Cloud Storage and Peer-to-Peer Storage
End-user considerations and product overview

Project : Gigaport3
Project year : 2010
Coordination : Rogier Spoor
Author(s) : Arjan Peddemors
Due date : Q2 2010
Version : 1.1

Summary

Recent years have seen a strong increase in products and services that support end users to store and backup their data over the Internet. When data is stored at server clusters within the Internet, this kind of data storage is referred to as *cloud storage*. When relying on members of a group storing each other's data, it is called *peer-to-peer (P2P) storage*. Contrary to cloud storage services, P2P storage products are typically not subscription based, but rather depend on group members that are part of a peer network to trade resources, primarily disk capacity and network bandwidth. This document provides a description of general cloud storage and P2P storage principles, and presents an overview of currently available cloud- and P2P storage products and services that may be of interest to SURFnet end users (researchers, staff, and students). The product overview is a scan of existing products and their applicability, not a full evaluation of those products (i.e., we have not executed test runs in real environments). Although cloud storage and especially P2P storage products are relatively new and have a limited track record in terms of reliability and security (and therefore must be considered between moderately mature and experimental), their unique properties may address storage needs of SURFnet end users well. This document provides support for those SURFnet end users that consider using a cloud- or P2P storage product to backup data, share files with others, or synchronize files between multiple computers.

Colofon

Program:	Gigaport3
Workpackage:	WP3
Activity:	Storage Clouds
Deliverable:	Storage Clouds
Access rights:	Public
External party:	Novay, http://www.novay.nl/

This project was made possible by the support of SURF, the collaborative organisation for higher education institutes and research institutes aimed at breakthrough innovations in ICT. More information on SURF is available on the website www.surf.nl.

Contents

1 Introduction

Computer users rely on the safe and efficient storage of their data in computer files. Traditionally, computer data storage is organized as a storage hierarchy, where data, when not in main memory (primary storage), is first stored on disk drives and internal flash memory (secondary storage) and later transferred to tape, optical disks, external flash memory or a similar medium for backup (tertiary storage). Primary storage is fast but has limited capacity, while – on the other end of the spectrum – tertiary storage combines long access times with large capacity. Contrary to primary storage, secondary and tertiary storage are non-volatile (i.e., do not require power to maintain the stored information). By combining all types of storage in a computing system, a large amount of data can be accessed by applications and at the same time be safely stored for a long time.

Both secondary and tertiary storage may be realized using a fast network. Data transfer for remote storage and retrieval of many types of digital objects has been a long time Internet service. Protocols for file transfer (e.g., FTP, SCP), as well as protocols for the establishment of network file systems (e.g., NFS, SMB), are in widespread use. Protocols supporting network file systems are, however, considerably less suitable to support secondary storage in a wide area network setting (such as the larger Internet) due to restrictions in throughput and latency, and are therefore only applied in local area networks. They typically assume a centralized model, where data is stored and synchronized (amongst multiple clients) at a single location.

Due to decreasing costs, hard disks become more and more attractive to use not only as secondary storage, but also as backup medium in tertiary storage. Compared to tape, they have short access times and high read and write speed. To ensure durability, it is important that any data carrier (hard disks, tape, etc.) used for backup is not at the same location as the original data.

1.1 Cloud Storage Services

A recent trend is to implement tertiary storage or combined secondary/tertiary storage as an Internet service – a so called *cloud storage* service. Such a service is offered by software running on a collection of servers, with data from client machines stored at the hard disks of multiple server nodes (see Figure 1). Typically, a *cloud storage process* on a client node transfers (part of) the data available in local storage back and forth to an entry point of the cloud storage service. This entry point makes sure that the data from the client is distributed over other server nodes. The cloud storage process keeps local data synchronized with data stored at the cloud storage service: new data generated locally by the user is uploaded to 'the cloud', data is retrieved from 'the cloud' when local data was lost.

Examples that offer core functionality for cloud storage are Amazon S3 [3] and Microsoft Live SkyDrive [21]. One advantage of cloud storage backup, contrary to a local backup to a medium like optical disk, is that the data is automatically geographically dispersed (which reduces the risk of losing data). Another advantage is that cloud storage often offers additional features such as synchronization of the data between multiple computers or the sharing of data with others. An example of such a service with enhanced features is Dropbox [11] (based on Amazon S3). Typically, it keeps a copy of often-used files on the local computer and synchronizes data with the storage servers in the background. In that way, file operations such as reading and writing is fast because it operates on local disk storage only. Unless storing a small quantity of data, cloud storage services are paid services, where the user pays, for instance, per amount of data kept in storage and the amount of network traffic generated to upload or download files.

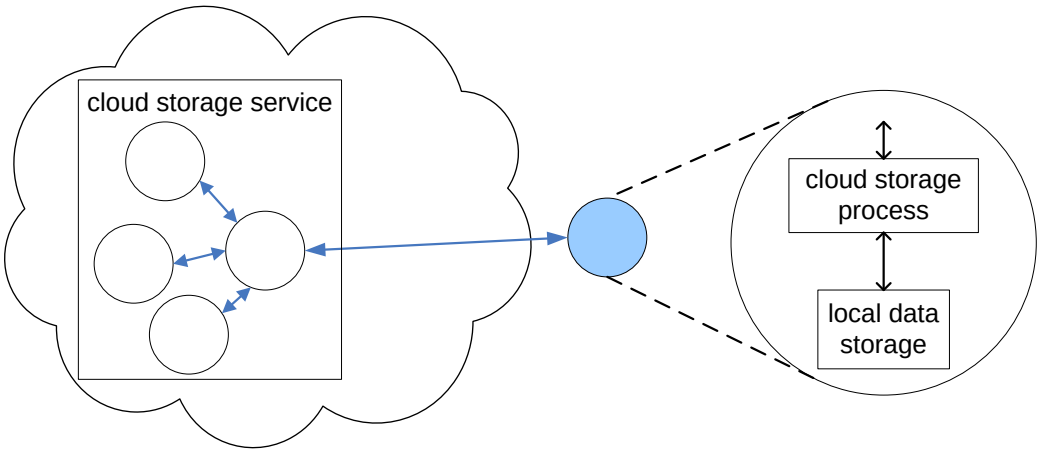

Figure 1: a node connected to the Internet stores data at a cloud storage services, using a cloud storage process. The cloud storage service distributes received data over multiple servers.

1.2 Peer-to-Peer Storage Services

Sharing of resources between nodes on the Internet is currently a wide spread phenomenon. Nodes participate in a distributed network to share CPU cycles, disk storage or network bandwidth, while requiring limited coordination of a central authority. These nodes use resources of other nodes and vice versa, and together form a *peer-to-peer* (P2P) network. P2P mechanisms are applied on a large scale for the one-off sharing of files between Internet users; they are, for instance, part of the Gnutella [14] and the BitTorrent [6] file sharing protocols.

An opportunity exists to realize a distributed storage facility without the need for a paid subscription, but rather trade local resources (disk space, network bandwidth) to store data at peers, using P2P technologies. A large number of research projects addressed this topic and currently a number of relatively new products exist that allow users to collaboratively create a *P2P storage facility*. From the perspective of the user, the cost of sharing existing resources with peers may compare favorable to the repeated monetary cost of a storage subscription. Especially when these resources are abundant and of high quality, as is the case for the network connectivity of many SURFnet participants and the general availability of low-cost hard disks, P2P storage may be a viable and attractive alternative to both traditional storage methods as well as cloud storage services.

P2P storage works as follows. Nodes on the Internet cooperate in an overlay network consisting of peers, where each peer is both a consumer and a producer of data. Peers want to store data in the peer network for various reasons: they may want to backup local data, keep files synchronized between different machines, or share a part of their data with others. From the perspective of a single node in a P2P storage network, a *P2P storage process* exists that replicates local data to peers and in return receives data from peers to store for later use (by these peers). This is depicted in Figure 2. The local data of a peer may be stored in a redundant fashion at other peers, such that, when one or a few peers leave, the data is highly likely to still be available at one of the remaining peers. This same principle is also applied in the back-end of cloud storage services: by redundantly storing user data on server nodes, no data is lost when one of the servers becomes unavailable.

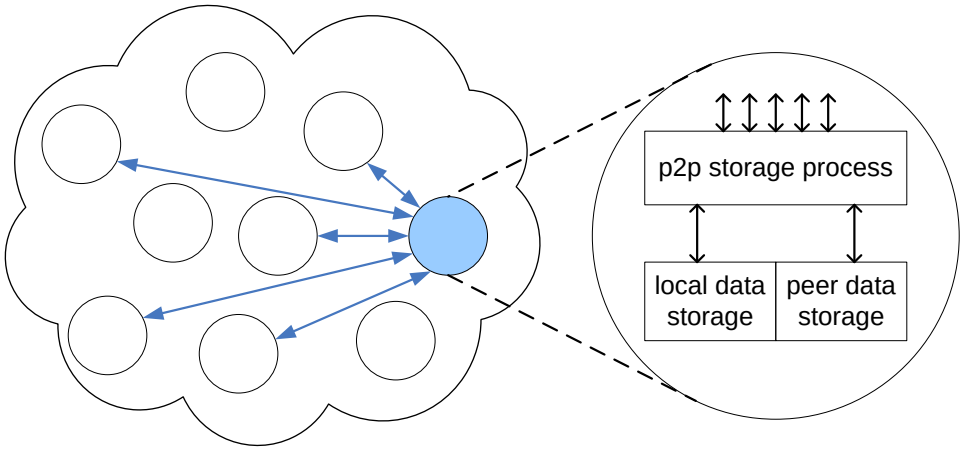

Figure 2: peers realize a distributed storage facility on the Internet, where every node redundantly replicates local storage to peers and in return stores data of peers (governed by a P2P storage process)

Both cloud storage and P2P storage benefit from fast Internet connectivity. Like the capacity of hard disks, the speed offered by Internet Service Providers (ISPs) to end user subscribers has increased exponentially, a trend that is likely to continue in the foreseeable future with the advent of technologies such as fiber optic cables to homes (FTTH).

These technological developments are a driving force behind the rise of cloud- and P2P storage services, and bring them in reach of a large group of consumers. It is likely, as with the rise of cloud computing in general, that storage of data in the cloud – in one form or another – will be used by a large group of people in the future.

1.3 Document Objectives, Scope and Outline

The *main objective* of this report is to present an overview of existing cloud storage and P2P storage concepts and products that may be useful to researchers, staff and students (end users) of the participants of SURFnet. As such, it provides support for those SURFnet end users that consider selecting and using a cloud- or P2P storage product to backup data, share files with others, or synchronize files between multiple computers.

In order to reach this objective, we present a *product scan* of cloud/P2P storage products, together with a classification of their functionality. In addition, we discuss a specific *scenario* – synchronization and backup – to highlight features and characteristics that are relevant for SURFnet end users. We assume the reader has basic knowledge about computer operation, file systems, and Internet data communication. Although cloud storage and especially P2P storage products are relatively new and have a limited track record in terms of reliability, security and scalability (and therefore must be considered moderately mature at best), their unique properties may address storage needs of SURFnet end users well.

