SEVENTH FRAMEWORK PROGRAMME

CloudSpaces

(FP7-ICT-2011-8)

Open Service Platform for the Next Generation of Personal Clouds

D2.1 Roadmap of outcomes

Due date of deliverable: 31-03-2013
Actual submission date: 25-03-2013

Start date of project: 01-10-2012 Duration: 36 months

Summary of the document

Document Type	Deliverable
Dissemination level	Public
State	Final
Number of pages	46
WP/Task related to this document	WP2
WP/Task responsible	URV
Author(s)	Pedro García López, Adrián Moreno, Marc Sánchez Artigas, Marko Vukolic, Hamza Harkous, Thanasis Papaioannou, Hao Zhuang, Stuart Langridge
Partner(s) Contributing	URV, EUR, EPFL, CNC
Document ID	CLOUDSPACES_D2.1_130325_Public.pdf
Abstract	The present document will define in detail the Personal Cloud paradigm, its relationships with Personal Computing, and it will explain how the next generation of Personal Clouds will provide advanced services like interoperability and privacy. Roadmap of outcomes, project dependencies and partner roles.
Keywords	Cloud storage, synchronization, sharing, interoperability, Personal Cloud

Table of Contents

1 Executive summary

The present document has three main goals: creating a taxonomy of existing Personal Clouds, describing the open challenges that Personal Clouds face, and finally explaining the Roadmap of Outcomes of the CloudSpaces project.

In section 2 (Taxonomy of Personal Clouds) we will try to provide a definition and characterization of existing Personal Clouds. We will also describe the state of the art of existing systems.

In section 3 (Open Challenges) we will explain the current challenges and open problems for the next generation of Personal Clouds. We will address three main challenges in Privacy, Interoperability and Scalable Data Management.

Finally, in section 4 (Roadmap of Outcomes) we will provide a detailed roadmap of actions that the CloudSpaces project will undertake in the first two years of the project in order to deal with the aforementioned open challenges.

2 Taxonomy of Personal Clouds

2.1 From the Personal Computer to the Personal Cloud

In the last few years, the Personal Computer (PC) paradigm has been slowly waning as traditional PCs have been outnumbered by other devices like smartphones and tablets. The traditional PC as the user's unique device including a box with a monitor, mouse, and keyboard is coming to an end. In the next years, users will access their digital lives from a variety of heterogeneous devices like TVs, smartphones, tablets, laptops or even wearable computers.

This change has profound implications in the way we interact with information. The Personal Computer implies that users have absolute control of their device's hardware and software. What it is more important, user's personal data remains private as a direct consequence of PC's local storage. But this recent multi-device approach clearly implies access to Cloud resources that may compromise the privacy and control of user's information.

As stated in the title of this section, we refer to this new multi-device model as the Personal Cloud. The Personal Cloud model defines a ubiquitous storage facility enabling the unified and location-agnostic access to information flows from any device and application. Ubiquitous free Cloud services like GMail, Facebook, Flickr, Dropbox, and many others permit users to access their personal information from any device and location. But we pay a price for these services: the loss of control over our own data. In particular, the business models of most of these Cloud services relies on the monetization of user's data.

On the other hand, several Cloud services (e.g. Wuala, MEGA, SpiderOak, Bitcasa, etc.) have emerged allowing the user to reclaim control over his data through client-side encryption. With most of these providers, the user further has the choice between keeping his data encrypted or sharing it in plain form with others. The latter option has the advantage of allowing data to be processed at the server side, thus enabling additional services (e.g. document editing, printing). Nevertheless, this approach of specifying a privacy policy per group of files is not convenient for most of the users, and this partially justifies the small user base of these services. The current solutions for privacy preservation are analogous to antivirus programs where the user has to scan each file or group of files manually and individually before using them. Most users would prefer to have the security analysis run in the most automatic and transparent manner possible. Along these lines, there are no Cloud solutions ensuring that the privacy of user's data is automatically guaranteed, with minimal intervention on his side.

In addition, we are facing a decisive battle to decide who is going to store our data in the next years. In a recent report, Forrester research forecasts a market of $12 billion in the US in paid subscriptions to personal storage services by 2016 [1]. In this line, we are witnessing a generalized battle of Cloud providers to store our information: ranging from Dropbox, Box, SugarSync and Ubuntu One, to Apple's iCloud, Microsoft Skydrive and Google's Drive. These Store Wars represent a battle for supremacy that create walled gardens environments favouring vendor lock-in and precluding interoperability.

But the future is not that bleak. The Personal Cloud model with all its advantages can be realized without relinquishing our privacy or requiring a significant effort and expertise.

The CloudSpaces project aims to create the next generation of Personal Cloud where all users, regardless of their technical expertise, will retake control of their information. In the next sections we will try to present a clear definition of the Personal Cloud concept, and we will shed light on the current state of the art about Personal Clouds on the market.

2.2 Definition of Personal Cloud

First of all, we must propose a definition and classification for the concept of Personal Cloud.

3S Personal Cloud definition: The Personal Cloud is a unified digital locker for users' personal data offering three key services: Storage, Synchronization and Sharing. On the one hand, it must provide redundant and trustworthy Cloud data storage for users' information flows irrespective of their type. On the other hand, it must provide syncing and file exploring capabilities in different heterogeneous platforms and devices. And finally, it must offer fine-grained information sharing to third-parties (users and applications).

We also include two external definitions:

Respect Network's definition [2]: A Personal Cloud is a Cloud-based virtual computer that combines event processing and personal data. Personal Clouds serve as the online representative of an entity, often a person. Personal Clouds have an operating system that runs applications, processes events, and manages data on behalf of and under the direct control of their owners.

PC Magazine Encyclopedia's definition [3]: A small server in a home or small business network that can be accessed over the Internet. Designed for sharing photos and videos, Personal Clouds enable viewing and streaming from any Internet-connected personal computer and quite often from major smartphones. Although Personal Clouds function in a similar manner to any private Cloud set up in a company, their primary feature is easy installation for the average personal computer user.

As we can see from the previous definitions there are still divergent views of what it is a Personal Cloud. Probably the most restrictive one is the PC Magazine definition which reduces the term to home-based small personal server resources. On the contrary, the Respect Network definition is the most ambitious definition considering the term a Cloud-based virtual computer with both processing and data management capabilities.

The CloudSpaces 3S definition aims to provide a least common denominator for Personal Cloud Services. It goes further than classical Storage as a Service (STaaS) models because it mandates syncing and sharing as essential services. But it does not include computing or applications like the model envisioned by the Respect Network. In our model, applications will interact with the Personal Cloud thanks to interoperability standards.

The actual implementation of the Personal Cloud can be realized using different architectures such as a public Cloud, a private Cloud, or even a hybrid Cloud. Furthermore, the name "Personal" should not be misunderstood: the Personal Cloud is not only aimed for end-user's private digital information, but it can also refer to enterprise data and professional activities.

We cannot say that a Personal Cloud can be classified as IaaS (Infrastructure as a Service), SaaS (Software as a Service), or PaaS (Platform as a Service), because we consider that the three of them can also be included in the Personal Cloud model. For example, providers such as Dropbox, U1 or Box tend to offer storage APIs to third party applications (IaaS). And such providers also offer web-based rich interfaces to their stores (SaaS). Finally, although less prevalent today, Personal Clouds could offer richer PaaS services to third-party applications willing a more flexible interaction with their data (e.g. U1DB data store or the "Cloudlets" concept [4]).

Finally, depending on their implementation of sync and share features, we can classify Personal Clouds as oriented to **content collection**, **distribution**, or **collaboration**. Collaborative Personal Clouds must provide advanced sync (versioning, conflict resolution) and sharing (access control) capabilities to permit treating Personal Cloud spaces as collaborative workspaces. Here, we summarise these three purposes.

2.2.1 Collaboration

User group	Read access	Write access
Owner	Yes	Yes
Rest of participants	Yes	Yes

Table 2.2a: Access control in a collaboration setting

The most obvious use case for many people is to have access to a set of files: so that everyone in the group can work on these files together. For example, the files could be those making up a report, on which everyone in the group collaborates. There may be multiple textual documents, multiple images, and so on. Each participant can edit any document at any time, and gradually the final article is pulled together through collaboration by all participants. This has the advantage that collaboration can be asynchronous: participants do not need to work on the same document(s) at the same time.

Collaboration is problematic because of conflicting edits. If two people work on the same document at the same time, there is a risk that one person's edits will overwrite the other person's. Similarly, if editing takes place "offline" (i.e. an edit happens and then the document is re-sent to the collaborative repository) it increases the risk of conflicting edits.

Any solution proposing to have many people accessing a repository of files —for the purposes of collaboration— must take this into account. Some possible solutions are:

- *Real-time*. Only allowing direct online access to documents and reflecting all changes in those documents immediately to all consumers (e.g. as used by Google Docs).
- *Locks*. Locking a document for changes and, therefore, only allowing one person to change it at a time, before releasing the lock (e.g. classic source control systems such as Visual SourceSafe).

- *Versions*. Storing multiple conflicting versions of a document in the repository and providing some way to resolve the conflicts (e.g. modern source control systems such as git, bzr, and subversion; most Personal Cloud solutions such as Ubuntu One).

2.2.2 Collection

User group	Read access	Write access
Owner	Yes	Yes
Rest of participants	Yes	Yes*

* No write access to documents created by other participants.

Table 2.2b: Access control in a collection setting

Collection is the concept of using the repository as a "dump box" for many participants. This is useful to accumulate documents from many sources. One common example is that of a group of friends returning from holiday; each person took many photos during the holiday, and the friends would like to collect all of the photos from everyone in one place. So a "collection"-style Personal Cloud repository is created: each person can add their photos to this "collection" repository, and everyone can view all the photos, but no person can delete photos provided by another person. Thus, ensuring the integrity of the repository.

It is also possible to deny read access to all participants except for the files they particularly added to the repository: this makes possible to use a collection-based repository for (non-anonymous) voting, submitting suggestions, or contribution toward a wider cause.

Collection is a well-defined use of a Personal Cloud repository, and is commonly used on sites such as Facebook for collecting photos of an event. One might also think of hashtags on Twitter as being conceptually similar to collection-style repositories: hashtagging a tweet is similar to contributing a thought to a collection-style repository defined by the hashtag.

2.2.3 Distribution

User group	Read access	Write access
Owner	Yes	Yes
Rest of participants	Yes	No

Table 2.2c: Access control in a distribution setting

Distribution deals with a use case where one (or a small number) of repository owners want to share documents with a large number of repository "consumers". The consumers have

read-only access to the repository, and the owners can add new files to the repository in order that the consumers have access. This style of repository is commonly used for shared sources of documentation, publishing, and so on; for example, a magazine could be distributed in electronic form by having the publisher add a copy of the magazine in digital form to the repository and then granting magazine readers (or the whole public) read access to the repository. Most web sites could be thought of as Personal Cloud distribution-style repositories.

2.3 State of the art review of existing Personal Clouds

Before beginning with Personal Cloud comparison we will introduce some important concepts and provide definitions of the main aspects that will be analysed. To clarify, we will classify features into the following sections:

- **Storage**. We will take into account features such as Personal Cloud's infrastructure or whether they use advanced techniques such as data deduplication.
- **Sync**. In this part we will describe and analyse aspects related to file synchronisation such as P2P syncing or file versioning.
- **Share**. Different types of data sharing and collaboration will be considered. Including privacy-aware data sharing.
- **Privacy**. We will analyse the type of encryption used by Personal Clouds while files are being transmitted and when they are at rest. In addition we will analyse authentication protocols and software licenses used.
- **Platforms**. We will see the tools provided to allow external access to users' data and the platforms supported by each Personal Cloud.

