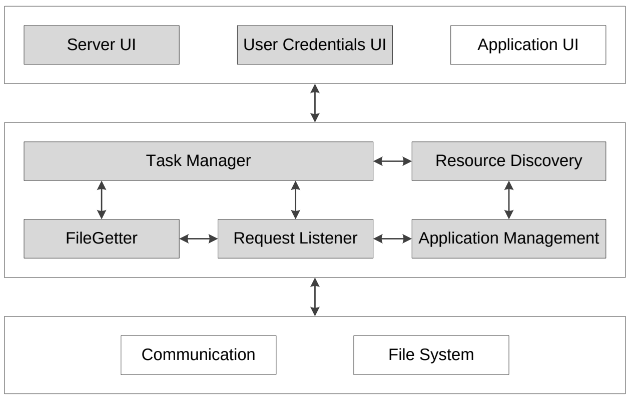

Figure 3. Server architecture. In gray are the modules implemented during the project and in white are the already existing technologies, which FileCloud will use.

Similarly to the client, the server is also composed by three layers. The upper layer is implemented by three modules. The **Server UI** is the main interface of the server, where the user can view all the important information about the state of the server. The **Application UI** is not explicitly implemented since it is the interface of the chosen application to do the task. The **User Credentials UI** is the interface where the user can input his File System's credentials so FileCloud can have access to it.

The middle layer is the server's core and is implemented by five modules. The **Task Manager** is responsible for starting/stopping the server, as well as storing all relevant information. The **FileGetter** is responsible to communicate with the FileSystem of the user. The **Resource Discovery** is the complementary module to the one in the client. It is responsible to let the clients discover the PC. The **Request Listener** module listens to requests to start a task. Finally, the **Application Management** module is responsible to search the installed applications and to know which application is the proper one for the file type and action specified.

Finally, the bottom layer is similar to the one in the client. We have a **Communication** and a **File System** modules, which are existing technologies used by FileCloud.

3.4. Execution Flow

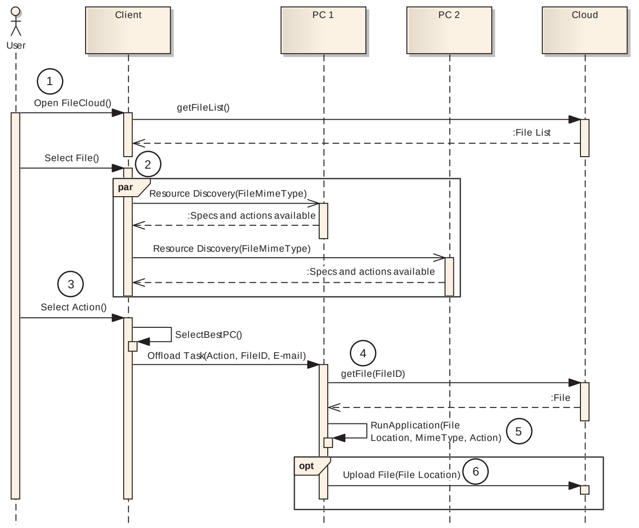
User
Client
PC 1
PC 2
Cloud
1
Open FileCloud()
getFileList()
:File List
Select File()
2
par
Resource Discovery(FileMimeType)
:Specs and actions available
Resource Discovery(FileMimeType)
:Specs and actions available
3
Select Action()
SelectBestPC()
Offload Task(Action, FileID, E-mail)
4
getFile(FileID)
:File
RunApplication(File Location, MimeType, Action)
5
opt
Upload File(File Location)
6

Figure 4. FileCloud execution flow.

As shown in Figure 4, in order for the system to function properly, a FileCloud client must be running in the mobile device and a FileCloud server running in at least one neighbor PC. By the time the client starts, it connects to the cloud-based file system and retrieves the meta-data of the user's file list (step 1). Selecting one of the files, FileCloud starts to search for available PCs in the neighborhood by sending a LAN broadcast message (step 2). Each PC receiving this message, searches for what actions can be performed on that type of file and returns it together with the PC's specifications. After knowing all the available PCs, they are sorted in order to select the best one available for the task. Now a list of the available actions that can be made to the file is shown on the screen for the user to select. Selecting one of these actions, FileCloud connects to the best PC and offloads the task to it (step 3). Then the meta-data is retrieved from cloud file systems and, if the file is not present on the file system's sync folder, the real file is downloaded (step 4). Finally, the proper application is executed for the user to use it (step 5). When the application is closed, the file is transparently uploaded to the cloud (step 6).