We provide a short overview of basic principles underlying cloud- and P2P storage networks, but do not discuss technical aspects in great detail. For further reading, see surveys like [28], [18], and [4] which give a more elaborate discussion on specific topics. Especially [4] provides a thorough analysis of non-functional properties of P2P content distribution systems and their mapping on design features. Furthermore, it gives an overview of infrastructure technologies such as distributed hash tables and how they are used in a number of existing services.

In this document, we assume that users actively manipulate and manage files on their computers (e.g., stored on the computer's hard disk or a LAN network drive), and are familiar with the way files and directories (folders) are organized in tree-like structures. As such, they have a need to backup, share or synchronize their data. This differs from the model of operation where all data is exclusively stored in 'the cloud', as is the case with for instance Google Apps [15], and operated on through web applications such as on-line word processors. In this model (which we do not address here), data management takes on a different form, because the primary location of the data is not the local system but rather the web application environment in the cloud. Users do not directly control the data when stored exclusively in the cloud, and therefore must, for example, consider how to migrate the data in case of unsubscribing (i.e., must have an exit strategy).

Some online storage services exist that focus primarily on sending large files from one user to another (or to a group of other users). An example of such a service is FileSender [12] (to which SURFnet contributes). Typically, these services do not store data in a permanent fashion. For that reason, we do not consider these services here in more detail.

The remainder of this report is organized as follows. Section 2 introduces the most important principles that underlie the operation of a cloud storage service and a P2P storage network. Section 3 provides a classification of storage products, to be able to better indicate the differences between products. Section 4 compares cloud/P2P storage tools and products. Section 5 provides a scenario that highlights aspects of using synchronization and backup in a SURFnet context. Finally, Section 6 presents our conclusions.

2 Basic Principles

In this section, we discuss the most important principles used to create and maintain cloud storage services and P2P storage networks. Although the functionality required to *access* both types of storage from the user's computer(s) is quite different (i.e., is more elaborate for P2P storage access), many technologies play a role in cloud- as well as P2P storage, because the back-ends of cloud storage services are often organized similar to P2P networks. Therefore, in this section the term 'peer network' or 'peer system' refers to networks and systems in which the user's computer is taking on the role as one of the peer nodes in a larger storage network but also refers to the back-end systems and networks of cloud storage. The way the described principles are applied may differ per storage type, which is indicated in the text.

Most providers of cloud/P2P storage products do not disclose the exact architecture of their systems and software. Some have supplied general descriptions of applied technologies. Also, a substantial number of scientific papers have been published that focus on various aspects of cloud/P2P storage. This section discusses principles that are highly likely to be used in many of the currently available cloud/P2P storage products.

2.1 Redundancy

A peer network may have variable degrees of dynamics. In some cases, peers join and leave the peer network at a high rate, while in other cases this rate is much lower. Also, in some networks, peers are offline for a substantial fraction of the time, without leaving the network completely (i.e., they intermittently connect to the other peers). This is the case, for example, when peers are end user computers that are switched off when not in use. In contrast, peers (servers) in the back-end of a cloud storage provider are almost always online, being only unavailable in cases of maintenance or hardware failures. The rate at which nodes join and leave the peer network is called the *churn rate*. From the perspective of the needs of an individual node, peers are used to replicate files (or part of files) managed by this individual node. Because of churn and temporal unavailability, a node can under no circumstances rely on peers to be available at a time in the future, and therefore must store data at the peers in a *redundant* way, i.e., must ensure that the data placement at peers is organized such that in case of loss of a certain fraction of the peers, the data can still be recovered.

A common side effect of redundantly storing data at peers is the dispersion of data in a geographical manner (assuming that the peer network itself is geographically distributed). This effect reduces the risk of losing data in case of local physical destructions such as those caused by fire in a building or – at a larger scale – caused by natural disasters.

Storing a digital object redundantly at peers implies that the total amount of data used to store this object at the peers is larger than the original size of the object. The *level of redundancy* is measured by the redundancy factor, which is the size of the data at peers divided by the original size. A 5 MByte file and a redundancy factor of 4, for example, requires 20 MByte of storage capacity at peers. A high redundancy level has as a drawback the consumption of a high amount of network bandwidth and storage capacity, but has as a benefit that few peers need to be available to restore the original content.

The successful operation of a P2P network with autonomic individual nodes depends on sharing resources in a fair way. For P2P storage, this translates into allowing peers to store an amount of data on a node that is equal to the amount of data that this node stores with its peers. Consequently, a redundancy level that requires a lot of storage capacity at peers also requires a lot of disk space at a local node. Therefore, users of P2P storage facilities must always *balance* the level of redundancy with the amount of network traffic required to reach this level and the amount of storage capacity necessary to match its usage of storage at peers. For cloud storage back-ends, with nodes governed by a single administrator, load balancing replaces the issue of fairness, i.e., fairness is not an issue, but spreading the storage and access load over all servers is.

Once data is distributed over peers, it is necessary to perform *maintenance* by regularly checking how many peers are still available and how many of them hold on to the data they were requested to store. Depending on the churn rate, the level of redundancy decreases over time, which must be repaired by redistributing parts of the original data to new peers. As a consequence, networks with high dynamics require more network resources to maintain a particular redundancy level.

Redundancy of data is also at the heart of the most common P2P application, file sharing, but is used in that case for an entirely different purpose. File sharing is essentially a mass distribution channel for arbitrary files that replicates data to interested nodes and uses these nodes to further distribute data to other interested nodes. So, at any time, a file may be stored at many locations in a P2P file-sharing network.

While redundancy protects against loss of data due to (moderate) peer churn and temporal unavailability of a part of the peers, it does not compensate for other negative events and erroneous system behavior. For instance, bugs in the software – possibly introduced by a software update or triggered by malicious nodes – may make a large fraction of the peers unavailable or ill behaving. Also non-technical events, such as negative publicity, can lead to the withdrawal of a high number of users, which results in data loss and possibly an avalanche of other activities (peers massively trying to neutralize too low a redundancy factor, which overloads remaining peers, which makes even more peers unreachable). Peer churn and temporal unavailability is measurable, so it is possible to calculate the probability of unavailable data given a redundancy factor (see next subsection). Determination of the chances of data loss due to other types of events is likely only possible over time, when more experience with P2P storage solutions is obtained.

2.2 Erasure codes

Redundancy can be accomplished by simply replicating files at a number of peers. When the probability that a peer is online at a later stage is not very high, replication requires the data to be distributed to a substantial amount of peers to guarantee a high probability of finding the data online with at least one peer.

Another method for obtaining redundancy is to apply *erasure codes* to the file data. In information theory and telecommunication, erasure codes are used as error correction codes to compensate for errors occurring during the transmission of data. The original data is encoded into a longer message such that at only a fraction of the data suffices to reconstruct the original data the receiving end.

When applying erasure codes to storage, a file is first divided into *k* fragments and then encoded into *n* fragments, with $k < n$ and a constant fragment size. The ratio n/k indicates how much extra storage is used compared to the original file size. The main property of erasure codes is that *any k* fragments are needed to restore the original file. With similar storage requirements and peer availability, the usage of erasure codes delivers a much higher probability of successful data restore than straightforward replication [36]. This is the reason that erasure codes are used widely in P2P storage systems. The complete cycle of storage and retrieval of a file in a typical P2P system is depicted in Figure 3.

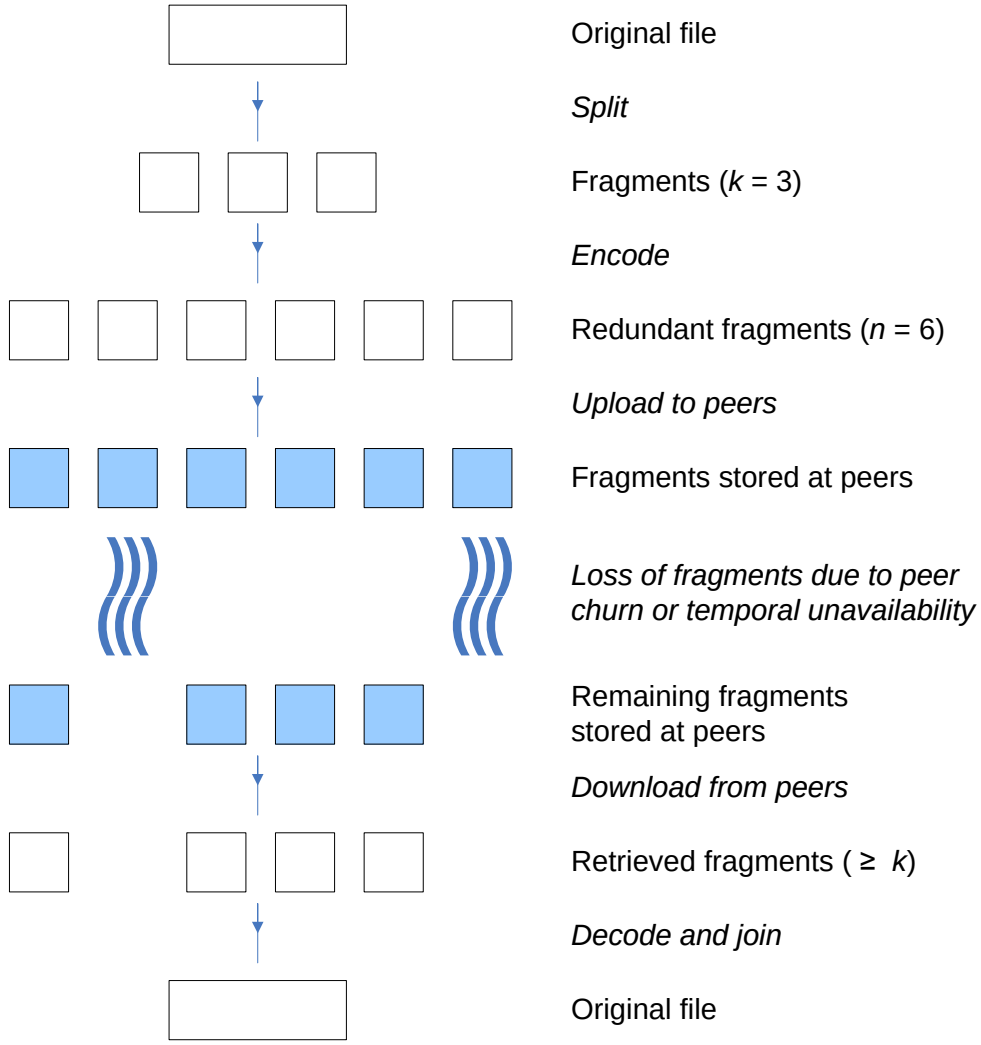

Figure 3: actions and data states during store and retrieval of a file in a P2P storage network, using erasure codes for the encoding and decoding of file fragments

To indicate the difference between the usage of replication and erasure coding, let us consider an example with a peer network consisting of 10000 peers of which 10% is offline, and assume a replication factor of 2 (a file consumes twice the original size in storage at peers). For this case, the probability that at least one of the two replicas is available is 0.99 (two nines). When fragmenting the original file in 8 pieces using erasure coding (storing 16 fragments at peers), the probability of finding at least 8 pieces online is 0.99994 (four nines).

Note that for cloud storage back-ends, the probability of peers being online is generally very high, which makes simple replication good enough for this static peer network setup.

