

# A Distributed Social-Networking Infrastructure with Personal-Cloud Butlers

Seok-Won Seong, Sudheendra Hangal, Chris Brigham, Debangsu Sengupta, Greg Bayer, Jiwon Seo, Juan Batiz-Benet, Karl Uhlig, Snir Kodesh, Monica S. Lam  
Computer Science Department, Stanford University  
prpl@cs.stanford.edu

## ABSTRACT

The convenient way for users to share data these days, be it photos, IM, or just their GPS locations, is for them to sign up as friends to the same social networking sites. As a result, not only are we inundated by invitations from our friends to join a large number of different networks, we lose the privacy of our personal relations, and end up with our data scattered across many different web sites. The problem of having to deal with disparate data repositories remains even when we are not sharing data. In addition to our various PC's, today's cameras and cell phones can generate and hold gigabytes of content. Further, mail servers contribute to data overload by storing our large e-mail attachments.

This paper introduces the notion of a Personal-Cloud Butler, a program that presents a coherent view of data stored in a federation of distributed devices and supports sharing of private data among friends. Central to the system is a PrPI (PRivate-PuBLic) semantic index, which is used to co-ordinate all the storage devices. With the help of URIs (Uniform Resource Identifiers), the protocol eliminates data copying by allowing applications to retrieve data directly from storage. The Personal-Cloud Butler and the PrPI semantic index are analogous to the browser and HTML, except that the former is designed as an API operating on private and shared data and the latter is designed for visual display of public data. By providing active mediation of access to a distributed collection of data, we enable new applications that integrate together information from federated data stores while honoring data privacy.

We have implemented a prototype of the Personal-Cloud Butler. We have also developed a collection of applications to illustrate the benefits of this technique. Our applications range from shared contact information, movie suggestions based on friends' calendar and public information, friends' restaurant reviews, collective memory of photos taken by a group of people, and personal feeds. The key benefit of the infrastructure is that these applications can be developed with just a few hundred lines of application-specific code.

## Keywords

Social networking, Cloud computing, Mobile devices

## 1. INTRODUCTION

Social networking applications have greatly improved our

Copyright is held by the author/owner(s).  
WWW2009, April 20-24, 2009, Madrid, Spain.

way of interacting with friends in our digital life. Consumers today have a myriad of web services to choose from, many of which are free. Free data storage services provide highly available data that can be shared with anyone, any time and from anywhere. There are a large number of social networking applications that help us make new friends and keep up with our existing ones. In addition, many software applications such as Picasa, Photoshop and Google docs operate on our data in the cloud.

Cloud computing services are primarily *application-centric*. They offer a range of services, from narrow ones like sharing current GPS locations among friends, to broader ones like Facebook, where users have the ability to even define new applications. Users are required to upload their personal relationships and data according to the services they desire. However, this application-centric model has a number of drawbacks, including data lock-in, loss of privacy and inconvenience to users by requiring them to manage data scattered across diverse sites.

This paper attempts to address these deficiencies by proposing a *person-centric* social-networking infrastructure, where a person's data is logically collected in one place, access control policies are used to restrict access, and social-networking applications can be executed in a distributed manner anywhere without a central service.

### 1.1 Centralized Application-Based Social Networking

Social networking applications are very different from traditional web applications aimed at disseminating information to the general public. They operate on an individual's personal cloud of data, their relations, and their private data, which are only shared with a relatively small number of users. The use of an application service provider model for social networking leaves a lot to be desired, as described in more detail below.

**Disparate data sources.** While plenty of companies are vying to store all our data in the cloud, the reality is that most people's data still resides on their home computers. The uplink bandwidth of broadband connections and its projected improvement is no match for our ability to capture photos with ever-increasing resolution.

Application-based cloud services require us to explicitly upload data to different web sites. It is difficult to keep track of all the data, given that it is scattered across diverse devices and web sites, depending on how the information is acquired

and what kind of function we wish to perform on it. Just take photos alone - they can still be stored on our camera or cell phone, saved on a personal computer, attached to an email on one of our many email accounts, uploaded to Flickr or Facebook to share with my friends, or uploaded to Shutterfly to be printed.

**Inconvenience.** The application-based model requires us to inform each of the sites hosting our desired applications of our relations and to upload the data. Not only that, sharing with our friends involves providing them links to multiple services, which often require separate registration.