3.5. Implementation

Two different programming languages were used in the development of FileCloud. On the server-side it was used Microsoft's .NET C# and and the client was written for the Android platform.

Although the tests were executed on the Windows platform, the Mono (www.mono-project.com/) .Net execution environment allows the execution of FileCloud on other platforms, such as Linux distributions. The underlying Cloud File System used was Google Drive. Regarding the server, a minimalist user interface was developed. The interface contains a TextBox, which shows the log of the current session a button, where the user can press to start or stop the server to, respectively, allow or deny remote accesses and, finally, a ListBox with the list of users logged in.

When starting the server, two different threads are created. One thread is responsible for the Resource Discovery and another for listening the user's requests. Each thread uses their TcpListener and is listening on different ports, being the Resource Discovery on port 3000 and the other on port 3001. The TcpListener class provides simple methods that listen for and accept incoming connection requests in blocking synchronous mode.

For the application management, it was build a small database with predetermined application for each combination of MIME type and action. The File Getter class uses the Google Drive API to get the files metadata from the cloud. There was also developed a method to get the real file, in case there isn't a copy on the Drive's Sync folder.

The demonstration client was developed in Java programming language, to execute on Android devices. The access to the cloud files uses Google Drive's API to retrieve the metadata of the files. The resource discovery sends a message with the MIME type of the file. This socket is received by the TcpListener on the server and then, through the same socket is received the response. In this response includes the computer specifications and the actions available for the type of file wanted. After receiving all the responses from the computers, it is created a list of these computers and they are sorted by a rating value.

After selecting the action, the best computer is contacted, with the File ID, the action to be performed and the user's email account the action. The server, receiving this information, checks through the email, if already has access to the user's Drive. If not, the user has to authenticate before the task starts. The server will save a proxy to the user's Drive, until he logs out or until the server is shutdown, removing the burden from the user of having to authenticate every time a task is offloaded. Finally, the proper application is searched and automatically launched for the user.

3.6. Resource discovery

The evaluation and discovery of available resources weren't the focus of this project, but in order to implement a functional prototype it was developed a simple resource discovery mechanism, that is described in this section. Systems, such as Condor's ClassAd [13] or STARC [14], could be integrated in FileCloud providing a more advanced system to discover host and evaluate the available resources. Furthermore, locations should be taken into account when evaluating resources.

Host discovery is performed by means of a LAN broadcast, while the evaluation of the found hosts is made using a technique similar to the one presented in STARC. With a suitable requirement (display size, memory and processor characteristics) STARC sorts the available computers for FileCloud to select the best one.

3.7. Application Management

The AppManagement module verifies the existence if the applications installed in the operating system in order to return the available actions for a specific type of file during the Resource Discovery. After the selection of the action it is responsible to run the appropriate application.

FileCloud is able to use any type of multimedia (image, video and music), text files and the most common application MIME types for documents (Word, Excel, PowerPoint and PDF). FileCloud allows opening (for visualization) every type of file, and editing to a sub-set.

FileCloud is also able to use Google Docs (executed in a web browser) to view of edit files if a suitable application is not installed on the desktop computer.

4. Evaluation

In order to evaluate FileCloud, some test where performed: battery energy consumption when viewing videos (Video Visualization), usability when editing an image (Image Editing) and writing text (Writing), and finally, the communication and resource evaluation overhead (Overheads).

4.1. Video Visualization

The goal of these tests is to evaluate how energy consumption varies when viewing videos (of different sizes) on the mobile device and on a remote computer (by means of FileCloud). In this test three samples with different sizes and lengths where used. The properties of each sample are shown in Table 1.

