# PSCloud: A Durable Context-Aware Personal Storage Cloud

Sobir Bazarbayev[†], Matti Hiltunen[×], Kaustubh Joshi[×],
William H. Sanders[†], and Richard Schlichting[×]

[†]University of Illinois at Urbana-Champaign, [×]AT&T Labs Research

## Abstract

Personal content from mobile devices is often irreplaceable, yet current solutions for managing and synchronizing this data across devices to ensure durability are often limited. For example, one option is to use a cloud storage service such as Dropbox to store a master copy and then synchronize each device with this copy. While common, this model is excessively rigid, forces users to use more expensive cloud storage than is needed, and is oblivious to the different costs of network access by device and time of day. This paper proposes an alternative approach that uses storage on all of a user's mobile devices, home servers, and cloud storage accounts to create a single unified personal storage system called PSCloud in which data is automatically cached, replicated, and placed to enable reliable access across all devices while minimizing network access and storage costs. This approach is based on a per-device *network context-graph* that tracks connectivity relationships between a user's devices and storage options over time. Preliminary experiments show that combining such context with techniques that exploit content similarity across devices to make placement decisions can lead to substantial reductions in cloud storage and network usage.

## 1 Motivation

Mobile devices have become the primary computing platform for millions of consumers, with individuals often owning and using multiple devices such as smart phones, tablets, and laptops. Moreover, these devices are no longer used just to consume content, but also to produce and modify personal content, such as pictures, videos, and documents. The personal nature of this content makes it highly valuable to the individual—often irreplaceable—meaning that it must be backed up to protect against device loss or theft, and its access enabled across all of the user's devices. This represents a significant challenge given the increasing volume of a person's digital content, the limited storage capacity of mobile devices, and the wide range of external storage options in terms of connectivity and cost.

Most current approaches to addressing this problem fall into one of two categories, neither of which is completely satisfactory. The first is for the user to replicate and distribute their content across their devices manually. Doing so places the burden on users, forcing them to track the location and state of their content, including which copies have been modified and when. Historically, users could simplify matters by relying on a single master copy of their content on a single device—often their home desktop—and treating their mobile devices as a cache, but the increasing number and prevalence of mobile devices is making this option increasingly less feasible. It also does not automatically solve the backup issue.

The second approach is to use a cloud storage service such as Dropbox, Google Drive, or Box to store a master copy of the content, and then synchronize devices to this cloud store. While often providing better durability, it is not a panacea. For one thing, today's cloud-based synchronization is often an all or nothing proposition. That is, a master copy of all data is expected to reside in the cloud, and, with the exception of some domain-specific solutions such as iTunes Match, either all of it is synchronized with each device, or the user must manage the data manually. Another issue is that cloud storage applications typically ignore the cost of network access completely. This can lead to less-than-optimal scenarios, such as unexpected and excessive use of a limited cellular data plan to synchronize data.

In this paper, we propose an alternative vision of an intelligent *personal storage cloud* called PSCloud that can automatically enhance the durability of a user's personal content and dramatically simplify its management. The key aspects of the approach are as follows.

**Uses Diverse Storage Assets.** Our approach utilizes the storage available on all of a user's mobile devices,

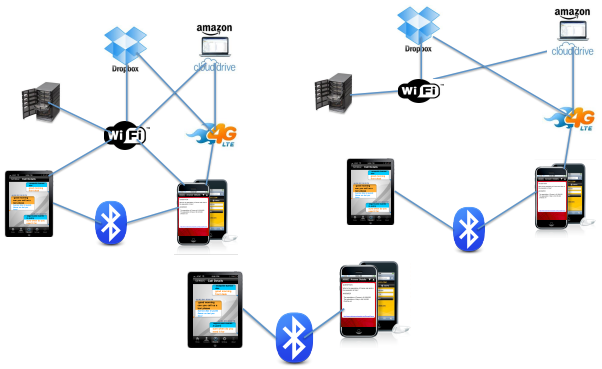


Figure 1: Usage scenarios

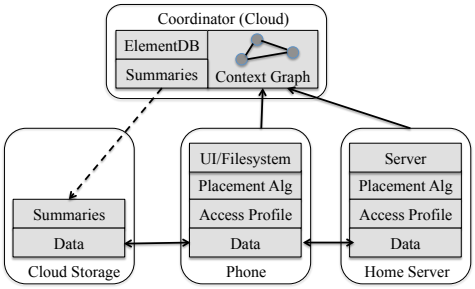


Figure 2: PSCloud architecture

cloud storage services, and home servers in a cost-sensitive way. It provides a global namespace, but automatically replicates and distributes content across the various storage options.