2.3 Security

Several aspects are important when considering security in P2P storage networks. Clearly, when storing data over a data communication network at untrusted, anonymous peers, (fragments of) files must be protected to avoid peers or others reading their content (either when in transit or when stored on the hard disk of a peer). P2P storage products – like many cloud storage providers – encrypt data with a secret key before the data leaves the computer. In case of erasure coding, the data at multiple peers must be combined to reconstruct the original, so this mechanism increases the level of confidentiality. Likewise, facilities are in place to ensure data integrity (hashing). Although one of the first concerns for many users introduced to P2P storage products, data integrity is in principle guaranteed. Obviously, care must be taken to not compromise the secret key. For cloud storage services, data confidentiality is less important when the user trusts the service provider (one other, well-known party to trust); such trust is very hard to reach in P2P storage networks because peers me be anonymous and numerous.

Some products allow the encrypted data to be shared with others, which means that a mechanism must exist to selectively allow access to (parts of) the data. Traditionally, access control and identity management have hardly been addressed in P2P systems, basically because most of these systems had a content distribution objective in which all data is public data. For storage, products use a variety of approaches in this respect, so there is not a dominant model to describe here[1].

Other security aspects play an important role in the design of P2P systems (including storage networks). Individual nodes as well as groups of nodes acting together may display malicious behavior, for instance by discarding stored data, returning false data, or by intercepting communication between peers. Although this is not likely to compromise the confidentiality and integrity of the data, rogue nodes may, to some extent, disrupt the regular operation of a peer network or get an impression of how much data is stored in the network by an individual node. Detecting that all nodes behave correctly must therefore be part of the system's functionality (see also section 2.6 on fairness and quota).

[1] One example is 'Cryptree' [16], which is used in Wuala (see section 4 for the product description). This mechanism generates keys that can be used to access a part of the owner's file tree with specific rights and issues these keys to others. Providing read and write access to others with Cryptree is fairly straightforward; operations such as revoking read access, revoking write access and moving folders, however, are more complex.

2.4 Asymmetric read / write

New data generated by a user must be stored redundantly at peers, which requires fragments or replicas to be uploaded to multiple peers. When restoring data by reading from peers, only the minimum set of fragments (for erasure encoded data) or one replica (for replicated data) must be downloaded from online peers. This makes reading and writing actions asymmetric, because the amount of network traffic required to write a file is larger than the amount of traffic required to read a file. This effect is exacerbated for a P2P storage system in case a node attaches to the Internet through an asymmetric connection – with (much) more downstream than upstream capacity, which is typically the case for home users using ADSL or cable modems. The larger amount of data involved in writing a file must then be transferred in the direction with the least capacity. On the other hand, reading data is relatively fast, which is beneficial in certain cases, for example when using P2P storage for backup: when all data must be restored because of loss of the original data, the recovery is relatively fast.

Note that limited upstream capacity is a problem when a large amount of storage is brought online at once, for instance when connecting to a cloud storage service or P2P storage network for the first time. Assuming an upstream capacity of 1 Mbit/s (typical capacity for current high-end consumer subscriptions), a file collection of 500 GByte takes 45 days to upload completely to peers, without redundancy. This clearly shows that connection capacity imposes practical limits on the amount of data that a single node can store in a P2P network. Obviously, the same is true for the use of cloud storage services, although in that case no capacity is used for redundancy purposes (traffic for redundancy is generated at the back-end).

2.5 Peer selection

In a cloud storage service, a single authority manages all server nodes. The number of nodes is determined by the required capacity, and could grow to a large network of server nodes. Newly arriving data must be partitioned over the set of nodes, and when a data item is requested, the nodes responsible for storing that data must be easily found. The administrator of the back-end determines when to add new nodes to the set of servers.

For constructing a P2P storage network consisting of user computers there are two approaches. The first approach – in line with P2P networks known from file sharing – takes an open policy by admitting new peers freely. Such networks may grow to a considerable amount of peers. The second approach builds a peer network by deciding upfront who is part of the network, and selects peers using social links between people: a group of friends, family, or colleagues decide to setup a peer network and use each other's resources. The first approach requires an automated mechanism to add peers to the network and to discover new peers, and is capable of supporting very large networks. The second approach uses manual selection, and therefore cannot easily grow to a large size. Typically, in small, manually constructed peer networks, all peers are used to replicate local data, and there is social pressure to keep nodes up and running.

In large peer networks (comprised of many nodes), the fragments or replicas of a file are stored on a small fraction of the complete set of peers. In these networks, querying which peers store a particular fraction/replica by means of broadcasting would flood the network. Consequently, another mechanism is needed to lookup fragments or replicas. To retrieve data based on an identifier of that data, many P2P networks use a technique called a *distributed hash table (DHT)*. A DHT helps to locate the data in a peer network, without the need for a centralized index: it uses the structure of the peer network for lookup in an efficient way. Some cloud storage systems use DHTs or similar technologies for retrieving data from server nodes. The Amazon Dynamo system [10], not to be confused with Amazon S3, uses a hashing approach. Others look more like a distributed file system, where metadata including storage locations of parts of files are stored on dedicated server nodes. This is the case for the Google File System [13] and the Hadoop Distributed File System (HDFS) [17].

2.6 Fairness and quota

In a large P2P network, nodes contribute variable amounts of resources (disk space and bandwidth) and individual nodes expect to receive a comparable amount of resources from others in return. This means that mechanisms must be in place to ensure that the exchange of resources is fair. If a P2P storage network has weaknesses that allows for 'free riding' (i.e., users that take from others but do not give in return), the network is in danger of losing its value. For small, manually constructed networks, trading resources on a 'tit-for-tat' basis may be less of an issue. For cloud storage back-ends it is important to assign load to a node in proportion to the resources available on that node.

Large P2P systems use various mechanisms to enforce quotas for individual nodes. One solution is to let peers monitor each other, and punish overusing nodes by deleting their data. Apart from disk capacity and bandwidth, the fraction of the time a node is online is also an important parameter to consider: if a peer has a low uptime, its resources are virtually useless, because other peers have no opportunity to access them. It is not clear what will happen in case of sudden changes (especially shrinkage) of the peer network: a situation may occur, for instance caused by negative publicity, in which a lot of peers leave the network in a short timeframe. For an in-depth analysis of security issues, including a discussion on quota and monitoring, see [35].

2.7 Versioning

For most file systems on regular hard disks, whenever a user changes an existing file, the previous version of that file is lost. For some purposes, such as backup, keeping track of older versions of a file or directory is desired. This is the reason that some (but not all) of the cloud- and P2P storage products have versioning functionality, although this functionality may be implemented in various ways. In case of tight integration with the local file system, a storage process may monitor every change made to file and generate a new version in the online storage facility for that file. In cases where the storage process occasionally looks for changes, intermittent changes of a local file may be lost and fewer versions are stored in the peer network.

Note that versioning functionality increases the amount of storage capacity required. The cloud storage back-end simply uses more disk space to store all version of a file. The peers in a P2P storage network also must store more data and therefore increases the amount of local peer storage that must be provided in return. Especially when large files are changed often, such as could be the case for database repositories, versioning may result in excessive consumption of storage capacity[2]. Some storage products allow setting a limit on the number of versions stored for a single file or directory.

2.8 Efficient transmission

Online files need to be updated efficiently when they have changed locally (this also relates to verioning). Some files, such as log files or mail archives, change frequently while the changes are relatively small. A naive approach is to simply update the online version of this file as a whole. This works fine as long as the file is small, but requires a high data transmission capacity and long transfer times for large files.

A tool called 'rsync' [27] addresses this problem. It synchronizes two files that are stored at different locations, while generating a minimum amount of network traffic. Rsync's algorithm is based on calculating checksums over chunks of data of one copy, transfer these checksums to the other end, and then compare them with checksums calculated for the other copy. This detects which parts of the file differ, after which only the differences between the versions are transferred over the network.

The rsync algorithm or a similar algorithm may be used in cloud storage products, but requires that the file at each end is stored in the original format, i.e., is not stored in an encrypted form in the cloud (which would make comparing checksums useless). Comparing encrypted copies normally does not work, because a minor change in the original usually results in a completely different encrypted copy (compared to the previous encrypted copy). For P2P storage, encryption of the data is necessary, which makes this kind of efficient file updates impossible. For cloud storage, the property of data security must be balanced against the property of efficient updates.

An alternative approach which combines transmission efficiency with data security is to determine differences locally (for instance by keeping track of checksums of chunks), encrypt these differences and send this to the online storage. The main disadvantage of this approach is that for a file that changes often, many file 'diffs' may accumulate at the online storage. This may result in a longer download time. Note that the way in which a file is stored may influence how efficient the diff data is handled: Amazon Dynamo (which is used by Amazon internally as a data storage facility) stores data items in one piece, while the Hadoop HDFS splits large items up in chunks. Chunked storage is more likely to work well with mechanisms such as those implemented by rsync.

[2] Some mechanisms may be in place which store new versions in an efficient way, for instance by only storing the *difference* between versions, at the cost of introducing extra complexity.

3 Classification of Storage Products

In this section, we provide a number of criteria that are suitable for classifying storage products. These criteria are: 1) the type of storage, 2) the overlay network centralization, and 3) the network structure used in the peer network. All of these criteria are used to compare P2P storage products in the next section. We use only the type of storage criterion to compare cloud storage services (in the next section), because the back-end organization details of these services is mostly unknown.

Cloud- and P2P storage technologies are used for various *types of storage* needs. Products can be classified based on their primary use. We discern the following types of storage usage:

- *Backup*. Using the service as a backup facility for files stored locally on a computer (which is part of the peer network). This may involve keeping track of versions of files, as they change over time.
- *File synchronization*. Keeping the same file tree that exists on a number of different computers in sync. When one file is changed on one computer, the copy of that file on the other computers is automatically updated. This type of functionality must deal with conflicts, e.g., in case the same file is changed on multiple computers at the same time.
- *Distributed file system*. The online storage capacity is used to implement a distributed file system. One or more computers access the storage in a manner that is very similar to local file systems (i.e., applications on the computer may access remote files as if they were available through a local file system).
- *Content sharing*. Parts of the file tree stored online are used to share data with other people. By providing credentials to others, they can use the storage facility to read the part of the tree they were granted access to. The level of sophistication in terms of granularity of access rights may vary per product.

In many cases, storage products combine multiple storage types. Also, some overlap between the different types of storage exists.

P2P systems can also be classified based on the *overlay network centralization* they apply (i.e., the degree in which the organization of the peer network is centralized; see [4] for an in-depth discussion). Many aspects and characteristics of a P2P system are profoundly influenced by the chosen level of centralization. The scalability of a system is high when all nodes participate in all tasks, but some tasks may be hard to execute efficiently in a fully distributed manner. We identify the following types of centralization:

- *Fully decentralized*. All nodes in the peer network have exactly the same tasks. As a consequence, this organization does not define centralized functionality, making it scalable and resilient to failure. In a fully decentralized system, however, tasks such as the lookup of data (based on an identifier) and the tracking of the contributions and behavior of individual peers (to prevent free riding and to purge malicious nodes) is not trivial. Note that even for fully decentralized networks, a network 'entry point' is still necessary, for those nodes that wish to connect to the network for the first time.
- *Semi decentralized*. Some of the nodes take on extra responsibilities (sometimes they are called 'supernodes') but the loss of these special nodes is gracefully handled by the system. Supernodes can be useful when a task of the system cannot be efficiently implemented in a fully distributed manner: some data lookup mechanisms rely on supernodes to improve performance. They can also help to circumvent limitations of individual nodes, which, for instance, may be caused by firewall or network address translation (NAT) configurations.