Five different experiments were performed:

i) local viewing with the screen on maximum brightness,

	Sample 1	Sample 2	Sample 3
Length	00:08:52	00:04:49	00:02:15
Resolution	640x360	640x360	640x360
Size	44.89MB	18.41MB	10.45MB

Table 1. Properties of the video samples.

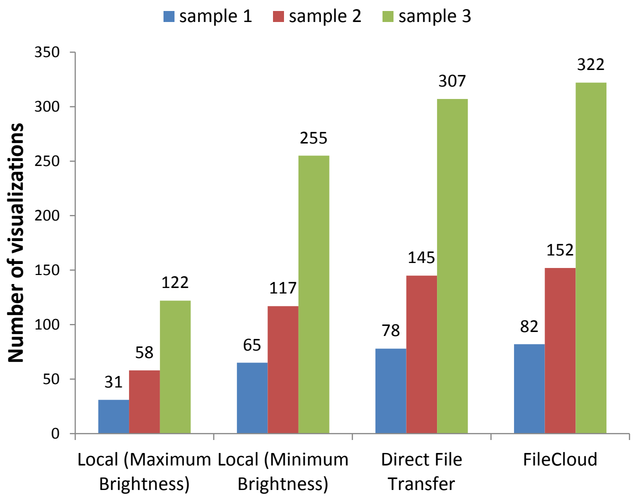

Figure 5. Video visualization results: Number of visualizations.

ii) local viewing with the screen on minimum brightness,

iii) remote viewing with direct file transfer, and

iv) remote viewing with FileCloud.

During tests i) and ii) wireless network was off, while on tests iii) and iv) the screen was naturally dimmed when the tablet got inactive. Each test consisted on continuous and repeated viewing of a video file until battery drained completely. The base test consisted in obtaining the maximum battery life of the mobile device, with the screen always ON.

The number of total video visualizations and battery duration is presented in Figure 5.

As we can see in Figure 5, comparing the results of the same sample, FileCloud allows the most number of visualizations. With respect to power consumption File Cloud is close to the minimal power consumption possible: the red horizontal line on Figure 6 represents for how long the tablet is able to be on with Wi-Fi off and the screen dimmed to the minimum. File cloud does not add much power consumption overhead to that base consumption.

It is evident that when using a remote computer to view the videos the battery drain on the mobile device is lower. This happens because the tablet display is dimmed, when the user is viewing the video on the remote computer. The cost to have the wireless ON is lower than the gains from

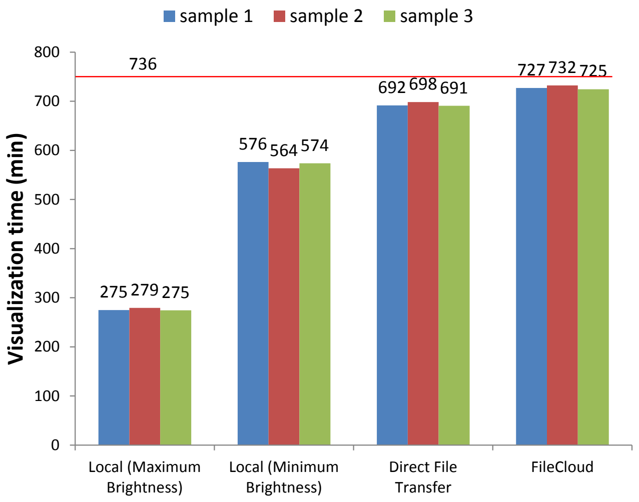

Figure 6. Video visualization results: Battery time.

turning off the display. It is also possible to observe that the use of a cloud file systems reduces network usage and as such power consumption.

4.2. Image Editing

Another advantage of FileCloud is the use of bigger screen and better data input methods (mouse and full size keyboard). This test evaluates if productivity gains (from using better data input methods) are enough to overcome the overhead from launching the file from FileCloud and from the communication between the mobile device and the remote compute.

It was made two tests of editing images with different degrees of difficulty: a very simple task of inverting an image's colors and a cropping test since it requires more skill from the user. With these tests we can know with more accuracy, if compensates to use FileCloud, even in very simple tasks.

4.2.1 Invert colors

As aforementioned, this test consisted on a simple image manipulation (invert colors).