**Performs Intelligent Content Management.** The mapping of content to storage is done intelligently based on cost, capacity, and usage patterns. For example, our approach preferentially uses storage owned by the user on their mobile devices or home server over storage that incurs additional costs such as cloud storage. It also leverages *usage context* to optimize content placement, that is, takes into account which files are accessed when and on which devices.

**Is Network Aware.** Our approach recognizes that different types of storage are accessible through different network interfaces (e.g., WiFi, cellular, Bluetooth) with differing cost profiles at different times of the day. It uses these different network connections in a cost effective manner by prioritizing storage nodes that can be accessed cheaply at a given time.

**Exploits Content Similarity.** Our approach leverages content similarity across devices to enhance durability and reduce data transfer. Specifically, it uses a cost model that reflects the combined effects of storage cost, network cost, and content similarity to make optimal content placement decisions. Recent surveys [9], [13] suggest that many households own multiple mobile devices, and we postulate that there is a large degree of similar content among those devices. If that similarity can be efficiently and effectively tracked in an online manner, it can provide an excellent source of ready-made data redundancy for content backup, as well as reduce the amount of network bandwidth needed.

To illustrate our goals, consider three different scenarios in which a user's mobile devices, home storage, and cloud storage services might be connected using networks with greatly varying performance and cost characteristics, as shown in Figure 1. In the "at home" scenario at top left, all mobile devices and storage nodes are accessible through a relatively low cost and high performance WiFi network. In addition, there is plentiful local storage available through the user's home server. In the "on the road" scenario at the top right, connectivity is more limited; for example, the smartphone can only communicate with cloud storage or a home server through a cellular network that may charge based on bytes transferred, while the tablet can only directly communicate with the smartphone using Bluetooth. Finally, the "international travel" scenario at the bottom is an extreme situation where high roaming costs force a user to rely solely on local connectivity options such as Bluetooth between devices carried on the trip. The time-dependent graph that encodes this connectivity between all of a user's devices and storage options is called the *network context graph*. Given such a graph, we can find times throughout the day when paths with the lowest costs are available and can schedule content synchronizations at such times to minimize cost. Alternatively, we are also able to find the lowest cost path at any given time to choose storage from among different options.

The remainder of this vision paper describes the architecture of PSCloud and provides preliminary results that quantify some of the savings that can be expected from its use.

## 2 Architecture

The architecture of PSCloud is shown in Figure 2 and contains one or more mobile devices, cloud storage services, and home storage servers along with a single coordinator. Each of these element types have unique characteristics.

*Mobile devices.* Are the only type of elements that can be used to both store and consume content, and run computation. They have a fixed and moderate amount of storage, and multiple connectivity options (WiFi, Cellular, Bluetooth). They are also the most likely to be partitioned from other elements.

*Home/Office Servers.* Can run computation and store large amounts of data cheaply, but have a fixed upper limit. They are always connected to cloud storage, but

their connectivity to mobile devices depends on the network context; low cost, high bandwidth connectivity is available only if the mobile device is on the same local network as the server.

*Cloud Storage.* Does not have a fixed storage capacity, since user can pay to get more. The amount of storage cost-effectively available may be more than mobile devices, but less than home servers. Unlike mobile devices or home servers, cloud storage elements cannot run any computation.