2.3.1 Storage

Personal Cloud companies offer sophisticated storage services to end-users and enterprises by making use of raw storage. This **storage backend** is often provided by data center owners such as Amazon or Rackspace. However, some other Personal Clouds do have their own infrastructure and do not outsource this task.

Therefore, we can classify Personal Clouds depending on their **nature**: public, private or hybrid. We call Public Clouds the ones that whose services and infrastructure are offered off-site over the Internet. In contrast, a Private Cloud is one in which the services and infrastructure are maintained on a private network. Additionally, Hybrid Clouds include a variety of public and private options with multiple providers. For instance, a company could keep sensible and business critical information in their Private Cloud, while using a Public Cloud to store the rest of their data.

Personal Clouds usually offer their services based on a Freemium business model [1]. In other words, a product is offered free of charge, but a premium product with advanced

[1] Freemium is a business model by which a product is offered free of charge, but a premium product with advanced features at a charge.

features is offered at a charge. Therefore, the storage quota offered for free is an important feature that users consider.

Some Personal Clouds apply restrictions on the **maximum file size**, which can vary depending on whether the file is synced on the desktop application or uploaded through the web interface, or even the type of account.

In order to optimize the storage and bandwidth consumed when maintaining different versions of files some Personal Clouds use **data deduplication** techniques. Using data deduplication, when a user is to upload a file that already exists in the Cloud, he will not need to actually transfer it. Instead, a logical relation between the file and the user will be created. This technique can be applied to different scopes (i.e. across all files in the system, across user's files, etc.) depending on the privacy policy.

Such popular services need to scale well in order to be successful. A system is said to be **scalable** when it is able to accommodate a significant growth in its amount of work and continue its normal functioning. This feature is normally inherited by the storage backend that underlies the service.

Personal Cloud	Nature	Free storage	File size limit	Deduplication	Scalable
Google Drive	Private	5 GB	10 GB	No	Yes
Dropbox	Public (Amazon S3)	2 GB	300 MB (web only)	Yes	Yes
Ubuntu One	Public (Amazon S3)	5 GB	Unlimited	Yes	Yes
Box	Private	5 GB	250 MB (free acounts)	?	Yes
SugarSync	Public (Carpathia Hosting)	5 GB	Unlimited	?	Yes
Cubby	Private	5 GB	2 GB (web only)	No	Yes
SkyDrive	Private	7 GB	2 GB (client), 300 MB (web)	?	Yes
iCloud	Private	5 GB	?	?	Yes
SpiderOak	Private	2 GB	Unlimited	Yes	Yes
Wuala	Private	5 GB	40 GB	Yes	Yes
ownCloud	Private	Customizable	Customizable	No	No

Table 2.3a: Personal Cloud comparison on storage features

As we can see in table 2.3a, most of the analysed Personal Clouds use their own storage backend, others such as Dropbox or Ubuntu One rely on well-known storage providers such as Amazon to store users' data. Most Personal Clouds allow users to increase their free storage by referring friends to their service. We can also observe that there is a huge difference on the limit set to file sizes. OwnCloud lets administrators assign custom values for storage space and file size limit.

As far as we know, only Dropbox, SpiderOak, Ubuntu One and Wuala implement data deduplication mechanisms, but they differ on the scope where this technique is applied. Whereas Dropbox, Ubuntu One and Wuala deduplicate data across all users, SpiderOak only does it across a single user data due to privacy concerns. We assume that, due to the design of the storage backend, all services are scalable in the sense that they can fetch more

resources as the demand increases. We don't consider ownCloud to be scalable since it is designed as a local repository.

2.3.2 Sync

One of the key aspects of Personal Clouds is **file synchronization** (or syncing). We understand it as a two-way file synchronization, which means that a locally modified file is updated in each location this file is present. In addition, if a file is modified remotely, these changes will be automatically updated locally, with the purpose of keeping every copy of a file identical in all locations.

In this line, some companies also implement **P2P file syncing**. We understand P2P file syncing as the ability to keep two or more files identical in different locations without resorting to a central service. It allows companies to reduce the outgoing traffic from the data center, which translates in cost savings. It is also useful for users that want to store the same files on two or more computers avoiding the need to resort to the server.

Some solutions enable users to sync files from **multiple folders** throughout their file system. For instance, when syncing you will need to tell the application that "Folder A" on your laptop should sync with "Folder B" on your desktop. Contrarily, others solutions only sync files that are stored inside a specific folder created for that purpose.

Another interesting feature is **versioning**. File versioning allows users to restore previous versions of a file. Personal Clouds may limit the **version history** to a maximum number of revisions to be kept in the system or a specific period of time. For instance, a company could claim storing all versions of a file done in the last 30 days. This may also vary depending upon whether it is a free account or not.

Personal Cloud	File syncing	Multiple folder syncing	P2P syncing	Versioning	Versions saved
Google Drive	Yes	No	No	Yes	30 days or 100 versions
Dropbox	Yes	No	LAN restricted	Yes	30 days
Ubuntu One	Yes	Yes	No	No	—
Box	Yes	No	No	Yes	10
SugarSync	Yes	Yes	No	Yes	2 (free), 5 (paid)
Cubby	Yes	Yes	Yes	Yes	Unlimited
SkyDrive	Yes	No	No	Yes	25
iCloud	Yes	No	No	No	—
SpiderOak	Yes	Yes	No	Yes	Unlimited
Wuala	Yes	Yes	No	Yes	10
ownCloud	Yes	?	No	Yes	?

Table 2.3b: Personal Cloud comparison on syncing features

About half of the Personal Clouds implement multiple folder syncing. Cubby implements a protocol called *DirectSync* [5] that bypasses the Cloud by using P2P technology for syncing files. Dropbox also has some sort of P2P syncing protocol *LAN sync* [6] that syncs different computers inside the same local network. Again, there is a big discordance about the policy used when storing the file history.

2.3.3 Share

Sharing is an attractive feature that most of Personal Clouds provide, whether it is with users inside the service or with people outside the Personal Cloud. **Internal sharing** is usually offered as an integrated functionality in the user interface. Whereas **public sharing** is commonly offered as direct HTTP links that allow other users to access to certain files or folders.

The sharing infrastructure must ensure **privacy-aware data sharing** from other Personal

Clouds. It must also guarantee controlled access to external users and applications of one's personal data.

Real-time collaboration allows multiple users to edit a file at the same time. So users can see where in the document or file a particular editor is currently writing.

Interoperability between Cloud providers gives users a sense of control and ownership over their Personal Cloud. Since the Personal Cloud is the mediation point between users and services, being able to easily withdraw from a particular Cloud provider increases ownership and control; the Personal Cloud feels truly personal, because the user is not "locked in" to one provider.

Personal Cloud	Internal sharing	Public file sharing	Privacy-aware data sharing	Collaborative editing	Interoperability
Google Drive	Yes	Yes	No	Yes	No
Dropbox	Yes	Yes	No	No	No
Ubuntu One	Yes	Yes	No	No	No
Box	Yes	Yes	No	No	No
SugarSync	Yes	Yes	No	No	No
Cubby	Yes	Yes	No	No	No
SkyDrive	Yes	Yes	No	Yes	No
iCloud	Yes	No	No	No	No
SpiderOak	Yes	Yes	Yes	No	No
Wuala	Yes	Yes	Yes	No	No
ownCloud	Yes	?	No	No	No

Table 2.3c: Personal Cloud comparison on sharing features

In table 2.3c we can see that only SpiderOak is considered to implement a privacy-aware data sharing scheme. SpiderOak allows users to password protect all their *ShareRooms*[2] so that only the people they want to give access to their data can view or download their shared files. Each share room has its own private, secure URL so users can easily share them with only the people they want.

[2] `https://spideroak.com/Online_File_Share`

Only Google Drive and SkyDrive let multiple users collaborate simultaneously on the same file from any computer. When someone makes changes to a document, the other person can see the changes in real-time and respond to the edits immediately.

Today, no Personal Cloud provides interoperability with other platforms.

2.3.4 Privacy

Personal Clouds must ensure that user data is not accessed by third-parties and only authenticated users are granted access. Some companies use standard **authentication protocols** such as OAuth [7], others opt for using their own mechanisms.

As a security measure, some companies store user data encrypted. Many Personal Clouds can provide **server-side encryption**; meaning that users delegate to the Cloud the task of protecting their files and managing their keys. As an alternative, **client-side encryption** allows users to encrypt their data before it is transmitted to the server. So the user is responsible for managing the keys and the service provider is unable to decrypt their data, adding an extra layer of security.

Besides the fact of having the files secured when they are at rest in the server, it is also essential to assure their privacy when they are being transmitted to and from the server. Personal Clouds usually use **HTTPS** to communicate to their services either from the desktop application or other tools such as the REST APIs or the web interface.

Services and applications offered by Personal Clouds are released in a variety of **licenses** in order to govern their use and redistribution. Licenses grant rights and impose restrictions on the use of software. Software licenses can generally be fit into the following categories: proprietary licenses and free and open source. The significant difference that distinguishes them is the terms which state whether or not an end user can further distribute or copy the software.

Personal Cloud	Authorization protocol	Client-side encryption	Server-side encryption	Secure channel (HTTPS)	License
Google Drive	OAuth 2.0	No	No	Yes	Proprietary
Dropbox	OAuth 1.0	No	Yes (AES 256-bit)	Yes	Proprietary
Ubuntu One	OAuth 1.0	No	No	Yes	Proprietary (server); GPLv3 (client)
Box	OAuth 2.0	No	Yes (AES 256-bit)	Yes	Proprietary
SugarSync	Custom. Token-based	No	Yes (AES 128-bit)	Yes	Proprietary
Cubby	—	No	Yes (AES 256-bit)	Yes	Proprietary
SkyDrive	OAuth 2.0	No	Yes	Yes	Proprietary
iCloud	—	No	Yes (AES 128-bit)	Yes	Proprietary
SpiderOak	HTTP basic authentication	Yes	—	Yes	Proprietary; GPLv3 (some tools)
Wuala	—	Yes	—	Yes	Proprietary
ownCloud	—	No	Yes	Yes	AGPLv3

Table 2.3d: Personal Cloud comparison on privacy features

As we observe in Table 2.3d, most Personal Clouds implement the OAuth protocol on their API services, either the 1.0 or 2.0 version. Others like SugarSync use their own authentication mechanisms. In general terms, a user makes a call providing his credentials to obtain a refresh token. Next, the user will use this refresh token to obtain an access token that will grant access to the user's resources. As the access token has an expiry date, the user will have to use the refresh token periodically to extend its lifetime.

Very few Personal Clouds provide built-in client-side encryption. From the ones analysed in this comparison, only SpiderOak and Wuala encrypt files locally on the device before they are uploaded. Therefore, no one not explicitly authorized by the user can see their

data, not even the storage provider. As a side effect, it is impossible to recover the password in case the user forgets it.

Unanimously, all services establish a secure channel between user's computer and the storage provider before data is transmitted. That way, no one can eavesdrop on the transfer.

Only ownCloud is fully available in a free open source license. Ubuntu One, though, provides its client software and protocol libraries as open source. SpiderOak provides only a set of tools under GPLv3 license.

2.3.5 Platforms

In order to access to user data from an external application, Personal Clouds must implement an application programming interface (API). Providing a **public API** allows developers to integrate their application on top of the storage system. When used in the web environment, an API is typically defined as a set of HTTP request messages and XML or JSON response messages, also known as REST API.

Another way to allow external access to user data is through the **WebDAV protocol**. It provides a framework for users to create, change and move their documents. Most current operating systems provide built-in support for WebDAV.