- *Hybrid centralized/decentralized.* P2P communication is used to transfer data, but a centralized entity is responsible for such tasks as indexing (lookup of data), which makes this entity a single point of failure. For tasks that require a lot of CPU resources or communication resources, a centralized node quickly becomes a bottleneck, which was, for example, the case for early P2P file sharing systems such as Napster. Some tasks that do not consume many resources may be handled by a central entity, even when the number of peers in the network is large.

A third way to classify P2P systems is by *network structure* [4]. This determines how content is inserted into the peer network. The following types of network structure are discerned:

- *Unstructured*. The placement of content is not related to the overlay topology (i.e., the way peers are linked within the network). Lookup of data is best effort and usually floods a region of the network. Peer churn is handled well and it is easy to search for content with partial keys. Unstructured networks are generally unsuitable for large storage applications, because of the best effort lookup, i.e., there is no guarantee that existing data is found. For small networks, however, broadcast lookup may not be a problem (i.e., just asking all peers for a replica may work well).
- *Structured*. The topology of the network determines the placement of the content. An example of such an organization is a distributed hash table. The data can always be found in structured networks (if the peer is online) but peer churn is not handled well. Furthermore, compared to unstructured networks, it is harder to do wildcard lookups, i.e., to lookup data, the full key (not a part of the key) must be provided.

4 Products

This section provides an overview of cloud- and P2P storage products and services. Our main focus is on those products that are suitable for end users, although we also provide a short description of research projects and tools. The products are therefore discussed according to the following subdivision: 1) *research projects*, 2) *simple tools*, 3) *cloud storage products*, and 4) *P2P storage products*. The cloud storage services can be divided into those that offer a broad platform (which are suitable to build a broad range of web applications) and those that are strictly storage products. The P2P cloud storage products fall in two different categories: those that incorporate functionality for automatic network management (potentially scaling to large networks) and those that manually select peers to form a (small) network. The research projects deal with P2P storage from the first category.

The usage of online storage services is currently growing quickly, and the number of products has risen sharply over the last few years. Also, products are still evolving and becoming more mature. This makes the cloud storage market highly dynamic with a large number of suppliers and products that change often. The overview of products in this section therefore must be considered an incomplete snapshot of current (Feb. 2010) offerings; a comprehensive listing of products would be very long and goes beyond the purpose of this document. Also, new suppliers are likely to change the list of prominent products in the future (which would be the case, for instance, when Google were to extend its storage offering).

4.1 Research projects

Over the last decade, various research projects have investigated mechanisms for P2P storage. Most of them use one of the existing distributed hash tables, such as Tapestry [38], Chord [29] and Pastry [30], to lookup data in the peer network. A non-exhaustive list of projects, including a short description is given below:

- *OceanStore* [26]. Is a distributed, persistent storage system using erasure codes for redundancy. Data objects consist of blocks and are versioned; data is located using the Tapestry distributed hash table. A transaction mechanism is part of OceanStore.
- *CFS* [9]. The Cooperative File System (CFS) is a system that stores (replicas of) file system blocks at peers, using a distributed hash table (based on Chord).
- *PAST* [31]. PAST is a large-scale distribute system for the storage of immutable files that are widely replicated. The data granularity is at file level (not at file block level such with OceanStore and CFS). PAST uses the Pastry distributed hash table.
- *Ivy* [24]. Compared to the above systems, Ivy is a P2P file storage system that allows both reading and writing, as well as sharing data between multiple users.
- *A Cooperative Internet Backup Scheme* [20]. This system is focusing explicitly on P2P backup, and includes features to guard against free riders and misbehaving nodes. It uses erasure coding for redundancy.
- *PeerStore* [19]. Focuses on backup, and distinguishes itself from the above systems by decoupling the storage of metadata (such as the location of replicas) from the storage of the actual data.
- *LoDN* [5]. The Logical Distribution Network (LoDN) service is a P2P content distribution network for sharing large files in a community. Contrary to the projects above, LoDN is a live system with currently 42 TByte of online storage. LoDN is slightly out of scope here, because users do not participate as a peer but simply store data in the network.

4.2 Simple Tools

A number of tools are available that support rudimentary functionality to store, synchronize, and backup files between computers. Although missing many features that can be expected from full-fledged storage systems, these tools may work well in circumstances that require little advanced functionality. An advantage is that they have been available for a long time and therefore are well understood and contain very few (software) errors. Furthermore, they are available on a wide range of platforms.

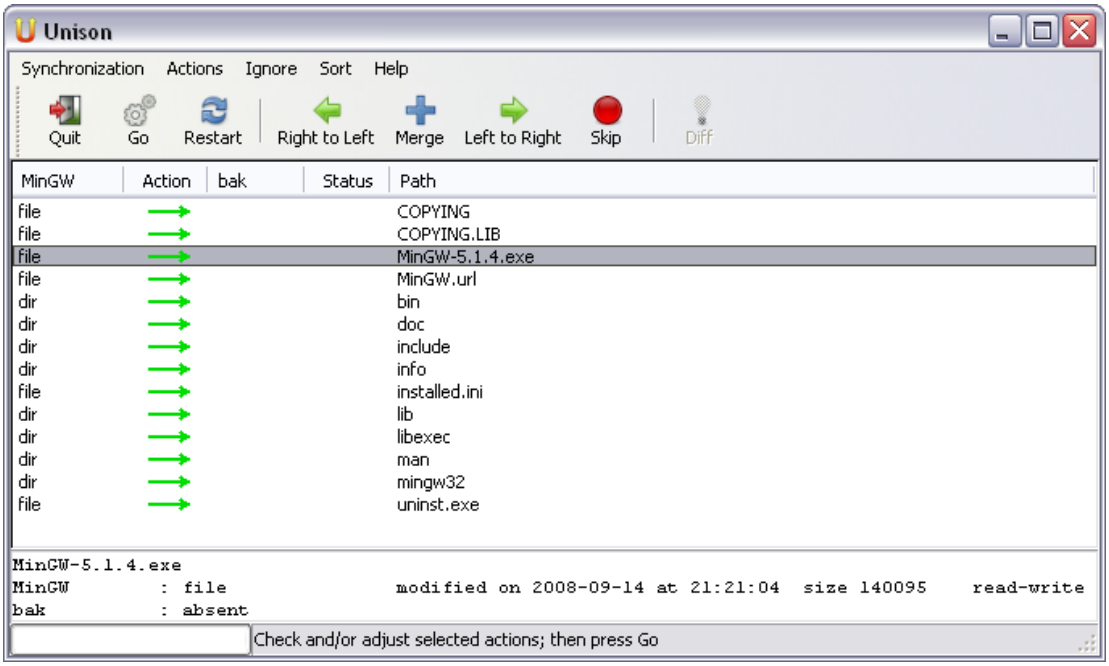

Figure 4: Unison file synchronization tool

The *rdist* and *rsync* command line tools synchronize files and directories between two locations. Compared to rdist, rsync determines the differences between two copies of the same file in order to only send the difference between them. When synchronizing over a wide area network, this is advantageous in terms of bandwidth consumption and time. To synchronize a part of a file tree with a remote copy, the user must run the rdist or rsync tool manually (i.e., synchronization is not an automatic process).

A more advanced tool is *unison*, which is available in the form of a command line tool and as a tool with a graphical user interface (see Figure 4). Unison synchronizes data between two directories and is based on rsync. Although mainly focusing at keeping file trees on two machines in sync, it is also capable of keeping file trees on more than two computers in sync. When the same file is changed on two machines, unison indicates that a conflict has occurred. Like rdist and rsync, unison does not keep older version of files and directories.

4.3 Cloud Storage Products

4.3.1 Amazon S3

Amazon S3 (Simple Storage Service) [3] is a web service that offers cloud storage through a simple HTTP-based interface. The Amazon web services platform is one of the most prominent and widely used cloud computing environments with data centers spread out accress the world, incorporating (virtual) server, database, payment, and other facilities: its S3 storage service stores many billions of objects. The S3 service stores objects in a reliable, distributed manner in the back-end. It charges for the storage used, the amount of data transferred in- and out of the Amazon environment, and per request issued for an object. Prices for storage are around $0.15 per GByte per month.
The storage inside S3 is organized in the form of *buckets* in which data objects can be placed. S3 does not support a hierarchical object structure such as common in file systems. The web interface defines simple actions, through HTTP PUT, GET, POST, and DELETE methods, on buckets themselves and on objects inside buckets. It is possible to share buckets and objects with others by setting the access control policy accordingly. Data is stored as supplied, i.e., is not encrypted before storing

Amazon S3 is not an end-user cloud storage product in the sense that it provides client software that takes care of backup, synchronization and sharing of files on end-user computers. We incorporate its description here, because it is used by other products that do deliver this functionality, using S3 as the basic storage provider.

The design of the S3 service is not made public by Amazon. It is likely, however, that a part of its functionality is based on (or at least inspired by) the Amazon Dynamo storage system described in some detail in [10]. Dynamo uses various P2P mechanisms on the back-end to ensure scalability and fault tolerance. Also, the design principles as described on the S3 web site (‘decentralization’, ‘autonomy’, ‘symmetry’, …) hint towards the usage of basic P2P technologies.

4.3.2 Dropbox

Dropbox [11] is a cloud storage provider and file synchronization tool using the Amazon S3 storage facility as a back-end. The Dropbox client runs as a background process on a local computer and watches file tree changes under a designated directory to synchronize with the data in the cloud (and, indirectly, with the data on other machines). It is available for a number of different platforms, including Windows, OSX, and Linux. Dropbox offers a free account for storing a maximum of 2 GByte of data. Paid subscriptions are available for 50 GByte and 100 GByte of storage.

The client may be installed on a number of different computers and then linked to the same account to keep the file tree in the Dropbox folder synchronized between all machines. Computers do not need to be online all the time, and a file may be altered on multiple machines before they synchronize again, which may result in update conflicts. Dropbox reports conflicts by storing conflicting version in the same location under another name (extending the original name). Furthermore, it stores older versions of an item, which enables undoing changes that were made by mistake. By default, Dropbox synchronizes machines moving data back and forth between individual computers and the back-end. It also has, however, functionality to keep computers in sync using a direct connection between these machines (which is called ‘LAN sync’).

The client software installs in a very straightforward way and has a minimal interface. On Windows, it displays an icon in the taskbar, which can be used to change preferences and to get a list of recently changed items. The Dropbox folder is a regular directory on the file system (see Figure 5); to place items in the Dropbox environment, they must be placed in this folder. The client does not support selecting a set of existing directories for inclusion, although it is possible to put them into the Dropbox folder using file system links. The files stored under a Dropbox account may be accessed using the Dropbox web interface. This interface also supports tracking changes and events, and enables the sharing of files and directories with others (also non-Dropbox users). See Figure 6 for a view on the same files as depicted in Figure 5.