**The Coordinator.** The coordinator is a distinguished cloud server that can perform some computation and store a small amount of state. The coordinator maintains systemwide state that is used by all the elements. This includes:

*a)* The *elementDB* is a list of all storage elements, their network interfaces, and their cost functions. Storage elements are registered with the coordinator. The cost function for a storage element specifies the cost the user incurs for using a particular amount of storage on that element. For elements with a fixed amount of storage, this is a step function - zero cost until space is available, and infinite cost thereafter. For cloud storage, the cost function represents the actual cost of storage tiers, e.g., zero cost until 5GB, $10/month for the next 10GB, and so on.

*b)* The network context graph is a multigraph where the vertices are storage elements, and the edges are network links, e.g., Bluetooth, LAN, WLAN, WiFi, cellular, that connect the storage elements as shown in Figure 1. The network context graph changes over time, and represents the state of connectivity of the user's devices at a given instant. Each network link is annotated with its estimated bandwidth, and with a network cost function that specifies the cost of the network based on the amount of data transferred, e.g., a cellular network may have a monthly cap of 2GB that is "free" with each subsequent GB costing $10, while a WiFi hotspot may have a fixed price independent of the amount of data transferred.

The coordinator also maintains a historical *availability profile (avlprofile)* for each edge in the graph. The avlprofile tracks the probability that the edge is available at different times of the day, separately for weekdays and weekends. A periodic background task on each mobile storage element reports network availability needed to construct the avlprofile. Thus, no direct edge can exist between storage elements such as cloud storage services that do not have any ability to run compute. Using the avlprofile, the PSCloud finds patterns in users' everyday lives of when they will have high speed, low cost access to various storage nodes, e.g., when the user is likely to be in the same local network as the home server.

*c)* The *Summaries* is a "master copy" of the PSCloud's global directory, file listings, and Bloom filters that summarize the contents of each storage element, and the free space on each element. The global directory list contains infrequently changing file metadata such as name, type, and permissions, and omits inode metadata that is likely to change frequently, such as atime, ctime, and file size. Each file may also be associated with a *file access profile* (described below), and a *dlist*, a list of storage elements which have a copy of the file. Both attributes are inherited - files without a dlist or file access profile use the closest ancestor value. The Bloom filters [3] for each storage element contain hashes of the files the element contains. They are used to make placement decisions based on content similarity. When creating a copy of a file, picking storage elements where similar files already exist leads to significant reductions in network use. To avoid the need for storage at the coordinator, it uses a cloud storage node to store these summaries.

**The Storage Client.** The storage client runs on mobile devices and provides the user with a frontend to the PSCloud. It also performs several tracking and optimization functions. Our current client implementation is a Dropbox-like app for Android. Users explicitly enroll the files they want the PSCloud to manage. Future implementations may integrate the client into Android as a native filesystem.

*a)* A *copy of coordinator data structures* is maintained by each storage client structures so that file replication and migration decisions can be made locally. These structures are lazily fetched by mobile devices, and lazily updated by storage elements. Thus, it is possible for elements to have out-of-date copies. Such inconsistencies are tolerable for the elementDB and the context graph because they are used by the mobile devices simply as hints, not as ground truth. For example, if the context graph contains a storage element that is no longer accessible, the mobile device simply updates its local context graph and recomputes the placement. However, for directory and file data, inconsistencies are not acceptable. For these, we follow the approach developed by Coda [7]: resolve metadata inconsistencies transparently to the extent possible, and notify the user of conflicts that cannot be resolved automatically. Because the focus is on providing a *personal* filesystem, the assumption is that the user would use one device at a time, so we believe such a design is reasonable.

*b)* The *File Access Profile*, or *faprofile* is a summary of how frequently the files in a directory (or a single file) are read, modified, created, or deleted. faprofiles are maintained by a background task on each mobile device, and allow the placement algorithm to compute the cost of various placement options more effectively. For example, files that are read-mostly can be replicated on multiple mobile devices without penalty. Frequently updated files

may be preferentially backed up on cloud nodes with good network connectivity rather than on home servers, and infrequently accessed files may be ejected from mobile devices to free up more space. The background task that computes faprofiles also updates the Bloom filters as needed.