Being able to access to users' stored data from a web browser is an essential functionality. **Web interfaces** typically allow users to manage their files (move, delete, upload, download, etc) and access to extra tools such as generating public links. Additionally, it is also important to provide **clients** for as much operating systems as possible to allow access to users regardless of their devices.

Personal Cloud	REST API	WebDAV support	Web interface	Windows client	OS X client	Linux client	Android client	iOS client	Windows Phone client	BlackBerry client
Google Drive	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No	No
Dropbox	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Ubuntu One	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Box	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
SugarSync	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No	Yes
Cubby	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No
SkyDrive	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	No
iCloud	Yes	No	Yes	Yes	Yes	No	No	Yes	No	No
SpiderOak	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Wuala	Yes. Limited	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No
ownCloud	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No

Table 2.3e: Personal Cloud comparison on privacy features

Wuala provides a limited API only supporting GET operations, it does not include any login or write facilities. Cubby and ownCloud, which do not provide a REST API, implement support for the WebDAV protocol. Though, Cubby has certain limitations such as not being able to move or rename files.

All analysed Personal Clouds have a web interface and most of them have clients for the main operating systems and mobile devices, which allow users to easily interact with their files. However, iCloud is mainly restricted to Apple's products.

3 Open Challenges

3.1 Privacy

Traditional Cloud Computing Solutions

The Personal Cloud paradigm aims at enabling users to pervasively access their data and backup their files online. One of the basic assumptions by the majority of commercial Personal Cloud services (e.g. Dropbox, Skydrive, Google Drive, etc.) is that the service provider is trusted to neither expose nor misuse users' data. This trust is typically guaranteed via service agreements acknowledged by the user and subject to certain legislations. However, this might not be enough to attract certain types of users to these services or to make existing users willing to share sensitive content to the Cloud.

Furthermore, sometimes the user's data is utilized by the Personal Cloud company to provide him with services of his interest, such as Cloud music streaming (Ubuntu One) or online file editing (Google Drive). On other occasions, his files might be mined for extracting his shopping preferences and displaying advertisements accordingly. In both cases, the user is not in control of the access to his data. Even if he is not willing to use such services, he has no choice but to give the Cloud service provider access to his data.

Cryptography Based Solutions

Commercial Solutions

On the other hand, several approaches are emerging as attempts to protect user's data from the misuse by the Cloud providers. Some existing commercial Personal Cloud services, such as MEGA, SpiderOak, Wuala, Bitcasa, etc, include encrypted file storage, where the client has full control on the encryption keys. The main encryption technique used in most of them [8, 9, 10] is **convergent encryption** [11], where the same plain files are mapped to the same encrypted files. Such a feature is **vital for data de-duplication**: since the server is able to store the two identical files once, this enables massive savings on the storage space required to satisfy the user's demand. This is mainly due to the fact that a large part of the content shared is common across the users. Regardless of the security issues associated with this approach, it still fits the business model of such solutions. Combining convergent encryption of files with asymmetric encryption for key sharing, some of **these systems provide the typical synchronization and sharing capabilities on the encrypted files**. Furthermore, some give the user the choice of sacrificing privacy for viewing the files on a web browser or for using other web-based services. In such cases, the files' encryption keys are sent to the server.

Academic Approaches

More sophisticated approaches exist in academic research. For example, Kamara and Lauter [12] built a **secure Cloud storage layer** to be deployed on top of existing Cloud infrastructure, where the Cloud service provider is not completely trusted by the customer. Such a layer provides data confidentiality and integrity features. The main cryptographic techniques deployed are **searchable encryption, attribute-based encryption, and proofs of storage**. Searchable encryption allows encrypting a search index so that its contents are hidden except to a party with the appropriate token [12]. With attribute-based encryption, a

user can encrypt a message under a public key and a policy. Decryption only works if the attributes associated with the decryption key match the policy used to encrypt the message. A proof of storage is a protocol executed between a client and a server, allowing the server to prove to the client that it did not modify its data [12].

Intermediate Choices

Based on the above, one can notice that the existing approaches leave the user between **two choices, either locally encrypting his data or revealing them in plain form to the Cloud provider**. There is **no intermediate choice**, where such data can be partially revealed in forms that allow certain services to be provided. It is important to notice here that CloudSpaces is not only envisioned as a medium of sharing generic binary files, but as a general framework that governs the interaction between the Cloud provider and the client and allows the latter to privately share his personal information or files. From that perspective, **various privacy levels** can be achieved based on the type of data that is being shared. We summarize below some of the most popular **privacy-enhancing techniques** to achieve such levels.

Privacy-Enhancing Techniques

- **The randomization method**: The randomization method is a technique for privacy-preserving data mining in which noise is added to the data in order to mask the attribute values of records [13]. The noise added is sufficiently large so that individual record values cannot be recovered. Complementary techniques are designed to derive aggregate distributions from the perturbed records. Data swapping [14] among different data streams is another form of data perturbation.
- **The k-anonymity model, l-diversity and t-closeness**: In the k-anonymity method [15], the central idea is to reduce the granularity of data representation with techniques such as generalization and suppression. This granularity is reduced in such a way that any given record maps onto at least k other records in the data. The l-diversity model was designed to overcome some weaknesses in the k-anonymity model, especially when there exists homogeneity of sensitive values within a group [16]. This also leaded to another privacy definition, t-closeness [17], which considered the sensitive attribute distribution in each class, and its distance to the overall attribute distribution. A variation of the k-anonymity method for anonymizing graphs called k-automorphism was proposed in [18].
- **Distributed privacy preservation**: In some cases, individual entities wish to derive aggregate results from data sets which are distributed across these entities. While the individual entities may not desire to share their entire data sets (data hiding), they may consent to limited information sharing with the aid of cryptography [19]. The overall effect of such methods is to maintain privacy for each individual entity, while deriving aggregate results over the entire data. Other cryptographic techniques such as fully homomorphic encryption [20] perform computations on encrypted data without decrypting. As these cryptographic techniques mature, they may open up new possibilities for Cloud Computing security. At present, fully homomorphic encryption is so computationally demanding that practical applications are years away.

Open Challenges

Automated Privacy-Aware Data Sharing

Another disadvantage of existing solutions is that user's privacy is dependent on user settings, not only on the system's intelligence. If the user does not select what files are important enough to encrypt, the system cannot help. One of the main reasons that kept the huge user base away from Cloud solutions integrating client side encryption is that most users prefer minimal involvement in tweaking the system to their needs. They prefer solutions that are tailored to their needs and work out of the box. Normal users do not prefer or do not have enough expertise to decide on the privacy technique to use on a file-by-file basis.

Hence, one of the main challenges in CloudSpaces is to allow the automation of privacy-aware data sharing decisions. We believe that, at the core of this automation, there should be a systematic mechanism for the **evaluation of the privacy risk** posed by having the data in a certain format or context. Risk evaluation takes as an input the data contents and semantics along with its social, organizational, and technological context and outputs a quantitative measure of the perceived privacy risk. Evaluating that risk is a product of measuring the likelihood of disclosure and the danger of such disclosure. The former depends on the form in which the data exists, the relations of the data sharer to the data observers, etc. The danger is dependent on the importance of the data.

This mechanism can be applied to any data in the context of CloudSpaces, whether on the user's disk or on the Cloud. On top of this, numerous applications and optimization problems can be envisioned, some of which we explain in what follows:

1. **Automatic privacy firewall**. The user can initially proceed in using the system as he does with traditional Cloud services (e.g. Dropbox or Ubuntu One). He can select to backup the data or share it with other parties (e.g. applications, users). However, the system automatically acts as a precursor to detect whether the privacy risk posed by having the data stored in its current format or moved from the user's machine to the Cloud is higher than the maximum risk allowed by the user's settings.
2. **Automatic recommendation of data outsourcing**. The decision of moving the data from the user's device to the Cloud (or vice-versa) depends on several factors other than the privacy risk. On one hand, there are the different types of costs, such as the user's bandwidth, storage cost, computational cost, etc. On the other hand, there are the services that the user expects from having the data in the Cloud or locally at his own device. These factors can be combined by the system to automatically generate the optimal policy of handling the data. This policy can be realized for instance by certain privacy enhancing techniques or by transferring the data to another device.

Such aspects of automated privacy aware data sharing have not been targeted before, especially in the Cloud setting. They relieve the user from the burden of specifying his privacy preferences for each file through learning such preferences. Generally, we aim at achieving automated privacy policies. Nevertheless, there are still some users who are interested in manually specifying their privacy policies for certain types of files. Hence our system still additionally has the following goal:

- Support for Manual User Policies for Privacy Aware Data Sharing: Users are allowed to set policies that manage the sharing of their data. Furthermore, such goal can be

extended to allow the automatic sharing of data within certain user groups based on user-set policies. The difference with existing Cloud solutions is that, previously, users had to manually set policies for each file or file type and has to share new files again with the same users. In our case, we envision that the users can set their sharing policies once for certain data types (or data semantics as we see later), and the data will be automatically transferred to others in a privacy-aware manner thereafter. Although the definition of sharing policies can be further automated, users might not find it convenient to trust a system to autonomously share data with others on their behalf.

Context Aware Data Sharing

In order for the system to take automated privacy decisions, it should be aware of the context in which the data is being shared and should include such context in the decision making process. For example, it is important to know whether the user is using a smartphone or a desktop computer. The parties with which the data is being shared is also significant: a public Cloud is different in the perceived risk from a private Cloud. A work colleague with which we want to share certain documents should be treated in a more trusted manner than a stranger with which we want to exchange photos. All such context information are crucial in arriving at the appropriate privacy decision, and we envision that such an automated privacy control system should take them into account.

Semantics Aware Data Sharing

Existing Cloud solutions do little when it comes to allowing the users to filter what they want to upload to the Cloud. The maximum control they currently provide is filtering according to file extensions. In our vision, future privacy-aware Cloud solutions should allow users to reason about data semantics instead of file extensions. Semantics include:

- File types (documents, media, etc.).
- File metadata (author in PDF files, album in audio files, etc.).
- File contents (extracted via text analysis).

A user would more probably care about the sensitivity of documents he has authored than music of a certain artist. Towards that purpose, the user's files, which have different formats, need to be organized in a way that allows the computer to take decisions concerning the sharing technique to be used for each. This process of capturing the data semantics into Semantic Web conceptualizations is called semantic lifting. Existing solutions perform data lifting through various mechanisms. Strigi[3] crawls the file system to extract metadata and file contents. Examples are the audio files from which the artist can be determined and the PDF files from which the author can be derived [21]. Other systems like Beagle++[4] and Aperature[5] perform semantic lifting up to varying degrees.

Via such techniques, semantic features can be learned and thus integrated in an intelligent system like CloudSpaces, responsible for taking the privacy decisions.

[3] `http://strigi.sourceforge.net`
[4] `http://beagle2.kbs.uni-hannover.de`
[5] `http://aperture.sourceforge.net`

Summary

An inherent issue in the current Personal Cloud model is that the user either has no control over the trade-off between the level of service provided and the level of privacy guaranteed or has to manually control such trade-off on file-by-file basis. In CloudSpaces, we aim at designing a system that grants the users automated control over this trade-off while still allowing them to specify their own policies governing the data being transferred to the Cloud. Such a system takes into account different factors, including semantic and contextual information and uses an array of privacy-enhancing technologies to protect the user's data from being exposed.

3.2 Interoperability

We consider both syntactic and semantic interoperability. In the Cloud computing context, the former means that Cloud providers are trying to communicate through standardized mechanisms, including interfaces, data formats and communication protocols. Semantic interoperability, on the other hand, means that information is not only exchanged but also interpreted in order to be used it.