Dropbox files are stored in a secure manner at the back-end, but they can be decrypted by Dropbox. This means that users must trust Dropbox and rely on Dropbox to handle their data securely, which may not be appropriate for some types of usage and data. If security is a high priority, other tools or service may be better suited than Dropbox. It is likely that having access to unencrypted data allows Dropbox to more efficiently transport data between participating computers and the back-end.

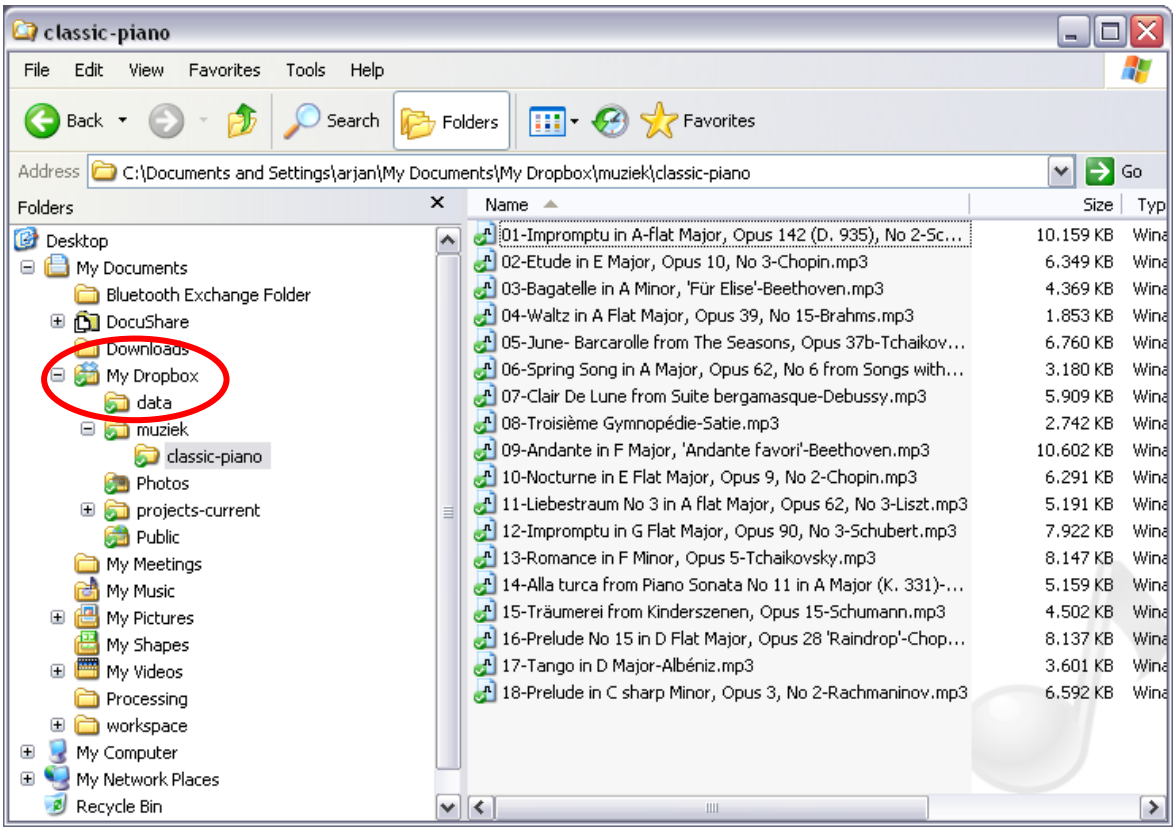

Figure 5: the Dropbox folder is a regular directory on the local file system (here positioned under the My Documents directory on a Windows XP system)

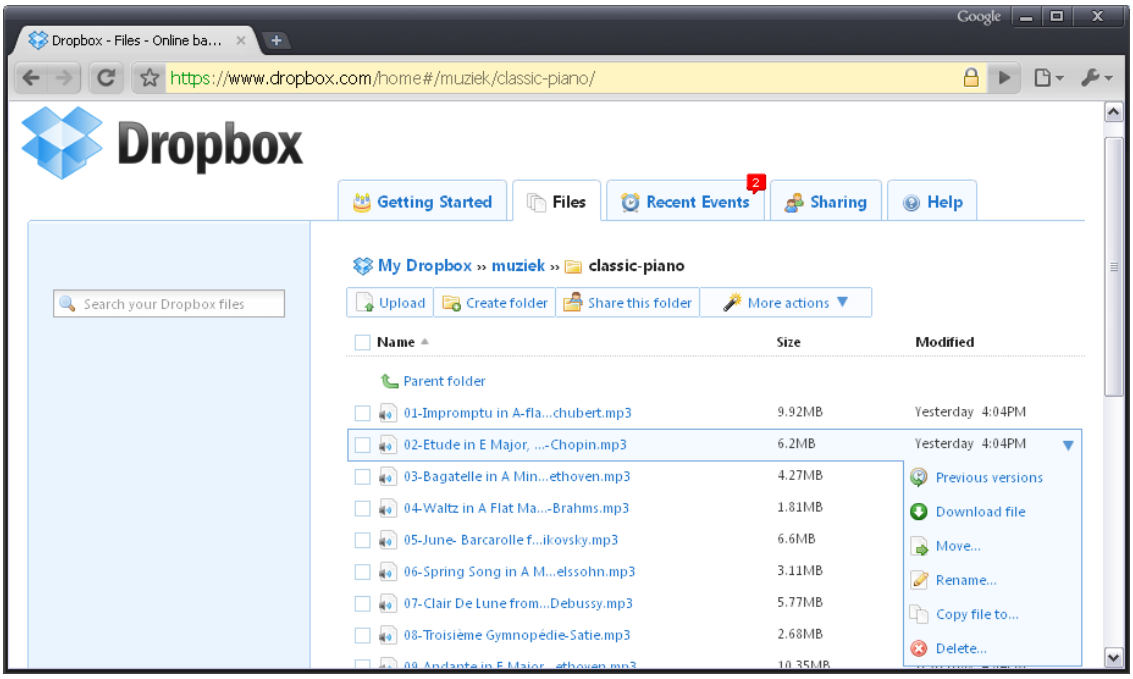

Figure 6: view on Dropbox files inside a browser

4.3.3 Microsoft Live Mesh / Live SkyDrive

Microsoft offers two cloud storage services with overlapping functionality, but with separated data stores at the back-end: Live Mesh [22] and Live SkyDrive [21]. Both services are free and have limits on the amount of storage, which cannot be increased by taking a paid subscription. The Microsoft Live platform is a web platform that hosts various web applications such as mail (Hotmail), calendar, and document editing tools; both Mesh and SkyDrive are part of this larger environment. Microsoft Live has evolved over the last number of years, and it not always clear to which extent its components are integrated. Additionally, it is not clear whether Live Mesh and Live SkyDrive will continue to exist in parallel or if they will be merged into one product. Compared to SkyDrive, Live Mesh currently is the service with the most advanced storage features; Skydrive, however, offers a larger storage capacity. No details about the back-end designs of both services are made public by Microsoft.

The Live Mesh service supports the synchronization of files and directories between multiple computers and the sharing of files with others. It uses a cloud storage facility with a maximum of 5 GBytes to share and synchronize. When a file on one of the connected computers is changed, the new version is synchronized in the background with the cloud storage as well as the other attached computers. Directories on the local file system can be added (on Windows) using the Windows Explorer; clients for other platforms are also available (OSX, Windows Mobile). Files and directories stored online can be viewed using a web browser, which also offers management functionality. Items can be shared with other users within the Live context. Live Mesh does not support versioning.

SkyDrive is a cloud storage service that allows users to store up to 25 GByte of data, with an individual file size limit of 50 MByte. It does not provide synchronization functionality between devices, and the primary interface is web browser based. A number of tools exist, however, that make uploading and syncing of local data more convenient; an example is SkyDrive Explorer [32], which is a Windows Explorer extension. Like Mesh, SkyDrive does not support versioning of objects. Individual items can be shared with others through the web.

Some confusion exists concerning the relation between Live Mesh, Live SkyDrive and another tool called Live Sync. Live Sync synchronizes files between individual computers, and, contrary to Mesh/SkyDrive, does not involve cloud storage. This means that for Live Sync to work, the machines must be online at the same time; something which is not necessary for Live Mesh (SkyDrive does not support multi-computer synchronization).

4.3.4 ADrive

ADrive [1] is a cloud storage service that supports backup and sharing of data. Its basic service offers a large amount (50 GByte) of free storage. ADrive provides access to the online data through a web browser, a WebDAV interface, FTP, and a desktop application that runs on a range of operating systems, although only the first can be used with the basic (free) service. Paid subscriptions offer more features than the basic service and provide variable amounts of storage capacity. The additional features are, apart from the extra access functionality: file versioning, encrypted file transmission, concurrent access, and support. ADrive does not provide functionality for keeping multiple computers synchronized and for automatic syncing of local data with cloud data.

Few details about the design of the ADrive service are provided, which makes it hard to judge important aspects such as reliability and scalability. Data is not encrypted before leaving the local machine.

4.3.5 Mozy

Mozy [23] is a cloud storage facility focusing on backup of files on individual computers. The Mozy client keeps track of changes on (parts of) the local file system, and submits these changes to online storage. All data stored in the cloud is encrypted at the local computer, which makes Mozy more secure than Dropbox and ADrive. Furthermore, Mozy uses incremental backups to only store differences between files rather than completely new versions of files for every small change. It supports restoring versions of files up to 30 days old.

Like Dropbox, it offers 2 GByte for free. A paid subscription for personal use offers unlimited storage capacity for $5 per month. Business users pay per amount of data used. Mozy does not have functionality for synchronizing multiple computers or sharing data with other users. Like many other products, Mozy supplies few details about the design of their storage facility.

4.3.6 Cloud storage product feature comparison

The table below summarizes the products described above, showing their key features for easy comparison.

	Amazon S3	Dropbox	Live Mesh	Adrive	Mozy
Product type	commercial, developer focus	commercial, end-users	commercial, end-users, developers	commercial, end-user	commercial, end-user, business user
Storage type(s)	backup, sharing	backup, sync, distr. file system, sharing	backup, sync, distr. file system, sharing	backup, sharing	backup
User interface(s)	web	client, web, file system	client, web, file system	web, client, FTP	client
Pricing	used storage, data transfer and operations	first 2 GByte free, subscription	5 GByte free	50 GByte free (limited funct.), subscription	2 GByte free, subscription
Quota Limits	none (paid)	100 GByte max. (paid)	5 GByte max. (free)	1 TByte max. (paid)	none (paid)
Back-end structure	distributed	distributed (Amazon S3)	unknown	unknown	unknown
Versioning	yes	yes	no	no	yes
File encryption	no	yes, at back-end only	no	no	yes
Platforms	web	Windows, OS X, Linux, Web, iPhone	Windows, OS X, Web	Windows, OS X, Various Unixes, Web	Windows, OS X

4.4 P2P Storage Products

4.4.1 Wuala

Wuala [37] is a commercial, distributed storage service that allows users to trade storage capacity in a P2P way and buy additional storage capacity managed by Wuala. Access to this service is supported with a Java client application that installs on a wide range of common operating systems. This product is proprietary, and the underlying architecture and protocols are not fully disclosed, which makes it only partly possible to identify all of its features in detail. Wuala's company (Caleido) was acquired by the external hard drive manufacturer LaCie in March 2009. Currently, Wuala reports to manage more than 100 million files in its network.