The faprofile may also include a user specified *vulnerability window*, which is the interval of time the file/directory may be retained on a single storage element without a backup. A window of zero indicates that the file must be synchronously updated. A longer window allows the placement algorithm to use cheaper storage and network options that may become available in the future, based on the avlprofile.

*c)* The *File Placement Algorithm*'s goal is to determine on which storage elements to place files so as to utilize all available storage, minimize the total storage and the network costs, and still respect the file's vulnerability window requirement. Using the various profiles described above, fairly sophisticated placement algorithms can be constructed, and we intend to explore them in future work. Our current implementation uses a simple scheme. If a non-local file is accessed from a mobile device, it is fetched, and the summary data structure is updated to reflect a new copy of the file. If more storage is needed on a mobile device, infrequently accessed files that have more than two copies are evicted.

When picking a storage element on which to place a new copy of file, the edges of the context graph from the current mobile device are examined, and the edge that minimizes "cumulative cost" is picked. The cumulative cost, defined as (NetworkCost(DataSize) + StorageCost(DataSize)) / Probability(Edge will occur within vulnerability window), includes both network and storage costs, and allows for network connectivity options that occur in the future, but are within the file's vulnerability window. The DataSize is computed using content similarity between the file and the potential target (using the Bloom filters). Our current implementation only uses single hop cost functions. Extending it to multiple hops is left for future work.

## 3 Preliminary Results

We have performed initial measurements to evaluate the feasibility of PSCloud and evaluate its possible benefits. First, to examine realistic network context graphs, we observed two users availability profile for one week including what networks the users' devices had access to, and when these devices were co-located. One user (User 1) had a WiFi-only Nexus 7 tablet and an Android 3G phone, and the other user (User 2) had a WiFi-only Nexus 7 tablet and an FTP server on his home network.

| Time | T1 W | P1 W | P1 Cell | T1 - P1 B | T2 W | T2 FTP |
|---|---|---|---|---|---|---|
| 24 - 4 | 0.44 | 0.66 | 0.33 | 0.92 | 0.84 | 0.82 |
| 4 - 7 | 0.44 | 0.75 | 0.25 | 0.95 | 0.86 | 0.86 |
| 7 - 8 | 0.50 | 0.50 | 0.50 | 0.63 | 0.86 | 0.86 |
| 8 - 9 | 0.38 | 0.90 | 0.10 | 0.63 | 0.86 | 0.86 |
| 9 - 10 | 0.33 | 0.69 | 0.31 | 0.63 | 0.87 | 0.73 |
| 10 - 11 | 0.38 | 0.58 | 0.42 | 0.60 | 0.87 | 0.73 |
| 11 - 12 | 0.38 | 0.42 | 0.58 | 0.47 | 0.71 | 0.64 |
| 12 - 13 | 0.38 | 0.36 | 0.64 | 0.31 | 0.61 | 0.61 |
| 13 - 14 | 0.38 | 0.58 | 0.42 | 0.12 | 0.75 | 0.36 |
| 14 - 15 | 0.62 | 0.29 | 0.71 | 0.12 | 0.77 | 0.54 |
| 15 - 16 | 0.38 | 0.25 | 0.75 | 0.12 | 0.71 | 0.50 |
| 16 - 17 | 0.38 | 0.36 | 0.64 | 0.12 | 0.71 | 0.53 |
| 17 - 19 | 0.38 | 0.42 | 0.50 | 0.12 | 0.77 | 0.38 |
| 19 - 24 | 0.44 | 0.60 | 0.40 | 0.70 | 0.75 | 0.52 |

Table 1: Hourly network profiles.

Both users had WiFi networks at home and at the office.

The network context over the week is presented in Table 1. For each of the six listed types of network-device connectivity, the table gives the percentage of each time window during which that connectivity was observed averaged over five week days. Time intervals for which similar results were observed are grouped together. The "T1 W" column shows the average percentage for Tablet 1 (of User 1) had WiFi access; this tablet had WiFi access less than 50% of the time for most time intervals. This user's Android phone was connected to a WiFi network ("P1 W" column) more than to a cellular network ("P1 Cell" column) during the night, but at certain times of the day (11:00 a.m. to 7:00 p.m.), the phone was connected to cellular network coverage more than 50% of the time. The "T1 - P1 B" column shows (for each time interval) the fraction of days during which this user's Android phone and tablet were within Bluetooth range of each other.