The workflows to perform the test were this following:

- **Tablet:** select on file on Gallery, select Edit option, choose Photo Studio application, invert colors, save the image exit.
- **PC using FileCloud:** use FileCloud to select the image, select the operation, wait for the desktop application to open (MS Paint), invert colors on the laptop, save the image and exit.

	Number of editions	Total time spent
Tablet	5	71.98s
FileCloud	5	56.07s

Table 2. Results of Invert colors test.

As we can see in Table 2, the time to perform 5 image conversions in a row on the mobile device is more than 1 minute (1m11s), while using FileCloud and a remote computer it only takes slightly less than 1 minute (56 seconds) to perform the same transformations.

In average, the user takes 11s to do the task the on the PC using FileCloud, while on the tablet it takes approximately 14s. With these results, we can conclude that, in the same period of time, even tasks that are very simple to perform on a tablet, gain from using a remote computer.

4.2.2 Crop

In this test, three users were asked to crop in image, the best they could in each of the workflows of this test. In this test, 3 users were asked to perform it, because unlike the previous test, this one requires some skill to crop the image, while the previous one, it was only needed to press some buttons.

The workflows to perform this test were the following:

- **Tablet:** select on file on Gallery, select Edit option, choose Photo Studio application, crop image, save the image exit.
- **PC using FileCloud:** use FileCloud to select the image, select the operation, wait for the desktop application to open (MS Paint), crop the on the laptop using the mouse, save the image and exit.

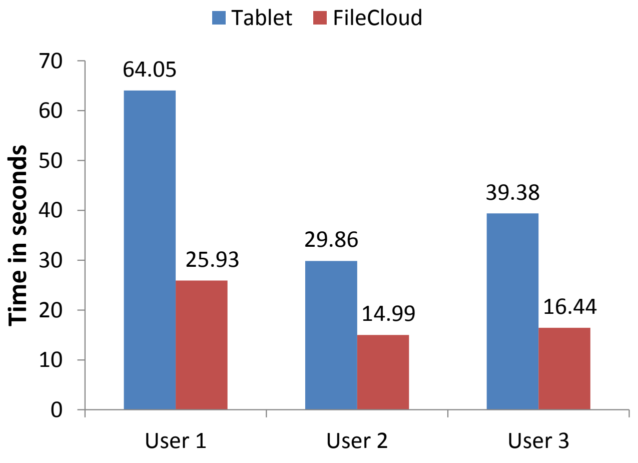

Figure 7. Crop test results: Time to perform task.

As Figure 7 shows, the results were different for each user. Some users are more comfortable doing the task on the tablet than others, for example, User 1 took more than 1 minute (1m4s) to crop the image while User 2 took less than half of the time (29.86s).

We can also conclude that even the more skillful users on the tablet, still gains by using FileCloud to perform the same task on the PC with a mouse. All users took approximately half the time to perform such task.

4.3. Writing

In this test, the same three users were asked to copy text as much as they could during 5 minutes on a document editor installed on the tablet, and then open a Word document using FileCloud, and again write as much as they could for the same period of time.

The workflows to perform this test were the following:

- **Tablet:** Open new document on Toshiba's Write application, copy as much words as they could, from the abstract of a selected paper, in 5 minutes, save the document exit.
- **PC using FileCloud:** use FileCloud to select a Word document file, select the operation, wait for the desktop application to open (MS Word), copy as much words as they could, from the abstract of a selected paper, in 5 minutes, save the document and exit.

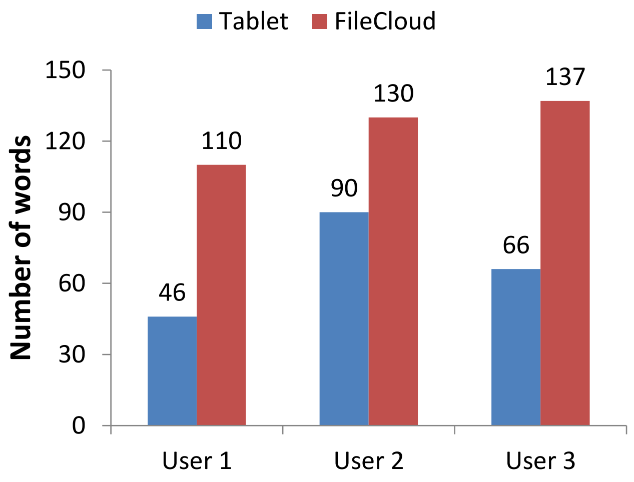

Figure 8. Writing test results: Number of words in 5 minutes.