3.2.1 Syntactic interoperability

A key feature of the Personal Cloud is the intuitive, easy to use interface to share files and collaborate. A Personal Cloud user can invite another person to share a folder and this person will only see that folder. But perhaps the greatest benefit is that changes in the shared folder are synced and pushed to every user that has given access to the shared folder. Folder sharing is a core feature for market leaders like Dropbox, SugarSync, and the likes, but also shows the "dark side" of Cloud Computing.

To join a shared folder, recipients need to have an account in the same Cloud service, which suggests that the model for sharing (other than public sharing) promotes significant lock-in. If folder sharing is used, there is a sunk cost of moving off the service. In addition to the costs of switching data, the adoption of an alternative solution will require to set up a new account for every user and install software clients on their devices. This sunk cost represents a barrier to exit for large collaborative groups.

Nowadays, it is quite common that a user makes use of distinct Personal Clouds in his everyday tasks. Imagine he wants to keep the same information synced in all his Personal Clouds. His only option is to tell each client to sync a specific folder in the case this capability is supported. Otherwise, he should manually duplicate his data across the different synced folders. In this sense, if a user is to share information with someone outside his Personal Cloud, he should generate a public link and hand it to its addressee.

The lock-in problem has been recognized by the European Network and Information Security Agency as a high risk [22], and the Personal Cloud is not an exception. This requires new technical means to share data horizontally across Personal Clouds and eliminate the lock-in problem. The root of this difficulty is the lack of interoperability. Currently there exists no protocol or proceeding to interoperate among different Personal Clouds.

In this sense, semantic modeling of data to provide a platform independent data representation would be a major advantage in the Personal Cloud. This method can formulate transformations from one representation to another, so the data appears native when it arrives at the target Personal Clouds, or Clouds, from the source. Semantics will allow Small and Medium Enterprises (SMEs) to enter the Cloud by providing richer forms of collaboration beyond folder sharing.

It is not hard to imagine a small enterprise that uses, not a single Cloud, but multiple Clouds to form an information-sharing environment in which business information is available in the right context, to the right person, and at the right time. This could be done for several reasons, including the amount of free storage per Cloud, differences in data sensitivity with regards to the proposed Terms of Service, the number and type of personal computing/storage devices that a particular Cloud storage provider supports, etc.

A large number of Internet services, including Personal Clouds, use OAuth [7] to authenticate applications and allow access to restricted resources through REST APIs. OAuth authentication eliminates the need to provide sensitive account information, such as a user name and password, to any applications that need access to user's data. Instead, OAuth protocol uses tokens that authenticate tools and applications to access certain services on the user's behalf and also provides the ability to restrict access using scopes. This makes it possible to authorize different applications with separate tokens, and revoke tokens individually, if necessary.

OAuth tokens allow the service to specify the scope of the access request. Therefore, services may have specific scopes such as read-only, read-write, or full-control. For instance, an application could have a read-only access token while another application could have read-write access to the same user data. Therefore, OAuth would provide all necessary tools to guarantee fine-grained access that would allow Personal Clouds to share concrete files and folders with external services.

Nevertheless, OAuth is just a first step and it is not solving the authorization problem among heterogeneous providers. In this line, two standards communities are building solutions on top of OAuth: OpenId Connect [23] and UMA (User-Managed Access) [24]. Whereas OpenId Connect offers single sign-on, session management, and identity claims, UMA aims to standardize access control by third parties on top of OAuth.

As far as we know, each Personal Cloud implements its own syncing protocol and does not follow any standard protocol for syncing files between clients and servers. In addition, syncing protocols tend to be quite complex. For these reasons, and taking into account that companies tend to be reluctant to significant changes, we consider it appropriate to address the matter of interoperability from the APIs point of view, which would be much less intrusive than adapting their syncing protocols.

As a first approach in building interoperable APIs, we consider implementing adapters for each Personal Cloud that would translate requests and responses from one service to another. In this line, we will also consider using existing standards like UMA to achieve interoperability and access control. This would substantially decrease the intervention from Personal Clouds, which would only be required to perform little changes in order to handle these external requests. The aforementioned adapter could be placed either in the client or the server side, depending on where we want to set the abstraction layer. In due time, and after assessing the correct functioning of these adapters, Personal Clouds could start to

integrate them into their APIs.

New forms of interoperability have been emerging in these days. On one side, there is the possibility to integrate an application into a Personal Cloud. As an example, Google Drive integrates third-party applications that use Drive SDK[6] so that users are able to use these applications directly over their stored files. On the other side, there is the antagonistic concept, which consists of integrating the Personal Cloud experience into an external application. For instance, Box Embed is an HTML5-based framework that embeds the entire Box experience on a website, forum, or blog.

3.2.2 Semantic interoperability

While the advanced progress in syntactic interoperability at the platform and infrastructure level provides the technical means for easily sharing data and applications across Clouds, semantic interoperability will make it possible to create repositories that transcend the differences between the vocabularies in disparate Personal Clouds. The abstraction from the file to an object format with metadata may be used to find data objects remotely by performing queries for specific metadata values.

To establish the two types of interoperability, we identify the following challenges.

- Schema matching
- Emergent semantics
- Adoption challenges

Schema matching: Different Cloud providers have different data schemas to store the data. Thus, for establishing interoperability, we need schema matching tools which establish attribute correspondences between the different Cloud schemas. The schema matching techniques consider two input schemas, which are designed independently and they would like to find the correspondences between the schema attributes.

Attribute correspondences are relations between the schema elements (i.e. attributes of the schema). They can be used to define schema mappings which are more complex rules that explain how the data stored in the Cloud is related to each other. Such mappings are needed for example to exchange data items or reformulate queries (i.e. translate the queries posed to one of the Clouds to a query against the other Cloud). More precisely, mappings express the exact logical relations while mapping expressions express the data transformation rules (see [25]).

However, Cloud schemas are often not just a homogeneous set of attributes, but they can be decomposed into smaller units or building blocks. In business settings, these building blocks often correspond to business concepts. It is very natural to consider not only correspondences between individual attributes, but rather complete business concepts. We call such correspondences between concepts micro-mappings. Micro-mappings are groups of attribute correspondences that relate to concepts from the involved business schemas, which improve automatically generated correspondences.

[6] http://support.google.com/drive/bin/answer.py?hl=en&answer=2500820

Meanwhile, we accept that the schema matchers are not perfect and we try to identify possible errors. Once we identify these situations, we would also like to correct these errors. Our design for schema matchers should deal with the potential problems. For example, how can micro-mappings be used to detect errors in initially created mappings and how can we eliminate these errors?

Emergent semantics: The overall principle of deriving semantic information from a collection of mappings and iteratively refining this information to capture the most probable semantic relationships is often referred to as Emergent Semantics technique [26]. In CloudSpaces, we shall design the emergent semantic techniques as well as the quality metrics. We plan to apply emergent semantics in the domain of the interoperability and collaboration across the multiple Clouds. On the technical level, this requires a precise understanding of micro-mapping composition issues and identifying further features of micro-mappings.

The semantic interoperability model implemented for the Personal Cloud should be addressed at the service and data level, which is the SaaS level. At this level, the focus is on enhancing the functionality provided by the Personal Cloud to achieve convergence between the available products and CloudSpaces. Since content sharing will be a complex, dynamic process with changing participants, relationships and tasks, semantic interoperability will need to be established in a distributed bottom-up approach.

By bottom-up approach, we want to emphasize that interoperability emerges through the interaction among the different entities, each one employing different semantics. For example, in CloudSpaces, different Cloud providers might employ different semantics for data storage. In this case, bottom-up semantic interoperability would emerge by having the different Clouds negotiating semantic mappings among them, whether pairwise or in larger coalitions. The term "bottom-up" contrasts with the "top-bottom" case where a standard invented by a standardization body imposes the semantic mappings to the various Clouds. This decentralized bottom-up process has been an important topic of investigation in the area of peer-to-peer data management [27] and decentralized agent communities [28]. In these cases, semantic interoperability has been in general established by mappings and transformation of data [29, 30]. To facilitate semantic interoperability, we will design and implement two types of mappings:

- Pairwise schema-level mappings that will be used to mediate the various schemas and ontologies used in CloudSpaces.
- Instance-level schemas that will be basically used to link and exchange data between related but heterogeneous instances.

Emergent semantic techniques iteratively improve initially constructed mappings. In order to measure these improvements, we need to develop quality metrics for the mappings. These metrics try to exploit other features than constraint violations: such as the number of correct circles or the difference of confidence values in the case of ambiguous correspondences. Some of the metrics can be directly used in emergent semantic algorithms, while others assess the quality of available correspondences and/or micro-mappings, thus they can contribute to designing appropriate repair techniques.

Adoption Challenges: Currently, the Cloud market is very fragmented, with several innovative companies (Dropbox, SugarSync) and major players entering the fields (Apple iCloud, Microsoft, Google, Amazon). Since all players are fighting for the market share, they could simply ignore CloudSpaces' initiative and wait for a de facto standard (winner vendor). Furthermore, there are no even widely accepted semantic interoperability standards for the Cloud. From a technical perspective, and in addition to lock-in avoidance, interoperability will bring additional benefits to the Personal Cloud like a greater fault tolerance by scaling over multiple Clouds or the possibility to automatically react to Service Level Agreement (SLA) violations. Thus, it is major challenge for them to adopt the interoperability open standards proposed by CloudSpaces.

However, the global role of Ubuntu with its Ubuntu One product can change this. Canonical is a major open source provider in the world, so the promotion of an open standard in this setting cannot be completely ignored. Our dissemination plan already planned joint activities and workshops with other providers (Dropbox, SugarSync) to promote the adoption of open standards. So the main contingency plan is to be inclusive with other products and leverage the global role of Canonical in this field.

Another driving force for the massive adoption of CloudSpaces is its bet on open source projects. The contribution to OpenStack Swift will deliver the first open source Personal Cloud solution to the market. Cloud providers and software providers can leverage this infrastructure to provide value-added services. Furthermore, the interoperability with Ubuntu One and the enhanced privacy model will give CloudSpaces a step forward for reaching the masses. In this line, TISSAT includes in its exploitation plan the massive commercialization of the CloudSpaces Personal Cloud platform on top of their data centers. To conclude, another contingency plan is to push our developments to massive audiences following OpenStack. This will ensure worldwide attention and adoption in many cases.

Finally, interoperability standard has a two-fold contribution, which attracts more and more Personal Cloud providers:

- Interoperability among Personal Clouds can help reach critical mass if many solutions can share and export data. This will avoid vendor lock-in and thus attract more users and companies to the platform.
- Interoperability with third-party applications can boost the creation of value-added services on top of the platform. Such value-added services (like eyeOS) are critical to compete with the vertical solutions offered by major players. An ecosystem of services and applications working on top of Personal Clouds can be attractive alternative to major integrated solutions.

3.3 Scalable data managament

Today, Personal Clouds can be seen as the 2010's counterpart of 1970's idea of a Personal Computer (PC). Similarities are obvious: namely, today's Clouds, and in particular Cloud storage, resemble very much the mainframe approach in the era that predates PCs. Like PCs

gave control over computing to individual users, Personal Clouds should give control over Cloud Computing to individual users.