A participant in the Wuala network can choose to share disk resources with others. Wuala provides a limited amount of free storage capacity (currently 1GByte) and it is possible to purchase additional storage capacity. In that case, the Wuala software acts as a client only, not providing resources to peers in the network. A main feature of Wuala, however, is the possibility to exchange disk and bandwidth resources with peers without additional monetary costs. In this mode, a node acts both as a server and a client. All data stored in the network is first locally encrypted and then stored in pieces – after erasure coding – at peers in the network. Other important features are the sharing of (a part of) the file tree with friends or groups, and to provide public, unrestricted read access to (a part of) the tree.

Every participant may install the Wuala software on multiple computers. When sharing resources, one or multiple of these computers needs to provide peer access to local storage; all computers, however, have full access to the files stored in the peer network. This means that Wuala can be used to keep files synchronized between machines. It is not clear how Wuala deals with synchronization conflicts that arise when multiple nodes change the same file at the same moment.

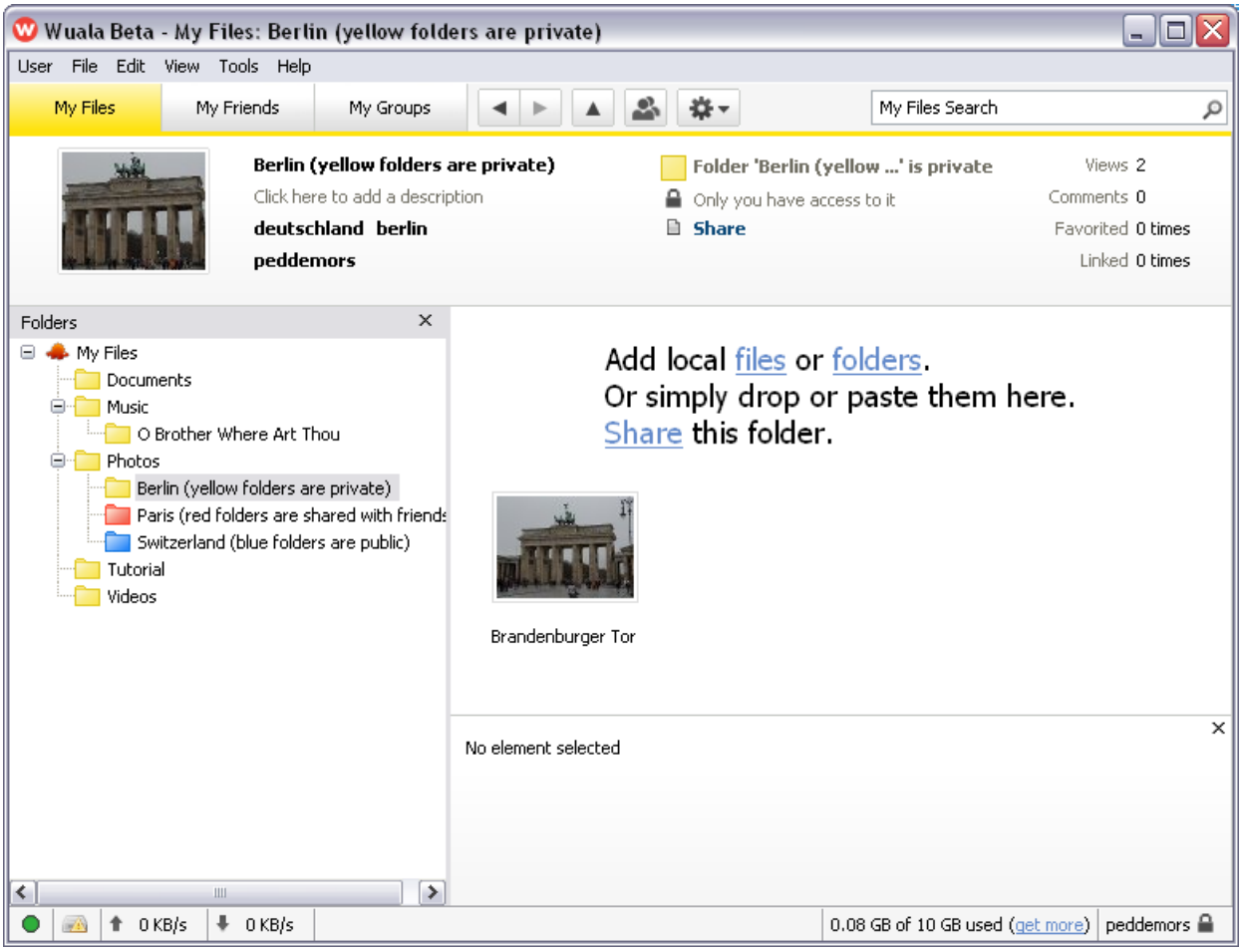

Figure 7: Wualu client GUI showing files available in distributed storage

Installing the Wuala software is very straightforward and smooth. Access to the Wuala network is granted after registering as a user. The main user interface consists of a directory view on the files stored in the network. On MS Windows, drag and drop interaction with the regular file explorer allows for placing files in the network, while online files may be opened with local applications. Additionally, the application supports managing connections with friends and memberships of groups, and changing permissions on files and directories. A snapshot of the user interface is shown in Figure 7.

When a participant's computer is online for a high enough fraction of the time, it is allowed to trade storage with peers in the network. This threshold exists to avoid storage of file fragments at nodes that later have a high probability of being offline. The amount of online storage is proportional to the amount of local storage offered for trade, and is calculated with a simple formula: online storage = online fraction * local storage. So, if a node offers 10 GByte local storage and is online for 80% of the time, the online storage available is 0.8 * 10 GByte = 8 GByte. It is not clear how this formula accounts for the necessary redundancy when storing data at peers, i.e., when storing 10 GByte of data in the network, in reality more data is stored to accomplish redundancy. This would result in using more storage capacity at peers than providing local capacity to peers. The level of redundancy cannot be configured with the Wuala software, and it is not clear what the default settings for redundancy and erasure coding are. Once a certain amount of data is stored in the network and the online storage capacity drops below that amount due to a lower average online time of the participant's node(s), no additional data can be written to the network and the participant is notified of imminent loss of data. As a consequence, when a participants stores close to the maximum amount of data into the network, care must be taken to not reduce the online time (e.g., in case of long periods of absence). Note that, when starting the use of the service, it takes some time to build up a high online fraction.

The access to files and directories stored in the network is based on the 'cryptree' method [16]. This method allows for sharing of parts of the file tree with others with a fine grained level of control over the permissions, without revealing the identity of other participants with access. Furthermore, it deals with revoking the rights to access a part of the tree. All files are encrypted with a different key before storage into the network, and the list of keys for all files of a participant is encrypted by the participant password and stored on Wuala servers. Therefore, when the username and password is lost, all data is lost. Wuala does not provide insight into impact on the scalability of their service by the storage of individual file keys.

Compared to other products, Wuala currently offers the easiest way to store files in a P2P manner. The installation of the application is straightforward, and there is no need to manually select peers, which gets a participant up and running within a couple of minutes. Also, Wuala seems designed for a large peer network where peer churn may be high and peer availability relatively low. Unfortunately, little data exists on the operational aspects of the Wuala network: no information is provided about the risk of data loss, the number of users, the ratio between Wuala supplied storage and peer supplied storage, etc.

4.4.2 Tahoe

Tahoe [34] is an open source distributed filesystem that supports storing files into a network of peers. It is the enabling technology behind the Allmydata cloud storage company [2], where it is used to organize the back-end servers in a P2P manner. Contrary to Tahoe, however, the Allmydata cloud storage service does not expose P2P features to clients. Tahoe is written in Python, and uses various native libraries for specific functionality such as encryption and erasure coding. It runs on various popular platforms such as Windows, OS X, Linux, etc. and is now part of the Ubuntu Linux distribution.

Tahoe defines a number of different node types:

- Storage nodes (or server nodes)
- Client nodes (reading/writing files)
- Introducer nodes (connecting other nodes)

A node may combine multiple types of functionality to, for instance, upload own data (client) and to provide storage for peers (server). The Introducer is a centralized role that has as task to inform clients about available servers.

Installing and configuring the software is mostly manual and requires the manipulation of configuration files and execution of shell commands. Additionally, Tahoe is a software product only and does not provide access to a common existing peer network (except for a test network). Therefore, the usage of this product is most suited for those who intend to build their own peer network, for instance amongst a group of colleagues or friends. This requires the existence of an Introducer node.

A number of methods exist to access files in the peer network. Local client functionality may be accessed through a web interface, through the command line, or through a plugin on the file system. The web interface shows the directory tree in HTML pages, while the file system plugin allows local applications to seamlessly interact with files in the Tahoe system.

Access to the data stored in a Tahoe network is based on a short byte string, called 'capability', that uniquely identifies a file or directory and determines the location of fragments and the access rights for that object. Directory objects are simply a list of capabilities of the files in that directory. By sharing the capability, users can give others access to a file or a file tree. There does not seem to be a simple way to revoke access once the capability is shared with somebody else. Furthermore, Tahoe does not keep track of versions of files or directories. The erasure coded fragments of a file are distributed over the peers using a distributed hash table.

The origin as a cloud technology shows in the (lack of) features of Tahoe. There is a bias towards a configuration with a few peers that are more or less equally dimensioned in terms of storage capacity, and that are online for a large fraction of the time. Client are connected to all servers, so in case of nodes with a dual role, a fully connected mesh is established which does not scale well. Also, no mechanism exists to enforce fairness in the sharing of resources.