As shown in Figure 8, the number of words a user writes are higher using FileCloud for writing on a PC with a physical keyboard. The results on the tablet have a bigger difference between each other than the results on the PC, due to the fact that some users don't have much experience with virtual keyboards. Because of this inexperience, the users did more mistakes while writing and they took more time correcting the mistaken words. The number of mistakes could have been even more significant if instead of using a tablet with a 10.1" screen, the user used, for example, a smartphone with a 4" screen. Even the user that performed better on the tablet still had some gains using the PC (about 1.5 times more words). As for the other users they had a much better performance on the PC in comparison with the tablet test, since they wrote about 2 times more words.

4.4. Overheads

The overhead test has the goal measure the time it takes to perform resource evaluation and communication. In lengthy tasks the energy consumption is more relevant, but on small tasks the time needed to start the task on a remote computer is relevant. The test was made using 1 available PC in the neighborhood.

The total overhead cost of offloading a task is less than 1 second (0.875s). This value include the resource discovery (a broadcast), the evaluation of the available resources on the remote computers, retrieving of the file identification and the actual initiation of task. This small value is orders of magnitude smaller than any task that users usually performs on a mobile device, allowing us to conclude that this overhead will be unnoticed by the user.

5. Future Work

In this section it's pointed some ideas do to in the future in order to continue this project.

Security issues and discovery mechanisms were not handled in this initial work but are planned to be incorporated. It is necessary to guarantee the privacy of the data handled in the desktop computer, preserve the integrity of the desktop computer against any malicious attack from the user. To accomplish this, virtual machines, such as, sandboxes, could be integrated in FileCloud, this way, protecting the system. The message exchanged between the mobile devices should also be secured, with an cryptographic algorithm, for example, Public Key cryptography (also known as Asymmetric cryptography). This way, the messages exchanged are secured against possible attackers.

As for the resource discovery module, some work should be done in order to improve the discovery and selection of the PCs available on the neighborhood. For example, as illustrated in Figure 9, the location of the PCs should be take into account since the user don't want FileCloud to select a PC on the next room while there are other PCs available for the task in his room.

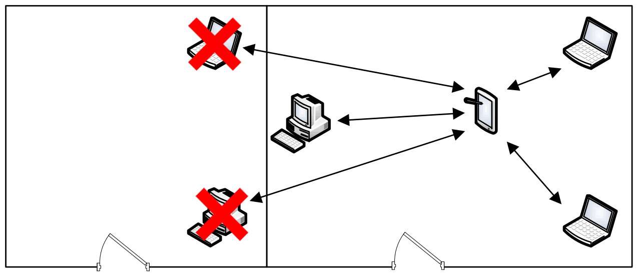

Figure 9. Future work on Resource Discovery module: PC's location.

Furthermore, more exhaustive tests with more users should be made in order to evaluate the system, though the current results are indicative of the gains that can be obtained.

The use of FileCloud in public places, for example kiosks, can also be studied. Currently, kiosks normally provide computers with access to a web browser, not having available applications for the users to perform some kind of tasks, such as image editing. Though, having FileCloud installed on these computers, the users could have, at least, access to the Google Docs editors. But more work could be done in this field. With the use of cloud applications, we could use the web browser to run these applications, providing the user more options of performs their tasks in these public places.

6. Conclusions

FileCloud aims to optimize the relationship between mobile devices and desktop computers in an automatic way, allowing the user to offload tasks to idle desktop computer in the neighborhood. FileCloud automatically chooses the most appropriate application to launch, freeing the user of the burden of having to configure a lot of stuff before offloading the tasks, making it very simple and straightforward for him/her.