From a personal perspective, Personal Cloud storage should satisfy a few requirements:

- **Durability**. One of the main requirements is that users never lose their precious data, photos, videos, etc. under any circumstances, which includes natural catastrophes;
- **Availability**. Data should be available to the user at any time and from any device, regardless of geographical location and allow the possibility of collaboration and data sharing with selected individuals (or, more rarely, with general public), without compromising any privacy or security concern;
- **Privacy and security**. Ideally, all personal data should remain private and protected against malicious attacks unless the user explicitly states the contrary and decides to release control over his data to storage vendors.
- **Performance**. Most of the data in circulation is unstructured: pictures, audio, movies, documents in editable format, etc. and very often continuously updated. All this mix is growing to huge sizes and is typically exchanged with data centers in form of small chunks of up 4 megabytes in size, which may cause performance bottlenecks due to Internet transfers [31]. Offering a good user experience is a critical requirement and may require rethinking every level of the system, including the data center level.
- **Ease of maintenance**. This requirements is central to the success of the Personal Cloud model as a vast majority of individuals are not IT experts. Ease of maintenance spans hardware maintenance, system configurability and upgrading, but also an effective logical organization of massive volumes of data.

Next we overview open challenges standing out in current versions of Personal Clouds, which include those orchestrated around *home devices* and those centered around *public Clouds*.

3.3.1 Challenges in Home Device Personal Clouds (HDPCs)

The main driving force for the home device Personal Cloud is the increasing connectivity and sheer number of home devices, which will only continue to grow in a near future. As such, having all our devices connected, synchronized and shared through a public Cloud is an unnecessary exercise, as it may expose very sensitive and private data to a third party. There is an obvious need for an infrastructure that will enable HDPC with privacy and security as the first class requirement.

Current Personal Cloud storage solutions orchestrated around home devices are still in their infancy. Until recently, no dedicated HDPC solution existed. Users were typically required either to resort to public clouds or P2P systems to fulfill their storage and data sharing needs, or to take the pain of manually backing up their data to multiple hard drives, share using email services or use portable storage media, with very limited overall availability and with a maintenance nightmare.

Things are slowly improving in the HDPC arena, but are far from ideal as many challenges remain open. Current and recent solutions, such as iOmega Personal Cloud, or myDLink Cloud, promise to offer shared storage to seamlessly share data across multiple home devices, while also providing location-independent availability with a certain level of redundancy and durability. For this reason, these HDPC solutions are typically based on Network Attached Storage (NAS) devices and employ classical RAID (Redundant Array of Independent Disks) systems. However, such solutions suffer from some fundamental limitations:

- RAID systems with large rebuild times coupled with (very often) consumer-level disk drives found in HDPC solutions tend to provide **insufficient durability and reliability** guarantees in particular when disaster tolerance is required.
- HDPC systems based on NAS **lack** the genuine **elasticity**, or the ability to scale up and down depending on the storage needs and access load, that public Clouds offer along with the associated cost savings.
- Inherent **privacy** of HDPC solutions is clearly **in friction with the requirement of data availability**. There is anecdotal evidence of HDPC solutions offering all the user data open to anybody knowing the right IP address, which is used in the HDPC system to provide availability to the owner of the data.
- Sharing capabilities, beyond Personal Area Network (PAN), very often resort again to using public Clouds, due to the lack of truly Inter-Personal Clouds.
- Effort needed for maintenance of the current HDPC systems is non-trivial.

To summarize, even if we can today connect several laptops or smartphones to a shared home-based storage and access this data from almost anywhere, these solutions lack true privacy-aware sharing capabilities and elasticity, have questionable durability and are difficult to maintain.

On the other hand, there are several promising research approaches to solve the above issues with home devices, albeit none of them is yet ready to provide an immediate change on the technology level. For example, [32] presents an approach in which users' home gateways are put in the center of the HDPC. A home gateway is always on and hence presents an interesting choice for a central pivotal point of the HDPC. Besides orchestrating HDPC storage and services, home gateways act as home device proxies to public clouds. Moreover by leveraging neighbor's gateways in a peer-to-peer fashion, this approach aims at offering numerous advantages to HDPC, including quick start up delay, robustness to failures and higher quality of experience.

Anzere [33] is a HDPC storage system for partial, policy-based data replication. Anzere accounts for dynamic changes in personal data repository membership and supports rich set of user-specified replication policies. For elasticity, Anzere resorts to VM acquisition at Public Clouds.

There are other related approaches that can be used in solving the current issues in HDPC storage solutions. One of these approaches are friend-to-friend (F2F) systems (see [34, 35]), which offer a tailored peer-to-peer storage solution where data is exchanged and stored only through nodes owned by trusted users. As such, F2F storage solutions aim at guaranteeing

dependability, privacy and uncensorability by exploiting social trust. Elasticity in these systems comes, in a sense, naturally by leveraging the additional storage offered by peers.

Whereas F2F solutions suffer from availability limitations (for example, in case no friends are online, then data stored in the system will not be accessible) the combination of home gateways with always on gateways and the F2F approach is an interesting avenue for future work. In this context, future challenges lie in understanding precisely the trade-offs between data redundancy, data placement and data availability. Research results are encouraging, as they show with as few as 10 friends, one can obtain good availability [34]. Data availability can be further improved by relying on always-on home gateways for mediation. Another important issue in this context is the level of programmability allowed by home gateways.

However, all these solutions do not necessarily contribute to manageability of HDPCs, which is in stark contrast with public Clouds, where data management is largely outsourced to large commercial cloud providers.

Several efforts aim at improving manageability of HDPC. For example, Perspective [36] provides HDPC storage solution relying on the abstraction of a *view* for efficient data location management. Perspective views simplify replica management in HDPC, being a semantic description of a set of files, specified as a query on e.g., file attributes, device IDs. This approach allows user easier navigation within HDPC data.

Eyo [37] a personal media collections storage system that introduces *device transparency* despite intermittent communication and limited storage, focusing on mobile devices. Device transparency is an abstraction that unifies the collections of objects on multiple HDPC devices into a single structure.

To conclude, we foresee that the research and technological solutions in future Personal Cloud storage will be a hybrid one, which will largely go in the direction of mixing HDPC storage with public Cloud storage tailored to the specific needs of users. This is precisely the research avenue we plan to focus on in the context of CloudSpaces. Besides, Cloudspaces semantic interoperability aim at developing more efficient data management schemes.

In the following, we also overview the challenges posed before public clouds in the context of personal storage.

3.3.2 Challenges in Public Personal Clouds

Many successful Personal Cloud services store data at remote data centers operated by third party vendors like Amazon. In addition to the evident security and privacy breaches and the risk of data lock-in, this practice makes data management highly dependent on the storage infrastructure and the properties of the data management service provided by big providers like Amazon.

When considering the architecture of Personal Clouds like Dropbox, which manage huge amounts of data during synchronization, one critical issue to be addressed is data transfer bottlenecks [31].

For Personal Cloud storage services, and mainstream Cloud computing in general, one of the most important concerns is the time and cost of transferring massive amounts of data

to and from the Cloud. At $120 per terabyte transferred, these costs can quickly add up, making data transfer costs an important issue for Personal Cloud services that base their business model on freemium. To minimize costs, personal storage services use deduplication to save traffic and implement a faster syncing process; for example, Dropbox checks whether a file has been uploaded before by any other user, and links to the existing copy if so. To detect duplicates, these services split every file into chunks of up to a certain size and calculate the hash value of all the chunks using a secure hash function like SHA-256 [38]. The hash values are transmitted from the client to the server and looked up in a hash index. If a chunk is missing from the index, the client uploads the chunk to the data center and adds a new reference of this chunk to the index. Otherwise, the chunk is not sent to the data center because a copy already exists, reducing the time and high cost of Internet transfers.

Yet despite these important cost savings, much room for improvement remains. A recent characterization of Dropbox [39] has revealed that the throughput of storage operations is still significantly low. One reason is that the typically small chunk sizes used to maximize deduplication limit the overall throughput due to TCP slow start-up times. Bundling smaller chunks to increases the amount of storage sent per operation, and reducing signaling traffic, opens the door to additional opportunities for improvement.

Another reason is the distance between the clients and the data centers, which tend to be in a single location like Dropbox in US. However, bringing storage servers close to customer is not trivial and poses commensurate research challenges [40]. There are several trade-offs to consider. With more than one data center, chunks must be replicated and distributed across sites to prevent any loss of deduplication opportunity. This may incur high inter-data center communication and coordination. For example, the Facebook design has a single master coordinate replication. This speeds up lookups but concentrates load on the master for update operations [40], which may not scale out in personal storage services. This requires novel frameworks to support effective deduplication at tens of locations across the globe.

Another opportunity to overcome the high cost of data transfer is to use content distribution protocols like Bittorrent [41]. Since its inception, Bittorrent has proved to be one of the most effective techniques for distributing large content, and major players in the market like Amazon are using Bittorrent to reduce network and hardware costs, as well as the time to download content. The Amazon S3 service is exposed through two different interface styles: SOAP and REST, and in addition to HTTP, the Bittorrent protocol can be used to download content. Every object in S3 has a torrent file associated with it that is available through a GET request that includes the request parameter torrent. The service replies with an HTTP response message containing the torrent file. Upon reception of the torrent file, the client can begin to download the object and make the downloaded portions of it available to other clients. This makes the distribution of information less bandwidth intensive for the S3 service [42].

The adaptation of Bittorrent to fit the requirements of personal storage services is an open issue. As observed in [39] only 40% of the Dropbox home users share at least two folders, which means that Bittorrent may not work well in these environments because of the little simultaneous demand for the same set of objects. There is therefore an obvious need for further research in this area.

4 Roadmap of outcomes and objectives

We organized this roadmap in objectives, deliverables and actions/outcomes. There are a total of 5 main objectives, 17 sub objectives, 19 deliverables, and 23 actions (first year). We only included a detailed roadmap of actions for the first two years. We will update the action list in every year review.

CloudSpaces will devise novel contributions to three main projects:

- Ubuntu One [43] is a mature Personal Cloud with millions of users led by Canonical. Ubuntu One offers advanced services like synchronization, sharing, and streaming among others.
- StackSync [44] is a Personal Cloud system providing syncing and sharing capabilities on top of OpenStack Swift [45]. It is aimed to regional Cloud providers providing professional storage solutions. The first prototype has been developed by URV and TST in the RealCloud spanish project.
- eyeOS [46] is a leading Personal Web Desktop with a huge community spread in more than 50 countries. eyeOS is the leading worldwide provider of open source web desktop software with 20.000 downloads per month and more than 1 million free accounts hosted in its demo server.

By the end of the project, we will demonstrate interoperability between StackSync/OpenStack and Ubuntu One, and complete integration of eyeOS on top of any of them thanks to the open service platform. Furthermore, we will benefit from the huge communities behind these projects to validate the results in open Internet trials, and to disseminate results early.

The outcomes of the project are detailed in the project deliverables. Deliverables in technical packages follow an incremental approach that continuously improves the previous achievements. Furthermore, every deliverable will normally contain contributions from all running tasks inside the WP.

Let us review the major outcomes in every technical WP:

- WP2 (Architecture) has two major outcomes: the CloudSpaces Architecture including all APIs, and the validation in open Internet trials. We will launch three massive Internet trials on top of StackSync, U1, and eyeOS. Captured traces from this real testbeds will help us to continuously improve the system, but are also essential for research partners to produce high impact publications.
- WP3 (Cloudspaces Storage) has one major outcome: the adaptive replication and synchronization infrastructure. On the one hand, this service will permit to aggregate heterogeneous user storage resources and provide adaptive replication, consistency and synchronization that will be optimized for the user perceived latency. Furthermore, the service will also provide adaptivity in terms of proximity-aware synchronization and sharing mechanisms.