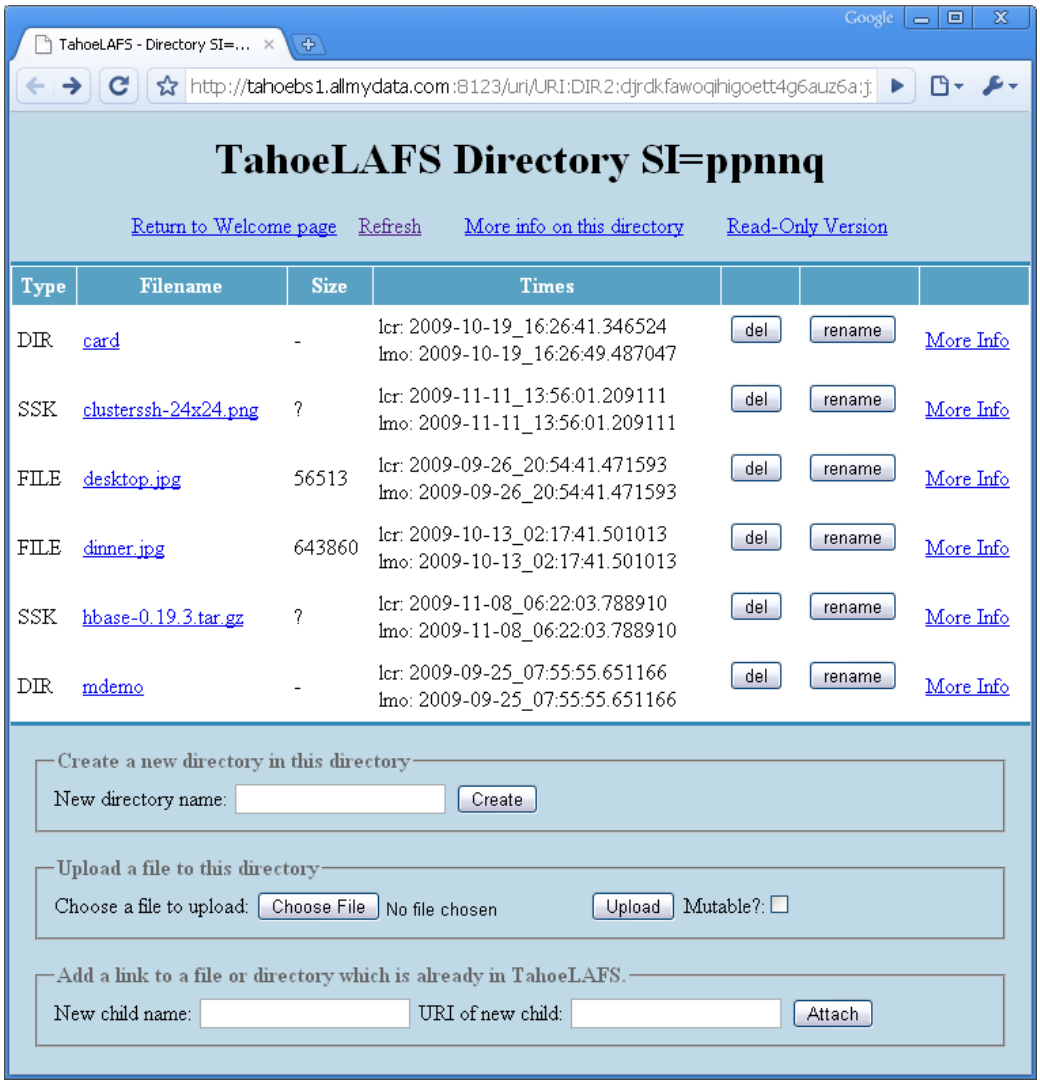

Figure 8: Tahoe web interface showing the content of a directory

For a more detailed description of Tahoe, see the architecture document [33].

4.4.3 CrashPlan

CrashPlan is a backup application that supports P2P backup between a set of self-managed computers and a backup service that stores computer backup data online. CrashPlan is not a full P2P solution, but rather relies on a user-selected set of peers. A common scenario for CrashPlan is to agree amongst friends to exchange storage capacity on each other's computers in order to safely backup files. In another setup, a user with multiple computers (for instance, at work and at home) uses storage capacity on one to backup the other and vice versa. Clearly, this product supports a different kind of P2P storage than Wuala and Tahoe.

Personal use of CrashPlan is free (but advertisement supported) and software is available for a number of different operating systems. The purchased version of CrashPlan does not show advertisements. The online service is based on a paid subscription, much like cloud storage subscription although CrashPlan charges per computer and not per amount of data stored and transferred.

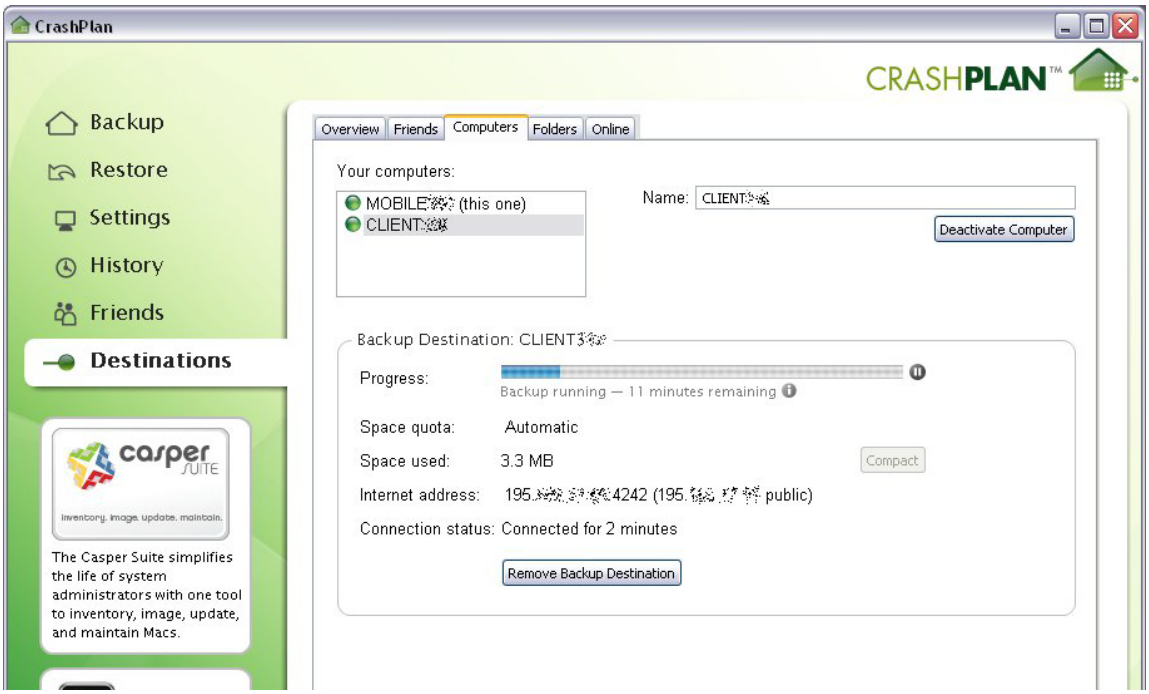

Figure 9: Backup in progress (from one machine to another), using the advertisement-supported version of the CrashPlan software

In a typical configuration, CrashPlan stores files available on an individual computer on multiple destination computers, preferably being a mix of local, offsite (on the Internet), and cloud machines (see
Figure 9). It encrypts all data. More advanced features are the possibility to seed the initial backup to the CrashPlan cloud with physical storage, in order to reduce the upload time when starting with a large amount of data.

4.4.4 Cucku

Cucku is a P2P backup application [8] of the same type as CrashPlan: it relies on the manual configuration of peers, typically being computers of friend or family (called ‘social backup’ by Cucku). The version of the application with basic features is available for free, and allows backup to a single peer only. A version with extra features, including the possibility to use multiple peers, is available for purchase. Cucku runs on MS Windows machines.

Apart from an account with Cucku, users must also have a Skype account. Skype is used for transfer of data between peers. Like CrashPlan, Cucku supports versioning of files, which means that if files are changed or deleted, old versions will remain available.

4.4.5 PowerFolder

PowerFolder is a P2P storage product that supports different kinds of storage uses. It has synchronization functionality to keep file trees in sync between computers. Furthermore, it acts as a backup facility and share and collaboration tool. The PowerFolder software is written in Java and runs on various operating systems. It is available as a free version (with limitations), and a purchased version (without limitations) that includes online storage and must be renewed every year.

Although focusing on a wider range of applications, PowerFolder is comparable to CrashPlan and Cucku. Like these product, PowerFolder is based on manual peer selection. The usage of online storage is comparable to CrashPlan.

4.4.6 P2P storage product feature comparison

The table below summarizes the products described above, showing their key features for easy comparison.

	Wuala	**Tahoe**	**CrashPlan**	**Cucku**	**PowerFolder**
Product type	commercial, basic product free	open source	commercial, basic product free	commercial, basic product free	commercial, basic product free
Storage type(s)	distr. file system, sharing	backup	backup	backup	sync, backup, distr. file system
User interface(s)	client, virtual disk	virtual disk, web, command line	client	client	client, web
Decentra-lization	hybrid	full	full	hybrid	full
Structured network	type unknown	distr. hash table	fixed topology	fixed topology	fixed topology
Peer selection	automatic through Wuala network	automatic through own network	manual	manual	manual
Versioning	yes, for pro users	no	yes	yes	yes
File encryption	AES	AES	blowfish	AES	AES
Redundancy	erasure codes	erasure codes	replication	replication	replication
Platforms	Windows, OS X, Linux	Windows, OS X, Various Unixes	Windows, OS X, Linux, Solaris	Windows	Windows,OS X, Linux
Remarks		used and supported by Allmydata	integrates with online storage		integrates with online storage

5 Scenario

Now, let us consider a scenario that illustrates how cloud- and P2P storage products may be used in a real situation. It concerns a student who is working on her thesis from various locations, using multiple computers. By picking this specific scenario, we cover various aspects such as backup and synchronization form different machines and locations, although we realize that not all types of usage of storage services by SURFnet end-users are covered. Indeed, for a staff member who is exclusively working from a fixed PC at the office using files on a network drive that gets regular (traditional) backups, this scenario may not apply. Many users, however, use multiple machines for their daily activities, and for them managing files efficiently and safely is important.

The scenario description is as follows. As input for the thesis, the student is analyzing and processing in the order of 20 GByte of data from a number of experiments she has previously conducted. Obviously, the student does not want to lose thesis files and experiment data files, and would like to keep track of older version of the thesis files (the files with raw data from the experiments do not change; additional large files, however, may be generated during the data analysis activities of the user). She has a regular PC at the university which she uses to process the data. She writes most of the thesis on her laptop, although parts of it are written on a university PC and another regular PC at home. The PC at home has a fixed ADSL Internet connection, while the PC at the university has a fast Internet connection through the SURFnet network. When available (i.e., at the university, at home, and occasionally at other locations), her laptop connects to the Internet through a WiFi wireless network. The scenario setup is depicted in Figure 10.

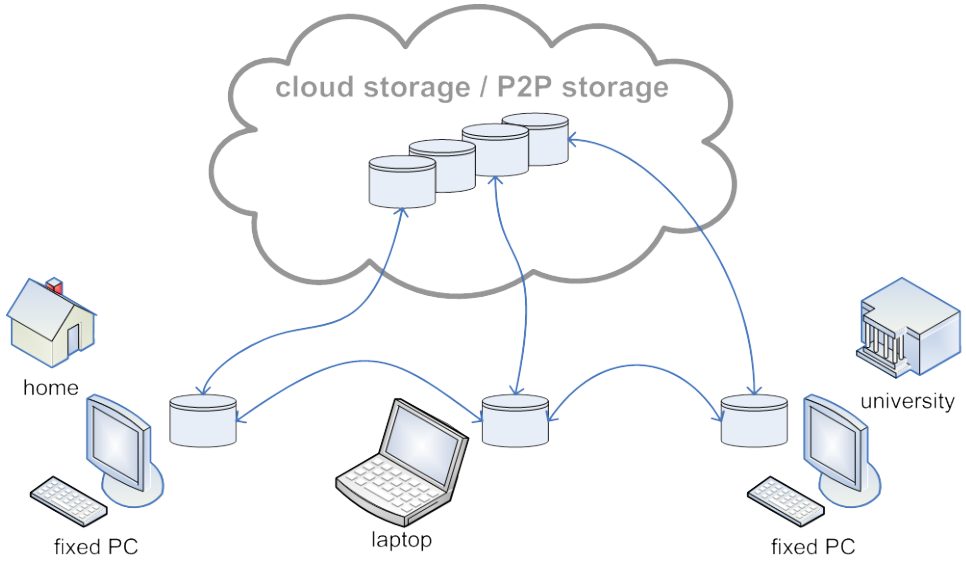

Figure 10: scenario configuration in which a student uses three different computers: a fixed PC at home, a fixed PC at the university and a laptop at various locations. Data on these machines is synchronized and stored at an online storage facility.