The "T2 W" column shows the WiFi connectivity of Tablet 2 (of User 2). The tablet had over 70% WiFi access during all time intervals except 12:00 to 1:00 p.m. The reduced WiFi access over the noon hour may reflect the user's tendency to step out for lunch with his tablet. Finally, the "T2 FTP" column shows what fraction of the time the user's tablet was on the same WiFi network as the user's home server.

Next, we explore the traffic reductions possible by tracking content similarity between a user's devices. To do so, we used the PSCloud's Bloom filter implementation to compare the overlap between data stored on the phones and laptops of two of our colleagues. The first user had two laptops, one for personal use having 25GB of data, and the other for work use with 135GB of data. After comparing the content similarity at a page block level, 10.9GB of the data on the two computers was the same, that is, 43.6% of the smaller devices data was already contained on the larger device. The second user had a phone with 4.5GB of data and a laptop with 65GB of data. For this user, content similarity analysis revealed that 4.3GB of data from the phone, over *95% of the content on the phone*, was also present on the laptop. While

far short of a representative study, these anecdotal examples strongly suggest that significant amounts of similar content—movies, pictures, music, and documents—exist on multiple devices owned by the same person or from the same household. As long as this similarity is tracked, there is no need to explicitly create a backup copy of data that is already present on another device, thus leading to substantial network savings.

Combining these results, we can estimate how much savings in cloud storage and cellular network usage the PSCloud could deliver. For example, assume a user has a smart phone with 32GB of storage, a tablet with 128GB of storage, and a home server with 1TB of disk. The user could naively choose to back up the mobile devices in an on-line cloud storage (160GB total). However, just taking advantage of the content similarity (assume 40%), 51GB of the data on the tablet is already included on the home server and 25GB of the data on the smart phone is already included on the tablet (or home server). Thus, only the remaining 74GB would need to be backed up in the cloud storage. Furthermore, if we actively use the home server to back up both of the mobile devices when the devices are at the home wireless LAN and use the tablet to back up the smart phone when away from home but co-located and reachable by Bluetooth, the demand for on-line cloud storage can be reduced to the size of the new content created on the mobile devices that is so critical that it cannot wait to be backed up on the home server (or the other mobile device). This storage requirement would likely be satisfied by the introductory free first tier of on-line cloud storage providers.

Finally, we evaluated the file access patterns on an Android tablet over one month of usage. We found that while some files were read and updated thousands of times, 60% of the files (out of 750) were accessed 20 times or less.

## 4 Related Work

PSCloud is unique in its combination of the following properties: a) a unified namespace that allows users to transparently use storage across heterogenous backends, some of which may be disconnected, b) automatic file placement that leverages file access patterns, network context, and storage costs to use multiple storage backends in a cost-efficient way, and c) the use of content similarity between data on multiple devices to reduce storage needs and improve data availability. While past work in distributed filesystems, P2P back up solutions, and cloud storage services provides a subset of these properties, we are not aware of any systems that provide them in combination.

Cloud storage services such as Google Drive and Dropbox provide a limited amount of free storage beyond which users can pay for additional capacity. They provide a partial unified namespace by allowing users to synchronize cloud data with their mobile devices on a folder-by-folder basis. Data must be manually partitioned by users into local vs. shared data. For shared data, the cloud is the "master copy", and storage across all of a user's devices cannot be utilized. Online backup services such as MyBackup [2] and BullGuard [1] allow users to backup selected content to cloud storage. However, there is no unified namespace across devices, nor is there any ability to utilize storage from another device. Distributed filesystems such as AFS [6] and GlusterFS [11] allow the use of storage at multiple heterogenous backends, but they do not support disconnected operation. Filesystems such as Coda [7] do support disconnected operation, but they again require that the file server act as a master copy. BlueFS [10] builds on techniques developed in Coda to efficiently integrate portable storage in an energy efficient way.