- WP4 (CloudSpaces Share) has two important outcomes: the privacy-aware sharing service, and the interoperability between heterogeneous Personal Clouds. The privacy-aware sharing service will provide a set of security mechanisms that will employ hybrid technical approaches for dealing with data leakage and unauthorized data access. This service will also measure and control privacy leakage to several Cloud providers, and it will asses trust of different entities with whom data is shared. Finally, an important functionality to be provided is the syntactic and semantic interoperability between different Personal Clouds. The sharing infrastructure will provide controlled and secure folder sharing with ACLs, but also data export/import, and data search to heterogeneous shared repositories.
- WP5 (CloudSpaces Services) has two major outcomes: the service platform that includes data and persistence APIs, and the proof of concept implementation using eyeOS Personal Web Desktop. On the one hand, the service platform must offer a coherent, integrated, and high-level interface to all platform functionalities. This is the upper stack of CloudSpaces that will be offered to third-party applications. Appropriate documentation and tutorials will be provided along with sample applications. Finally, a complete proof of concept demonstration of the platform will be presented using eyeOS Personal Desktop and tools.

The dependencies between technical WPs are layered and bottom-up. This means that WP4 has clear dependencies with WP3, and that WP5 has dependencies with WP4. For a more detailed explanation of project outcomes, please refer to the descriptions of Deliverables.

4.1 Objectives

These are the five global objectives as stated in the Document of Work:

1. **Scalable data management of heterogeneous resources**. We will develop a mediator data management agent offering advanced storage, sync and share mechanisms thanks to the interconnection of heterogeneous user resources and Cloud remote repositories. To this end, we will devise novel adaptive replication algorithms providing dynamic membership reconfiguration of untrusted repositories as well as advanced consistency mechanisms. Finally, we will also create new proximity-aware data synchronization and sharing techniques benefiting from device location awareness.
2. **Privacy-aware data sharing**. We will implement techniques ensuring secure, trustworthy and privacy friendly interactions within the CloudSpaces environment. To this end, we will propose a set of privacy-aware data sharing mechanisms that will employ hybrid technical approaches like obfuscation, anonymization, encryption, digital signatures, and information hiding. We will also introduce novel mechanisms for privacy-aware data exchange and exposure. In this line, we will estimate privacy leakage and provide feedback to the user regarding her/his actions. Building upon existing approaches on trustworthiness assessment, we will estimate the trustworthiness of the different resources, applications and users in the Cloud environment based on activity logging and data provenance.

3. **Interoperability of Personal Clouds**. We will avoid vendor lock-in thanks to both semantic and syntactic interoperability techniques. We will enable exchange of data and services between users and Personal Clouds with a new semantic framework for creating accurate and adaptive semantic mappings among entities as a negotiation process. We will define a semantic-enriched model that can capture all the metadata related to the various processes and data items in CloudSpaces, covering schema provisions for simple data lineage, declarative storage structures, and metadata enrichment. Finally, regarding syntactic interoperability, we will define and implement a standard way of exporting all data stored in a Personal Cloud. We will also enable data exchange and sharing among heterogeneous Personal Clouds.

4. **Standard Service front-end**. We will create a service framework with standard APIs enabling the interoperability of third-party applications with the data contained in Personal and shared Clouds. We will provide standard data management services (Store, Sync, Share) and an advanced persistence service offering Key/Value Store Semantics, advanced queries and consistency and synchronization with the Personal Cloud resources. All these services will be offered in a clear service framework promoting adoption and third-party development. Finally, we will demonstrate the capabilities of the aforementioned services with an advanced proof of concept: the eyeOS Personal Web Desktop.

5. **Validate and disseminate the platform contributing to open source projects**. CloudSpaces will mainly contribute to three open source projects. It will extend OpenStack Swift to include advanced scalable data management algorithms. It will extend Ubuntu One to incorporate interoperability and privacy mechanisms. And it will extend eyeOS to demonstrate the functionalities of CloudSpaces Services. We will also leverage the massive user communities of these open source projects like OpenStack, Ubuntu, and eyeOS to disseminate the results of the project and to validate the platform in real Internet scenarios with thousands of users.

And there are 17 subobjectives as we can see in the following table.

Objective No	Objective Name	WP	Task	Global Objective
1	Design the overall system architecture of the CloudSpaces toolkit	WP2	T2.1	4
2	Identify the set of interfaces and APIs, representing the services required on the Personal Cloud infrastructure	WP2	T2.1	4
3	Implement and test open source prototypes to obtain early feedback	WP2	T2.2	5
4	Describe stress-test Scenarios and Benchmarking Framework	WP2	T2.3	5
5	Explore novel adaptive replication and synchronization schemes dealing with aspects like load, failures, network heterogeneity and desired consistency levels. The benefits of these schemes will consist in optimizing for the user perceived latency	WP3	T3.1, T3.2	1

6	Extend the previously obtained replication-oriented storage protocols to erasure coded protocols in order to optimize the space consumption at individual data repositories	WP3	T3.1	1
7	Develop a mediator data management agent that may be deployed in home gateways, NAS devices, or private user resources	WP3	T3.1	1
8	Proximity Aware Sync and Share Mechanisms: Investigate data synchronization and distribution techniques that can be improved if devices are geographically close to one another	WP3	T3.3	1
9	Create a privacy-aware sharing infrastructure enabling the secure and controlled access to Personal Cloud resources from users and applications	WP4	T4.1	2
10	Design a set of ontologies that enable the interoperability of heterogeneous personal data	WP4	T4.2	3
11	Establish an overall metadata management framework that integrates, enriches, maintains, and queries metadata information of individuals in CloudSpaces	WP4	T4.2	3
12	Implement techniques ensuring secure, trustworthy and privacy friendly interactions within the CloudSpaces environment	WP4	T4.1	2
13	Exporting data from the Personal Cloud: Define and implement a standard way of exporting all data stored in a Personal Cloud	WP4	T4.3	3
O14	Data Service. The first goal is to define a service framework with standard APIs enabling the interoperability of third-party applications with the data contained in Personal and Shared Clouds	WP5	T5.1	4
O15	Persistence Service. This persistence service should provide at least Key/Value Store Semantics, advanced queries and consistency and synchronization with the Personal Cloud resources	WP5	T5.2	4
O16	Integration of Personal Clouds and Web and Mobile technologies. We will evaluate and adapt several Web standards and technologies to the Personal Cloud service platform	WP5	T5.1, T5.2, T5.3	4
O17	Integration of the eyeOS Personal Web Desktop on top of CloudSpaces. eyeOS must be the proof-of-concept application that demonstrates the potential capabilities of the CloudSpaces Service platform	WP5	T5.3	4

4.2 Deliverables

The project includes a number of Deliverables as described in the DoW that are in fact the coarse grained "Roadmap of outcomes" of this project.

Deliverable No	Deliverable Title	WP No	Lead beneficiary name	Estimated indicative person-months	Nature	Dissemination level	Delivery month
D1.1	Management Plan	1	URV	3.00	R	PU	3
D2.1	Roadmap of target outcomes	2	URV	6.00	R	PU	8
D2.2	Draft architecture specifications, use case scenarios and benchmarking framework	2	TST	12.00	R	PU	12
D2.3	Validation and Feedback analysis from open Internet trials	2	URV	24.00	R	PU	24
D2.4	Reference implementation of architectural building blocks	2	URV	35.00	P	PU	34
D3.1	Guidelines for heterogeneous Personal Clouds	3	EUR	20.00	R	PU	12
D3.2	Adaptive storage infrastructure	3	EUR	50.00	R	PU	24
D3.3	Final results and software release	3	EUR	37.00	P	PU	33
D4.1	Guidelines on Privacy-aware data-sharing	4	EPFL	16.00	R	PU	12
D4.2	Privacy-aware sharing infrastructure	4	EPFL	40.00	P	PU	24
D4.3	Final release of privacy and interoperability components	4	EPFL	40.00	P	PU	33
D5.1	Framework specs and API descriptions	5	CNC	12.00	R	PU	12
D5.2	Service Platform reference prototype	5	EOS	40.00	P	PU	24
D5.3	Final software release of the service platform	5	EOS	44.00	P	PU	36
D6.1	First version of the Exploitation Plan	6	CNC	7.00	R	PU	6
D6.2	Communication plan for dissemination	6	CNC	11.00	R	PU	12
D6.3	Community involvement, exploitation, and dissemination report	6	CNC	11.00	R	PU	24
D6.4	International Workshop with Personal Cloud Providers	6	CNC	1.00	O	PU	24
D6.5	Final dissemination, exploitation, and standardization report	6	CNC	8.00	R	PU	35
	Total			**417.00**			

R = Report; P = Prototype; D = Demonstrator; O = Other
PU = Public; PP = Restricted to other programme participants (including the Commission Services); RE = Restricted to a group specified by the consortium (including the Commission Services); CO = Confidential, only for members of the consortium (including the Commission Services)

As requested by reviewers we have included a more detailed and fine grained roadmap of outcomes including concrete development actions for the first year.

4.3 Actions/Outcomes for the first year

We have defined 23 actions/outcomes to be completed for the first year. We have included in this action list the seven deliverables of the first year from work packages WP2, WP3, WP4, WP5 and WP6.

Action No	Action Title	Description	Lead beneficiary name	Deliverable No	Month
A1	Roadmap of target outcomes	Roadmap of outcomes and objectives. Detailed definition of the Personal Cloud paradigm, its relationships with Personal Computing, and it will explain how the next generation of Personal Clouds will provide advanced services like Interoperability and privacy. Roadmap of outcomes, project dependencies and partner roles.	URV, all	D2.1	6
A2	Validation platform and deployment of testbed infrastructures	Acquisition, installation and deployment of new server equipment in TISSAT, CNC, EOS and URV. Justify the need for new hardware. This task will be documented and described in D2.2. Access to the testbed (proxies) should be granted to the rest of partners.	TST	D2.2	6
A3	Generation of traces from U1 and eyeOS	Acquisition, filtering and anonymization of traces following the format specified by URV in D2.2. Acquisition mechanisms will be automated for future traces. This task is very important for the research of academic partners.	CNC, EOS	D2.1	6
A4	Early prototype of Adaptive Personal Storage. a) Design; b) Early prototype with StackSync integration	We will explore novel adaptive replication and synchronization schemes dealing with aspects like load, failures, network heterogeneity and —desired consistency levels. The benefits of these schemes will consist in optimizing for the user perceived latency and reliability. Integration in StackSync.	EUR, URV	D2.2, D3.1	7 (a), 11 (b)
A5	Early prototype of Adaptive Cloud Storage. a) Design; b) Early prototype with StackSync integration	We will devise novel Cloud-provided seed swarming schemes to help Personal Clouds address increased QoS requirements like scalability and content distribution. Integration of BitTorrent with StackSync/OpenStack Swift.	URV	D2.2, D3.1	7 (a), 11 (b)
A6	Early prototype of Privacy-aware sharing service. a) Design; b) Early prototype with StackSync integration	Controls privacy leakage through Privacy Enhancing Technologies, i.e. data obfuscation, data slicing, data hiding, etc. Consider technologies like Attribute-based encryption (ABE) for the secure sharing service. Integration with StackSync.	EPFL, URV	D2.2, D4.1	7 (a), 11 (b)
A7	Design of Syntactic APIs	Public docs and spec of Share (token) and Store APIs	CNC	D2.2, D4.1	7