**Data lock-in, loss of data ownership and privacy of our relations.** Besides inconvenience, we are losing the privacy of our relations and data. In many cases, we only have very coarse-grain control over who can access our data. Companies that offer a broad range of services, such as Google and Facebook, can learn quite a lot about individuals. Furthermore, it is difficult, if not impossible, to move our data if we desire to migrate to other services.

**Scaling difficulty.** In this model, an application service provider runs a central service providing authentication and authorization, collecting everybody's information even though a given piece of information is shared with only a few friends at a time. If the service becomes popular, it requires the service provider to scale out quickly. Better efficiency can be obtained with a distributed implementation.

Data lock-in and difficulty in scaling both create a barrier to entry and hence reduce competition.

## 1.2 A Person-Centric Social-Networking Application Infrastructure

Instead of a centralized application-based approach to social networking, this paper proposes a very different model that is distributed and person-centric. Developing distributed applications is notoriously difficult. We propose a personal-cloud computing infrastructure that enables rapid application development over distributed data without compromising security and privacy.

The infrastructure we are proposing, named PrPI (for PRivate-PuBLic), provides the following programming abstractions.

**A unified view over a federated storage system.** We fully embrace the consumers' desire to take advantage of the many web services available. To that end, the PrPI infrastructure creates a federated storage system out of existing services and storage devices. The user can continue to take advantage of free e-mail services and access data they have uploaded on Facebook. At the same time, he can choose to place sensitive data only on his PC at home, with possibly an encrypted backup in the cloud. Once the user registers the various data repositories, the infrastructure automatically tracks the changes, requiring no more human intervention. Our infrastructure presents a unified, location-agnostic view of the data in a user's personal cloud.

**Person-centric with fine-grain access control.** Instead of uploading data to be shared to specific sites, a user would simply specify, at fine granularity if desired, with whom he

wishes to share his content. A user can run a PrPI application from anywhere. Once he presents his credentials, the application gains access to all of his data as well as those shared by his friends. This makes it easy to develop applications that require access to large collections of information owned by different people.

**High-level semantic queries.** Many of the emerging social networking applications, especially those on mobile devices, can be framed as a semantic query on data owned by a group of friends. Many of the top 50 Android applications have that flavor. Some well-known examples of such social applications include Loopt, which allows friends to share their current GPS locations, and Zintin which looks up friends in nearby locations. The PrPI infrastructure creates a semantic index of all the data and presents to applications a semantic query interface that facilitates retrieval of relevant information.

**Common data navigation infrastructure.** A large number of lines of codes in social networking applications are devoted to user interfaces. Especially in mobile devices where the screen real-estate is relative small, we believe that users should be allowed to actively navigate the resulting data rather than be presented with a passive view. The presence of type information in the semantic index allows generic data visualization tools to be used by PrPI applications in our infrastructure.

## 1.3 Key Concepts

There are two main concepts in the PrPI personal-cloud computing infrastructure. The first is the *PrPI Semantic Index*, which is a cross between a semantic data base and a semantic file system, and is used for tracking data in a personal cloud. The second is the *Personal-Cloud Butler*, which is a program that actively manages a user's personal cloud. Just like a browser is our window into the World Wide Web, the Butler is our window into our personal cloud.

### 1.3.1 PrPI Semantic Index

The PrPI Semantic Index is a semantic web that keeps track of all the data in a personal cloud. This includes personal information such as our relations, devices, calendar, contact information, etc., as well as meta-data associated with large data types, such as photos and documents. The meta-data includes enough information to answer typical queries about the data, and location of the body of the larger data types. In addition, it also keeps track of the access rights granted to our friends at fine granularity. With the help of a URI (Uniform Resource Identifier), the protocol eliminates data copying by allowing the application to retrieve data directly from storage.

### 1.3.2 Personal-Cloud Butler

In this model, every user has a Personal-Cloud Butler, a program that manages a user's personal cloud of information on his behalf. The Personal-Cloud Butler is so-named to suggest that it is entrusted with the most confidential information, and it carries out services with discretion, often with very high-level direction.

The Butler provides a unified, location-agnostic view of one's data and relations in the personal cloud. It actively mediates between all social-networking applications and the

various data repositories, providing an application interface that hides the complexities of authentication and authorization, enforces desirable access controls over data scattered on many data repositories, maintains an up-to-date PrPI Semantic Index of all the data in the cloud, and provides a high-level semantic query interface.