This student has a number of requirements concerning her data. She wants to keep files synchronized between the computers she is using to process and analyze the data from the experiment, and to write on the thesis. Furthermore, she needs to keep a backup of the files, so that, in case of loss of one of her computers, no data is lost. Also, older versions of a part of her files must be available, for instance to undo changes she made by mistake. There is no need to share the data with other persons.

The products and tools described in the previous section may meet these demands at different levels. We now discuss a number of aspects of the data handling, using three different products: Unison, Dropbox and Wuala. Unison is a basic tool that has most required functionality, works locally between the set of used computers, but needs considerable manual configuration, while Dropbox and Wuala have a surplus of features (for this scenario), are easy to install and configure, but generate Internet traffic. Dropbox stores data at a server back-end and Wuala stores data at peers (requiring local storage in return). As such, these three products have overlapping functionality but are based on different principles and therefore are good candidates for comparison.

5.1 Preparation

For first usage, the product must be installed and configured, and the available files must be brought into the system. For Unison, various configurations are possible. One of the computers is selected to run the Unison software, while the other computers must be made accessible from this computer (which depends on firewall settings and installed system software). The laptop may be a good candidate, as it moves in the university as well as the home network, and then can initiate synchronization with the PCs. To start, all three computers must have an identical copy of the file tree, which can be accomplished by a first run of Unison from the laptop to the home PC plus a first run against the university PC (assuming that the laptop holds all data to begin with). This initial synchronization process can be quick, because all communication goes over local area networks.

The Dropbox client must be installed and run continuously on all three computers and must login to the Dropbox service under a single account. The amount of data to be stored online requires a paid subscription of $10 per month, providing a storage capacity of 50 GByte. Data can be added to the Dropbox folder from all machines, and this data will be available at all machines after synchronization over the network. Dropbox provides a local sync features which supports the direct data transfer between two computers on the same LAN. Using this form of synchronization, new data must still be uploaded to the back-end, but not downloaded to individual computers that do not have these new files yet. So, LAN sync reduces the amount of data to be downloaded over the Internet.

For Wuala (like Dropbox), each computer must run the Wuala client and login to the network (using the same credentials). One of the machines, preferably the university PC, must trade disk storage so that capacity at peers becomes available. Now, all computers may add data to the online file tree, which eventually shows up at all machines, although upload may take considerable time (especially when uploading a lot of data from the home network). This is also the case for Dropbox, although Wuala transmits more data to create redundancy at peers and to sync between computers. Clearly, configuring Dropbox and Wuala is much more straightforward than configuring Unison.

5.2 Regular activities and maintenance

To keep the files synchronized and in backup, the products must be managed from time to time. Typically, Unison requires the student to execute a synchronization task from the laptop against both PCs, although it is possible to schedule this task at regular intervals. This requires the machines to be online at a specific point in time, so synchronization may fail if this is not the case.

The Dropbox and Wuala clients will take care of synchronization automatically, as long as the computers are online regularly (not necessarily all at the same time). Care must be taken, however, that the amount of new data generated does not exceed the available storage as part of the subscription (for Dropbox) or obtained trough trading (for Wuala). Additionally, the university PC with capacity for other peers must be online for a sufficiently high fraction of the time, because the amount of available online storage depends on this fraction. Wuala generates more Internet traffic to have redundant storage at peers.

5.3 Recovery

When one of the computers crashes and loses all data, Unison can be used to retrieve the data available on the other two machines. Even in the case of a fire, data is still safe because it is stored on machines at different locations. Only in case of disasters like earth quakes, all three machines may be lost at the same time and therefore the data may be lost. The student will not be able to go back to previous versions of files, because versioning is not supported by Unison. Running a versioning tool such as CVS or SVN in combination with Unison may be suitable to solve this problem. This does, however, require additional attention from the user.

For Dropbox and Wuala, simply install the client on the new machine (replacing the crashed machine) and start using the online files. Dropbox first retrieves all data from online storage, which may take a long time to finish, or it uses LAN sync with another computer (which is much faster and comparable to Unison). Note that for Wuala files are downloaded on demand, so that the first usage of big files may take considerable time to retrieve. Old versions of files can be retrieved through the client for Wuala and Dropbox.

6 Conclusions

In this document, we have introduced various technologies used for cloud storage and P2P storage services and discussed a number of products that are currently available to end users. Although cloud storage and in particular P2P storage software is relatively new and does not have an extensive track record (not counting the basic tools), it may be useful for SURFnet end users that require backup, synchronization or sharing of their data. Cloud storage services store user data at server farms connected to the Internet. P2P storage works by trading redundant resources in the form of disk capacity and bandwidth.

Contrary to cloud storage services, P2P storage does not require a paid subscription. From many perspectives, however, cloud storage and P2P storage provide similar functionality and also must deal with similar problems. For example, both types of storage must deal with intermittent network connectivity, and must choose whether file operations are executed on online data or on local copies of the data (that get synchronized with their online counterparts later on). An important advantage of cloud storage is the guaranteed availability of the data (as long as the subscription is paid), while the availability of P2P storage capacity at peers depends on the provisioning of own resources (i.e., requires keeping a machine online with sufficient bandwidth and disk space).

Cloud storage services are currently available in many variants: the cloud storage market is growing and products are still evolving rapidly. Some services are part of a larger web platform and integrate with other facilities such as web servers, payment facilities, and online document editing, and primarily target developers of web applications. Examples of this category are Amazon S3 and Microsoft Live Mesh. Other services, such as Dropbox and Mozy, are more focused on end users or particular kinds of storage such as backup. In many cases, the design of the back-end systems of these services is not disclosed, although it is likely that this design is based on principles also used for P2P storage.

We observed that there are essentially two P2P storage product models: the ‘full’ P2P storage systems that may scale to many peers, and the manual, ‘social’ P2P networks that are small and consist of computers of other known people. Each type of product has its own features, making them suitable for specific situations. Currently, the only easily usable full P2P product with a rich set of features is Wuala. For the manual P2P products, both CrashPlan and PowerFolder are feature rich and mature.

It is interesting to note that a strict division between cloud storage and P2P storage systems is often not possible, because many products incorporate elements from both technologies. For instance, Dropbox has a LAN sync option to support direct (peer) synchronization between computers. And PowerFolder and CrashPlan also offer possibilities to store data online.

The products and services discussed in this document show that cloud- and P2P storage is now offered to end users in many variants, providing solutions to many different needs. Also, it is clear that the market for these products is highly dynamic and innovative, and is driven by the continuous increase in Internet connectivity speed and quality and decreasing prices for (hard disk) storage capacity. It is therefore likely that new features, products, and services will keep appearing in the nearby future, and that existing products will become more and more mature, offering better quality and choice to the user.

References

[1] ADrive cloud storage, http://www.adrive.com/
[2] Allmydata: Unlimited Online Backup, Storage, and Sharing, http://www.allmydata.com/
[3] Amazon Simple Storage Service (S3), http://aws.amazon.com/s3/
[4] S. Androutsellis-Theotokis and D. Spinellis, A Survey of Peer-to-Peer Content Distribution Technologies, *ACM Computing Surveys*, Vol. 36, pp. 335-371, 2004
[5] M. Beck, J.-P. Gelas, LoDN: Logistical Distribution Network, In *Proceedings of the Workshop on Advanced Collaborative Environments*, Sept. 2004
[6] BitTorrent Protocol Specification, http://wiki.theory.org/BitTorrentSpecification
[7] CrashPlan – Automatic Online Backup, http://www.crashplan.com/
[8] Cucku – Social Backup, http://www.cucku.com/
[9] F. Dabek, F. Kaashoek, D. Karger, R. Morris, and I. Stoica, Wide-area cooperative storage with CFS, In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP'01)*, October 2001
[10] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels, Dynamo: Amazon's Highly Available Key-Value Store, In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07)*, October 2007
[11] Dropbox, http://www.dropbox.com/
[12] FileSender, http://www.filesender.org/
[13] S. Ghemawat, H. Gobioff, and S.-T. Leung, The Google File System, In *Proceedings of the 19th ACM Symposium on Operating System Principles (SOSP'03)*, October 2003
[14] Gnutella Protocol Specification, http://gnet-specs.gnufu.net/
[15] Google Apps, http://www.google.com/apps/
[16] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer, Cryptree: A Folder Tree Structure for Cryptographic File Systems, In *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS'06)*, October 2006
[17] Hadoop Distributed File System, http://hadoop.apache.org/hdfs/
[18] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems, In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, Vol. 2, pp. 205-213, 2005
[19] M. Landers, H. Zhang, and K.-L. Tan, PeerStore: Better Performance by Relaxing in Peer-to-Peer Backup, In *Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, 2004
[20] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, A Cooperative Internet Backup Scheme, In *Proceedings of the USENIX Annual Technical Conference (USENIX'03)*, 2003
[21] Microsoft Windows Live Skydrive, http://skydrive.live.com/
[22] Microsoft Live Mesh, http://www.mesh.com/
[23] Mozy – Safe, Secure, Affordable Online Backup, http://mozy.com/
[24] A. Muthitacharoen, R. Morris, T. Gil, and B. Chen, Ivy: a read/write peer-to-peer file system, *SIGOPS Oper. Syst. Rev.*, 36(SI):31–44, 2002
[25] PowerFolder – Sync your World, http://www.powerfolder.com/
[26] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, Pond: the OceanStore Prototype, In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, March 2003
[27] Rsync, http://samba.anu.edu.au/rsync/

[28] S. Seuken, D. Charles, M. Chickering, and S. Puri, Market Design & Analysis for a P2P Backup System, In *Proceedings of the Workshop on The Economics of Networks, Systems, and Computation (NetEcon'09)*, 2009
[29] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, In *Proceedings of SIGCOMM*, 2001
[30] A. Rowstron and P. Druschel, Pastry: Scalabe, distributed object location and routing for large-scale peer-to-peer systems, In *Proceedings of IFIP/ACM Middleware*, 2001
[31] A. Rowstron and P. Druschel, Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility, In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP'01)*, October 2001
[32] SkyDrive Explorer, http://www.skydriveexplorer.com/
[33] Tahoe: A Secure Distributed Filesystem, http://allmydata.org/~warner/pycon-tahoe.html
[34] Tahoe – The Least-Authority Filesystem (LAFS), http://allmydata.org/
[35] D. Wallach, A Survey of Peer-to-Peer Security Issues, In *Proceedings of the International Symposium on Software Security*, 2002
[36] H. Weatherspoon and J. Kubiatowicz, Erasure Coding vs. Replication: A Quantitative Comparison, In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, LNCS Vol. 2429, pp. 328-338, 2002
[37] Wuala, Secure Online Storage, http://www.wuala.com/
[38] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, Tapestry: An Resilient Global-Scale Overlay for Service Deployment, IEEE Journal on Selected Areas in Communications, Vol. 22, No. 1, January 2004