A8	Persistence service documentation and releases. a) web storage; b) rest	Evolution, documentation and maturity of U1DB. Early release of Web persistence for M6-8	CNC	D5.1	8 (a), 11 (b)
A9	eyeFiles connection with Store API (U1)	Integration of eyeFiles component with Store U1 API. Development of eyeOS bus decoupled architecture.	EOS	D2.2 (Design), D5.1	8
A10	Design of secure sharing with ACLs	Establishes privacy policies against different stakeholders and checks that data access (and ACLs) is conformant to them through logging. Design of ACL mechanisms for external users and applications.	EPFL, URV	D2.2, D4.1	11
A11	Semantic ontologies for Personal Clouds	Definition of semantic ontologies for Personal Clouds.	EPFL	D4.1	11
A12	Proximity-aware Service Design	State of the art, use case scenarios, and design document specifying Proximity-aware services.	CNC	D2.2, D3.1	11
A13	Early implementation of Store and Share APIs	Implementation of prototype Store and Share APIs in U1 and StackSync/OpenStack	CNC, URV, TST	D2.2. D4.1	11
A14	eyeCalendar integration with U1DB	Integration of eyeCalendar component with U1DB Persistence service (Web localStorage)	EOS	D2.2 (Design), D5.1	11
A15	Public Release of StackSync Cloud testbed with real users	Public announcement to TST users of StackSync open community testbed. This will require the development of monitoring components for trace capturing, load balancing features in the Swift Proxy, and initial Web management interfaces for StackSync.	TST	D2.2, D5.1	11
A16	Communication plan	Definition of the required process and strategy for dissemination activities. Explain early results of community involvement activities.	URV, CNC, All	D6.2	12
A17	Draft architecture specifications, use case scenarios and benchmarking framework	Architecture specifications of the CloudSpaces toolkit. First draft of platform APIs: Store, Proximity, Share, Export, Privacy, Logging, Persistence Use cases, demonstration scenarios. First analysis of U1 and TST traces. Tool and integration specs for the eyeOS platform	URV, All	D2.2	12
A18	Guidelines for heterogeneous Personal Clouds	Guidelines on adaptive distributed synchronization and replication schemes for heterogeneous Personal Clouds. Early prototype of the Adaptive edge platform (adaptors,mediator). Deployment and management of the Adaptive cloud platform testbed (Openstack Swift, massive testbed, first user trials and distributed testing) Architecture, APIS, and use cases of Adaptive Proximity-aware services	EUR, URV, TST, CNC	D3.1	12

A19	Guidelines on Privacy-aware data-sharing	Preliminary design of privacy trustworthy data sharing framework and draft API. Initial privacy leakage metrics, initial trustworthiness assessment mechanism, initial privacy enhancing technologies against simple threat models. Preliminary design of data logging infrastructure. Share syntactic draft APIs and abstractions including ACLs. Personal information ontology specification. Early incomplete prototype of privacy-aware sharing component	EPFL, URV, CNC, TST	D4.1	12
A20	Framework specs and API descriptions	This report will include guidelines, open specifications, and documented best-practices for achieving syntactic interoperability of Personal Clouds. This includes both data exporting, data sharing, and application interconnection to Personal Cloud Services. Data API specs and early prototypes. Persistence API spec and early prototypes. eyeOS early incomplete integration with data services including eyeSync and eyeFiles. Candidate tools specs for eyeOS demonstrator tools demonstrating contact, file, and calendar data use. Early prototype of OpenStack Web data management integration with data services. All software will be released under an open source license such as GPLv3.	CNC, EOS, TST, URV	D5.1	12
A21	First version of the Exploitation Plan	Clear definition of the "end-results/products" Preliminary view on the market situation and the project's positioning SWOT analysis Preliminary exploitation strategy	EOS, TST, CNC	D6.1	12
A22	Communication plan for dissemination	Definition of the required process and strategy for dissemination activities. Explain early results of community involvement activities.	URV, all	D6.2	12

As we can see in the table, we define the action, but we also include the Deliverable where it contributes, the objective, the target date, and the responsible or responsible partners.

During the **first year**, academic partners are focusing on adaptive storage and privacy, and industrial partners are mainly targeting syntactic interoperability.

In the first semester, industrial partners must set up their testbed infrastructures as well as provide the traces from their systems (U1, eyeOS). In the first year review they will show both horizontal interoperability between Ubuntu One and StackSync (Store & Share), and vertical interoperability between eyeOS and U1 (Sync, Store, Persistence) and StackSync (Sync & Store). Furthermore, TISSAT will launch in M12 a live testbed with real users of StackSync/OpenStack.

In the first semester, academic partners will present their design and specification of the early prototypes to be presented in the first year review. Eurecom and URV will focus their developments on Adaptive Storage over StackSync/OpenStack. In the first year review, Eurecom will present an early prototype over StackSync improving reliability and latency of Personal Storage of home users with untrusted heterogeneous resources. URV will devote

resources to adaptive personal storage in the data center using Cloud swarming technologies. URV will present an early prototype integrating BitTorrent content distribution with StackSync/OpenStack in an adaptive way.

EPFL with the collaboration of URV will focus on privacy-aware sharing using Privacy Enhancing Technologies. Benefiting from context awareness, privacy risk functions, and encryption techniques, EPFL will present an early prototype in the first year review, showing the sharing of encrypted data between different users of Personal Clouds. This prototype will be integrated in the StackSync/OpenStack platform.

4.3.1 Dependences among actions

We outline four dependences between actions in the first year:

1. Actions A9, A13, and A15 depend on the correct delivery of Action A2 (Validation platform). Two integration efforts will be done during the first year that require public testbeds: eyeOS tools (eyeFiles, eyeCalendar) with U1, and between StackSync and U1 (Store and Share).

2. Action A22 (Communication plan) depends on Action A3 (Generation of traces). Traces are extremely important for research partners to obtain good research results during the project.

3. Action A13 (Early implementation of Store and Share APIs) depends on Action A7 (Design of Syntactic APIs). An initial design is required in M8 to implement first prototypes for M11.

4. Action A14 (eyeCalendar integration with U1DB) depends on action A8 (Persistence service documentation and releases). The U1DB web store implementation in M8 is needed for the eyeCalendar integration in M11

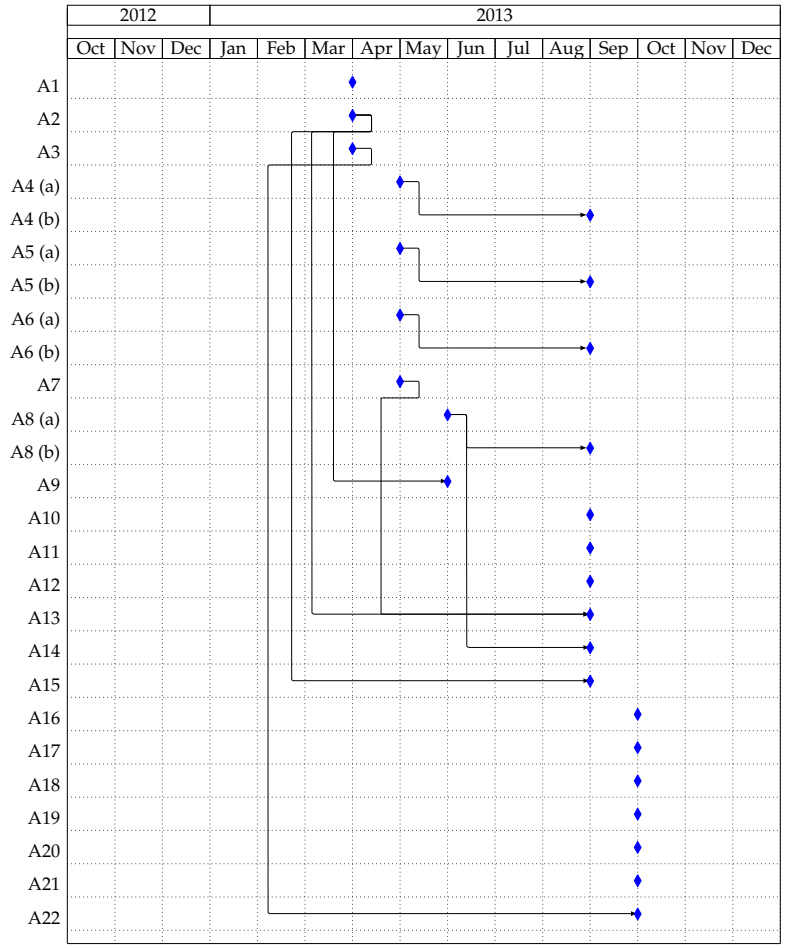

Figure 1: Dependencies among actions for the first year

4.4 Actions/Outcomes for the second year

We defined 11 actions for the **second year** instead of the 23 actions of the first year. These actions are high level and try to set a path that we can adapt after the first results and feedback of the first year review. In any case, the second year is focused on consolidation and maturing of results and on user involvement and testing.

Action No	Action Title	Description	Lead beneficiary name	Deliverable No	Month
B1	Public release of Privacy-aware sharing in StackSync	Open Source release of the privacy-aware sharing component (StackSync). First support for Access Control Lists (ACLS). Integration with sharing syntactic interoperability services (StackSync with U1).	EPFL, URV, TST	D4.2	18
B2	Public release of HPDC Adaptive Storage service	Open Source release of the adaptive storage components in the StackSync client. This includes hybrid cloud storage services, as well as the integration with content distribution technologies (BitTorrent). Prototype proximity-aware services included.	EUR, URV	D3.2	18

B3	Public release of service platform APIs	Open Source release of the reference implementation of the service platform. It includes Store and Share APIs specs and conformance tests. It will also include a first prototype of persistence services.	CNC, TST, URV	D5.2	18
B4	Public release of eyeOS collaboration tools and open testbed	Open Source release of eyeOS collaboration platform on top of the service platform (U1, StackSync). It will provide a workspace component offering access control and fine grained data sharing. It will also provide a prototype collaborative editor and a running version of the calendar tool over the U1DB Web persistence interfaces.	EOS	D5.2	18
B5	Launch of User communities on top of open testbeds	Once public prototypes are released, user communities will be created by invitation to evaluate the system and provide feedback.	EOS, TST, CNC, URV	D6.3	18
B6	Validation and Feedback analysis from open Internet trials	Complete analysis of traces from real Internet testbeds. This will include analysis ftom the three trials: TISSAT datacenter (Desired number of users: 1000s). The goal is to attract early adopters (SMEs, professional users, end-users) to help us test and validate the system. Beginning the First year of the project. U1: some features will be delivered in the official U1 clients, while others may be published in limited prototypes. Desired number of users (>1000). Beginning the second year of the project. EOS: real testbed using Cloudspaces eyeOS apps and achievements with its own community. Desired number of users: >1000. Beginning second year of the project. Validation of results and interoperability tests between the three communities.		D2.3	24
B7	Adaptive storage infrastructure	Report on distributed erasure-coding techniques and handling untrusted data repositories. Prototype of adaptive storage platform. Adaptive cloud platform services for personal clouds (OpenStack, swarm-based data distribution). Public release of Proximity-aware Prototypes and APIs. This software will be released under an open source license such as GPLv3.		D3.2	24
B8	Privacy-aware sharing infrastructure	Final design of privacy trustworthy data sharing framework and final API. Final design of data logging infrastructure. Full prototype of privacy-aware sharing component. Public release of Share and export APIs. Early design of semantic mapping mechanisms. All software will be released under an open source license such as GPLv3.		D4.2	24

B9	Service Platform reference prototype	The prototype will be published along with documentation, tutorials, and conformance tests. A prototype framework will be released offering these APIs. Validated APIs on top of U1 and OpenStack. Persistence service and tutorials. Integration of eyeOs and platform services. Public release of three eyeOS proof-of concept tools. Public release of OpenStack web management services. All software will be released under an open source license such as GPLv3.		D5.2	24
B10	Community involvement, exploitation, and dissemination report	Description of the Year 2 dissemination and collaboration activities with lessons learned and progress reporting. Explain results of community involvement activities.		D6.3	24
B11	International Workshop with Personal Cloud Providers	With the lead of the U1 team, we will invite to an interoperability workshop a large list of Personal Cloud providers. We will try to set up a working group between providers to agree on common APIs and standards. Description of the workshop activities and presentations.		D6.4	24

We established six actions (B6-B11) for the deliverables of the second year (M24) and 5 intermediate actions (B1-B5) for M18.