The Butler needs to be highly secure because it indexes all of a user's private data. We therefore prefer it to be placed under the user's direct control and not allow any third party direct access to data held by it. For example, users could potentially run the butler service on a highly reliable home gateway. An alternative would be for the Butler to run in the computing cloud, using services such as Amazon EC2 or Google App Engine, in which case the user may wish to keep the data encrypted on disk and only decrypt it while operating upon it in memory.

## 1.4 Experimentation

We have developed a fully working prototype of the infrastructure we present in this paper. In the prototype, our federated storage system can incorporate data from IMAP email clients, public information from Facebook, Flickr, Yelp, and Google using their web service APIs, file systems on any PC or laptop, and rich media on the iPhone, etc.

To test out our hypothesis that applications can be built easily using the PrPI infrastructure, we have developed a suite of applications including:

1. a phonebook application for a mobile device that integrates Facebook status with contacts on mobile phone and in email.
2. a collective memory service that displays photos from a group of friends in the same timeline.
3. a social reviews service that lets users search for reviews and ratings written by their friends or people in their communities.
4. personalized and customized RSS news feeds for friends.
5. a convenient movie scheduling application for a group of friends based on their preferences, availability and publicly available showtime data;.

With the code provided by the PrPI infrastructure, we were able to write these applications in only a few hundred lines of code.

## 1.5 Paper Organization

The rest of the paper is organized as follows. Section 2 describes how we represent a personal cloud in the system. Section 3 describes how the system keeps track of one user's data in a distributed collection of data storage devices. Section 4 describes how data is shared in this system. Section 5 describes a native API that enables applications to take full advantage of the semantic web representation. Section 6 describes the implementation of our prototype. Section 7 describes the set of native applications we have developed. We describe related work in Section 8 and conclude in Section 9.

## 2. THE PERSONAL CLOUD REPRESENTATION

We envision that there is a Personal-Cloud Butler service to manage each individual's personal cloud of data. A user

may have multiple personas, each of which may be given a different external name, similar to the collection of email addresses that we own today. Multiple of these personas are likely to map to the same Butler, because access control policies are used to restrict access. However, if desired, a user may have different Butlers for different personas.

Each Butler maintains a PrPI Semantic Index to manage an individual's personal cloud. As discussed in Section 1, the PrPI semantic index is a cross between a semantic database and a semantic file system. It contains all the smaller data objects like personal information and the meta information of the larger objects. For example, it knows when a photo is taken, where it is stored, and tags associated with the photo.

### 2.1 Kinds of Resources

All data objects in the PrPI Semantic Index are known as resources. There are two kinds of resources: large data objects where the body of the objects, known as *blobs* or binary large objects, are stored externally, and smaller data objects that are kept locally in their entirety in the PrPI Semantic Index. We refer to the former kind of object as a *blob resource*, and the latter as a *blobless resource*. All resources in the system have a unique PrPI URI (Universal Resource Identifier).

There are three special kinds of blobless resources in the Semantic Index:

**Identity:** An *identity* represents a PrPI user. Its URI has the form `prpl://identity/#uname` where *uname* is a unique name. Encoded in the unique name is the information on how to locate the user's Butler. One possible example for naming identities is OpenID [23]. Since OpenIDs are not universally adopted at this point, our system currently uses email addresses. We currently use a global directory to map an email address to its Butler.

**Group:** A *group* is a set of identities and it is used to facilitate the specification of access control. The URI of a group resource has the form `prpl://group/uname#gname` where *gname* is the name of a group given by user *uname*.

**Device:** A *device* holds one of the user's blobs. The URI of a device resource has the form `prpl://device/uname#dname`, where *dname* is the name of a device given by user *uname*.

All other resources in our system have a generic PrPI URI of the form `prpl://resource/#` followed by a unique identifier [1].

All resources have the following meta-data properties:

- Name: a human-readable and/or user-given name of a resource.
- Type: The name of the type of the resource. Each type may have type-specific meta-data. For example, a Device resource has a *Location* property, which specifies how the device can be contacted. A Photo resource may have a resolution of the picture, camera model, geographic location where the photo was taken, etc.
- Owner: the identity owning the resource.
- Last modification time: the time when the information in the resource was last changed.

For blob resources, additional meta-data include:

- Size of the blob: its length in bytes
- Content hash: message digest of blob content computed with the SHA-256 hash function
- MIME type: associated MIME content-type (e.g. image/jpeg)
- Source: this property is present only for resources owned by the user and not shared with other Butlers. It points to the Device resource holding the blob.
- Last modification time of the blob: the time when the blob was last updated.