FileCloud integrates cloud and mobile computing in a manner not yet experimented. While most systems resource to the cloud to extend mobile devices capabilities, FileCloud extends mobile devices with the help of nearby devices, but uses the cloud as the data communication subtract.

In FileCloud, the offloading of these tasks to a PC allows the user to take advantage of better resources (screen size, keyboard and mouse, better CPU) and reduce mobile device power consumption. These resources, together with other peripherals of the PC, can provide the user an environment that significantly improves the productivity during the task.

The FileCloud now runs on Android mobile devices and

Windows desktop computers and takes advantage of Google Drive, but clients and servers for different combination are possible to develop. FileCloud also allows the use of Cloud based desktop applications (such as Google Docs), not requiring the previous installation of the needed applications. If no suitable application is installed on the remote computer, the FileCloud server is capable of opening a Web Browser, and redirects it to the on-line application capable of showing or editing the file.

Experimental results prove that the offloading of tasks to remote computers reduces power consumption, but also increases productivity. While the task executes on the neighbor PC, the mobile device becomes idle and reduces power consumption. Productivity comes from the use of full featured applications (desktop applications as opposed of its mobile counterparts) and from the use of better data input methods (physical keyboard and mouse).

References

[1] Balan, R., Flinn, J., Satyanarayanan, M., Sinnamohideen, S., Yang, H.I.: The case for cyber foraging. In: Proceedings of the 10th workshop on ACM SIGOPS European workshop. EW 10, New York, NY, USA, ACM (2002) 87–92

[2] Li, Z., Wang, C., Xu, R.: Computation offloading to save energy on handheld devices: a partition scheme. In: Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems. CASES '01, New York, NY, USA, ACM (2001) 238–246

[3] Silva, J.N., Veiga, L., Ferreira, P.: SPADE: scheduler for parallel and distributed execution from mobile devices. In: Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing. MPAC '08, New York, NY, USA, ACM (2008) 25–30

[4] Kemp, R., Palmer, N., Kielmann, T., Bal, H.: Cuckoo: A Computation Offloading Framework for Smartphones Mobile Computing, Applications, and Services. Volume 76 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg (2012) 59–79

[5] Chun, B.G., Ihm, S., Maniatis, P., Naik, M., Patti, A.: CloneCloud: elastic execution between mobile device and cloud. In: Proceedings of the sixth conference on Computer systems. EuroSys '11, New York, NY, USA, ACM (2011) 301–314

[6] Guo, Y., Zhang, L., Kong, J., Sun, J., Feng, T., Chen, X.: Jupiter: transparent augmentation of smartphone capabilities through cloud computing. In: Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds. MobiHeld '11, New York, NY, USA, ACM (2011)

[7] Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual network computing. Internet Computing, IEEE **2**(1) (January 1998) 33–38

[8] Vanderdonckt, J.: Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments. In: Interraccion 2010. (September 2010)

[9] Demeure, A., Sottet, J.S., Calvary, G., Coutaz, J., Ganneau, V., Vanderdonckt, J.: The 4C Reference Model for Distributed User Interfaces. In: Autonomic and Autonomous Systems, 2008. ICAS 2008. Fourth International Conference on, IEEE (March 2008) 61–69

[10] Manca, M., Paternò, F.: Flexible support for distributing user interfaces across multiple devices. In: Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer-Human Interaction: Facing Complexity. CHItaly, New York, NY, USA, ACM (2011) 191–195

[11] Miller, B.A., Nixon, T., Tai, C., Wood, M.D.: Home networking with Universal Plug and Play. Communications Magazine, IEEE **39**(12) (December 2001) 104–109

[12] Sousa, J.P., Poladian, V., Garlan, D., Schmerl, B., Shaw, M.: Task-based adaptation for ubiquitous computing. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **36**(3) (May 2006) 328–340

[13] Raman, R., Livny, M., Solomon, M.H.: Matchmaking: An extensible framework for distributed resource management. Cluster Computing **2**(2) (June 1999) 129–138

[14] Silva, J.N., Ferreira, P., Veiga, L.: Service and resource discovery in cycle-sharing environments with a utility algebra. In: Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on. (2010) 1 –11