Actions B1 to B4 refer to public software releases of open source components. Consolidated prototypes will be published in privacy-aware sharing with fine-grained access control, adaptive personal storage, vertical and horizontal interoperability, and new collaborative proof of concept tools (workspace). These releases are the evolution of the prototypes presented in the first year review.

Just after these software releases, we will launch the user group by invitation (action B5). The CloudSpaces user group will be created with communities from industrial and academic partners. Users will begin to evaluate and use the prototypes in open Internet trials.

The second year must finish with more stable releases of open source components, analysis of results from open Internet trials, and finally an International Workshop with major Personal Cloud Providers to disseminate the results of the project. The workshop will also try to reach consensus in interoperability issues.

We will not delve into the actions of the third year, since they will defined later on during the project.

4.4.1 Dependences among actions

We outline four dependences for the second year:

1. Action B1 (First public software release of Privacy-aware sharing in StackSync) depends on actions A6 (Early prototype of Privacy-aware sharing service) and A13 (Early implementation of Store and Share APIs).

2. Action B2 (First public software release of HPDC Adaptive Storage service) depends on actions A4 (Early prototype of Adaptive Personal Storage) and A5 (Early prototype of Adaptive Cloud Storage).

3. Action B3 (First public release of service platform APIs) depends on action A13 (Early implementation of Store and Share APIs).

4. Action B4 (First public release of eyeOS collaboration tools and open testbed) depends on actions A9 (eyeFiles connection with Store API), A13 (Early implementation of Store and Share APIs) and A14 (eyeCalendar integration with U1DB).

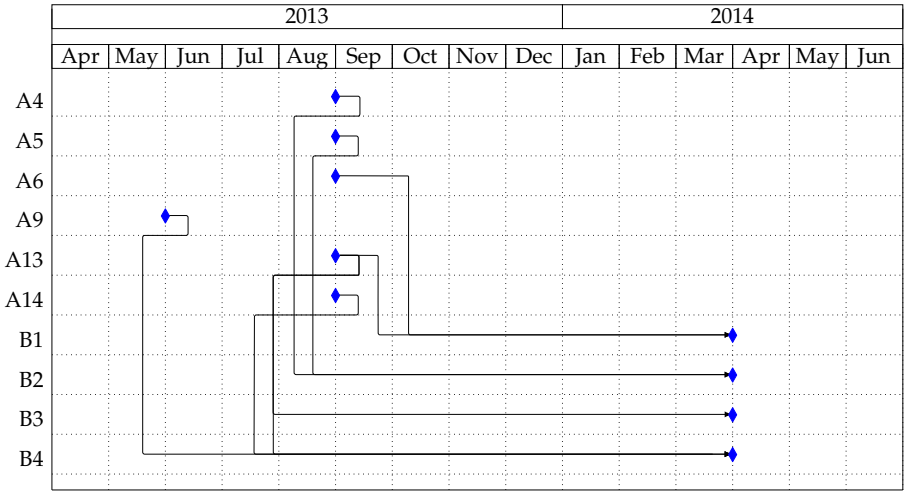

Figure 2: Dependencies among actions for the second year

Since these dependencies can become a risk for some outcomes, we will analyze contingency plans every 6 months in our project meetings.

4.5 Software releases

CloudSpaces will follow a time-based software release cycle, rather than a feature-driven one. This approach —currently followed by major open source projects like Ubuntu or OpenStack— will help us integrate and aggregate contributions from different sources in a more efficient way.

Furthermore, it is aimed to attract external users and developers to the project. We will plan the contents of each release in the CloudSpaces Design summit that will take place every six months.

We outline here the two first software releases:

1. Release 1 (M12):
    - Public Release of StackSync software and open testbed (A15)
    - Early prototype of Adaptive Cloud Storage (A5)
    - Early prototype of Privacy-aware sharing service (A6)
    - Early implementation of Store and Share APIs (A13)
    - eyeOS Personal Cloud Release (A9,A14)
2. Release 2 (M18):
    - Public release of Privacy-aware sharing in StackSync (B1)
    - Public release of HPDC Adaptive Storage service (B2)
    - Public release of service platform APIs (B3)
    - Public release of eyeOS collaboration tools and open testbed (B4)

References

[1] "Forrester, the personal cloud: Transforming personal computing, mobile, and web markets," http://t.co/DAXweKQw.

[2] "Respect network, from personal computers to personal clouds: The birth of the cloud os," http://respectnetwork.com/personal-clouds/.

[3] "Pc magazine encyclopedia," http://www.pcmag.com/encyclopedia/.

[4] "Openi project and cloudlets," http://www.openi-ict.eu.

[5] Cubby, "Directsync," http://help.cubby.com/knowledgebase/articles/143893-what-s-directsync-sync-without-the-cloud-.

[6] Dropbox, "Lan sync," https://www.dropbox.com/help/137/en.

[7] "Oauth," http://oauth.net/.

[8] S. Perez, "Finally! bitcasa ceo explains how the encryption works," http://techcrunch.com/2011/09/18/bitcasa-explains-encryption/.

[9] R. David Graham, "Mega and encrypted cloud deduplication," http://erratasec.blogspot.ch/2013/01/mega-and-encrypted-cloud-deduplication.html.

[10] A. Fairless, "Why spideroak doesn't de-duplicate data across users (and why it should worry you if we did)," https://spideroak.com/blog/20100827150530-why-spideroak-doesnt-de-duplicate-data-across\-users-and-why-it-should-worry-you-if-we-did.

[11] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), ser. ICDCS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 617–.

[12] S. Kamara and K. Lauter, "Cryptographic cloud storage," in Proceedings of the 14th international conference on Financial cryptograpy and data security, ser. FC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 136–149. [Online]. Available: http://dl.acm.org/citation.cfm?id=1894863.1894876

[13] C. K. Liew, U. J. Choi, and C. J. Liew, "A data distortion by probability distribution," ACM Trans. Database Syst., vol. 10, no. 3, pp. 395–411, Sep. 1985. [Online]. Available: http://doi.acm.org/10.1145/3979.4017

[14] S. Fienberg and J. McIntyre, "Data swapping: Variations on a theme by dalenius and reiss," in Privacy in Statistical Databases, ser. Lecture Notes in Computer Science, J. Domingo-Ferrer and V. Torra, Eds. Springer Berlin Heidelberg, 2004, vol. 3050, pp. 14–29. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-25955-8_2

[15] L. Sweeney, "k-anonymity: a model for protecting privacy," Int. J. Uncertain. Fuzziness Knowl.-Based Syst., vol. 10, no. 5, pp. 557–570, Oct. 2002. [Online]. Available: http://dx.doi.org/10.1142/S0218488502001648

[16] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," ACM Trans. Knowl. Discov. Data, vol. 1, no. 1, Mar. 2007. [Online]. Available: http://doi.acm.org/10.1145/1217299.1217302

[17] N. Li and T. Li, "t-closeness: Privacy beyond k-anonymity and l-diversity," in In Proc. of IEEE 23rd Int'l Conf. on Data Engineering (ICDE'07), 2007.

[18] L. Zou, L. Chen, and M. T. Özsu, "k-automorphism: a general framework for privacy preserving network publication," Proc. VLDB Endow., vol. 2, no. 1, pp. 946–957, Aug. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1687627.1687734

[19] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," SIGKDD Explorations, vol. 4, p. 2002, 2002.

[20] C. Gentry, "Fully homomorphic encryption using ideal lattices," in Proceedings of the 41st annual ACM symposium on Theory of computing, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[21] M. Sintek, S. Handschuh, S. Scerri, and L. van Elst, "Technologies for the social semantic desktop," in Reasoning Web, 2009, pp. 222–254.

[22] Enisa, "Cloud computing risk assessment," http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment.

[23] "Openid connect," http://openid.net/connect/.

[24] "User managed access (kantara inititative)," http://kantarainitiative.org/confluence/display/uma/Home.

[25] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," The VLDB Journal, vol. 10, no. 4, pp. 334–350, Dec. 2001. [Online]. Available: http://dx.doi.org/10.1007/s007780100057

[26] P. Cudré-Mauroux, "Emergent semantics: rethinking interoperability for large scale decentralized information systems," Ph.D. dissertation, EPFL, 2006.

[27] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth, "Start making sense: The chatty web approach for global semantic agreements," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 1, no. 1, pp. 89–114, 2003.

[28] A. Baronchelli, V. Loreto, and L. Steels, "In-depth analysis of the naming game dynamics: the homogeneous mixing case," International Journal of Modern Physics C, vol. 19, no. 05, pp. 785–812, 2008.

[29] K. Aberer, P. Cudré-Mauroux, A. Ouksel, T. Catarci, M.-S. Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. Neuhold et al., "Emergent semantics principles and issues," in Database Systems for Advanced Applications. Springer, 2004, pp. 25–38.

[30] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler, "Wsmx-a semantic service-oriented architecture," in Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on. IEEE, 2005, pp. 321–328.

[31] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

[32] J. Whiteaker, F. Schneider, R. Teixeira, C. Diot, A. Soule, F. Picconi, and M. May, "Expanding home services with advanced gateways," Computer Communication Review, vol. 42, no. 5, pp. 37–43, 2012.

[33] O. Riva, Q. Yin, D. Juric, E. Ucan, and T. Roscoe, "Policy expressivity in the anzere personal cloud," in Proceedings of the 2nd ACM Symposium on Cloud Computing, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 14:1–14:14. [Online]. Available: http://doi.acm.org/10.1145/2038916.2038930

[34] R. Sharma, A. Datta, M. Dell Amico, and P. Michiardi, "An empirical study of availability in friend-to-friend storage systems," in P2P 2011, IEEE International Conference on Peer-to-Peer Computing, August 31-September 2nd, 2011, Kyoto, Japan, Kyoto, JAPAN, 08 2011. [Online]. Available: http://www.eurecom.fr/publication/3452

[35] R. G. Tinedo, M. S. Artigas, A. Moreno-Martínez, and P. G. López, "Friendbox: A hybrid f2f personal storage application," in IEEE CLOUD, 2012, pp. 131–138.

[36] B. Salmon, S. W. Schlosser, L. F. Cranor, and G. R. Ganger, "Perspective: semantic data management for the home," in Proccedings of the 7th conference on File and storage technologies, ser. FAST '09. Berkeley, CA, USA: USENIX Association, 2009, pp. 167–182. [Online]. Available: http://dl.acm.org/citation.cfm?id=1525908.1525921

[37] J. Strauss, C. Lesniewski-Laas, J. M. Paluska, B. Ford, R. Morris, and F. Kaashoek, "Device transparency: a new model for mobile storage," SIGOPS Oper. Syst. Rev., vol. 44, no. 1, pp. 5–9, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1740390.1740393

[38] e. 3rd and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)," RFC 4634 (Informational), Jul. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4634.txt

[39] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: understanding personal cloud storage services," in Proceedings of the 2012 ACM conference on Internet measurement conference, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 481–494. [Online]. Available: http://doi.acm.org/10.1145/2398776.2398827

[40] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 68–73, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496103

[41] "Bittorrent," http://www.bittorrent.com.

[42] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon s3 for science grids: a viable solution?" in Proceedings of the 2008 international workshop on Data-aware distributed computing, ser. DADC '08. New York, NY, USA: ACM, 2008, pp. 55–64. [Online]. Available: http://doi.acm.org/10.1145/1383519.1383526

[43] "Ubuntu one personal cloud," https://one.ubuntu.com.

[44] "Stacksync," http://ast-deim.urv.cat/stacksync.

[45] "Openstack open source cloud middleware," http://openstack.org.

[46] "eyeos personal web desktop," http://eyeos.org.