Peer-to-peer storage systems combine storage at multiple mobile devices to create a unified abstraction. For example, Venti [12] is a network archival storage system that provides a write-once repository that can be shared by multiple client machines. It uses deduplication to ensure that each unique block is stored only once. [14] combines Venti with DHTs to provide a P2P backup solution. OceanStore [8] and CFS [5] are other P2P archival storage systems that use DHTs. Pastiche [4] is a P2P backup system in which peers utilize free disk space on other machines. Nodes minimize storage overhead by selecting peers that share a significant amount of data. Finally, [**?**] presents a P2P storage system designed specifically for dynamic collections of power-constrained mobile devices on personal area networks. It considers factors such as available power and storage as well as pair-wise locality.

To our knowledge, none of the approaches described above use contextual information about file access patterns, network, and storage costs to automatically distribute files so as to optimize costs.

## 5 Conclusions

We have presented a vision of a context-aware personal storage cloud called PSCloud that takes advantage of all of a user's devices as well as on-line cloud storage accounts to provide a seamless and cost-efficient personal storage and backup solution. We rely on context information related to the network, storage nodes, and file usage in order to optimize where to place files and when. Initial results indicate that the approach can result in significant cost savings for the user, while providing a higher-level abstraction and simplified content management.

## References


[1] BullGuard Mobile Backup. http://www.bullguard.com/products/-bullguard-mobile-backup-12.aspx.

[2] My Backup Pro. https://play.google.com/store/apps/-details?id=com.rerware.android.MyBackupPro.

[3] BRODER, A., MITZENMACHER, M., AND MITZENMACHER, A. Network applications of Bloom filters: A survey. In *Internet Mathematics* (2002), Citeseer.

[4] COX, L. P., MURRAY, C. D., AND NOBLE, B. D. Pastiche: Making backup cheap and easy. *ACM SIGOPS Operating Systems Review 36*, SI (2002), 285–298.

[5] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with CFS. *ACM SIGOPS Operating Systems Review 35*, 5 (2001), 202–215.

[6] HOWARD, J., KAZAR, M., NICHOLS, S., NICHOLS, D., SATYANARAYANAN, M., SIDEBOTHAM, R., AND WEST, M. Scale and performance in a distributed file system. *ACM Trans. Comput. Syst. 6*, 1 (Feb. 1988), 51–81.

[7] KISTLER, J. J., AND SATYANARAYANAN, M. Disconnected operation in the coda file system. *ACM Trans. Comput. Syst. 10*, 1 (Feb. 1992), 3–25.

[8] KUBIATOWICZ, J., BINDEL, D., CHEN, Y., CZERWINSKI, S., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., ET AL. Oceanstore: An architecture for global-scale persistent storage. *ACM Sigplan Notices 35*, 11 (2000), 190–201.

[9] MITCHELL, A., ROSENSTIEL, T., SANTHANAM, L. H., AND CHRISTIAN, L. The explosion in mobile audiences and a close look at what it means for news. http://www.journalism.org/-analysis_report/future_mobile_news.

[10] NIGHTINGALE, E. B., AND FLINN, J. Energy-efficiency and storage flexibility in the blue file system. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation* (2004), pp. 363–378.

[11] NORONHA, R., AND PANDA, D. Imca: A high performance caching front-end for glusterfs on infiniband. In *Proc. IEEE 37th International Conference on Parallel Processing (ICPP)* (2008).

[12] QUINLAN, S., AND DORWARD, S. Venti: A new approach to archival storage. In *Proceedings of the FAST 2002 Conference on File and Storage Technologies* (2002), vol. 4.

[13] SHANKLAND, S. Google: 500 million Android devices activated. http://news.cnet.com/8301-1035_3-57510994-94/google-500-million-android-devices-activated/.

[14] SIT, E., CATES, J., AND COX, R. A DHT-based backup system. In *Proceedings of the 1st IRIS Student Workshop* (2003), vol. 131, p. 146.