## 2.2 Representing Metadata in RDF

Resources and metadata are described in RDF [2] as a series of subject-predicate-object triples. For instance, the fact that Alice has a friend Bob with a phone number 123-456-7890 and an email address bob@b.com is described with the following triple:

```
prpl://identity/#alice@a.com foaf:knows
  prpl://identity/#bob@b.com
prpl://identity/#bob@b.com rdf:type foaf:Person
prpl://identity/#bob@b.com foaf:name "Bob"
prpl://identity/#bob@b.com foaf:mbox "bob@b.com"
prpl://identity/#bob@b.com foaf:phone "123-456-7890"
```

The types of resources and related properties (ontology) are described in the Web Ontology Language (OWL) [3]. We have two OWL specifications for the system: the default ontology that describes resources and their built-in properties, and the second a user-extendable ontology that describes basic data types such as Address, Calendar, Person, or Music.

## 2.3 Semantic Queries

Information in the PrPI Semantic web can be queried using the SPARQL (Simple Protocol and RDF Query Language) language [4]. An example SPARQL query is shown below: it looks up the semantic index for a person's name, phone number, email address, and profile photo.

```
SELECT ?name ?phoneNumber ?emailAddress ?thumbnail
WHERE {
  ?person rdf:type foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:phone ?phoneNumber.
  ?person foaf:mbox ?emailAddress.
  ?person foaf:img ?photo.
  ?photo foaf:thumbnail ?thumbnail.
}
```

## 3. A FEDERATED STORAGE SYSTEM

In the PrPI infrastructure, all applications running on behalf of a user request data through the user's Personal Cloud Butler. The service of a Butler is available over the web and applications can thus operate from any browser. The Butler maintains an up-to-date PrPI Semantic Index for all the data in an owner's federated storage system. The Butler keeps track of all the owner's storage devices. Each storage device in the PrPI infrastructure runs a Data Steward program. The Data Steward has two primary functions. It updates the Butler with the semantic index of data held locally. In

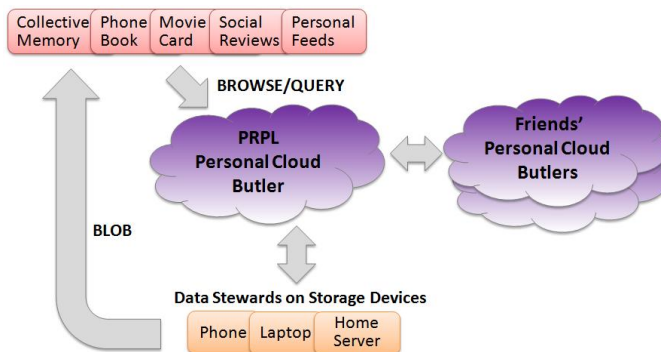


Figure 1: PrPI System Overview

addition, it is responsible for serving the blobs to applications directly, thus eliminating extra data communication through the Butler.

In this section, we present the basic design of the federated storage system, describing how a user can administer and access all his data in his personal cloud in a unified way through the Butler's web console. The Butler's sharing capability and its programming interface will be described later in Section 4 and Section 5.

## 3.1 Authentication

The PrPI infrastructure uses a certificate mechanism for authentication. A trusted certificate authority issues certificates for each user. The private key for each user is stored at the user's Butler. A user's devices and data managers need to connect to the user's Butler and register themselves by obtaining a certificate signed by the Butler. They store a private key corresponding to the certificate, which is used to authenticate themselves to their own Butler.

## 3.2 Data Stewards on Storage Devices

Every user's device in his federated storage runs a Data Steward operating on his behalf. It provides a uniform interface to the Butler and PrPI applications, hiding the specifics about how information is actually stored on the devices.

Specifically, the Steward performs the following functions:

- At configuration time, the Steward registers itself with the owner's Butler.
- It sends heartbeats to the Butler with updated device access information, such as the mobile IP address of a portable device.
- It generates the PrPI semantic index for the resources in storage and inserts them into the Butler's semantic index. As the resources change, it keeps the Butler semantic index synchronized.
- For all resources stored on its device, the Steward remember where the blobs are located. Specifically, it maps a PrPI URI to a source URI, such as one beginning with `file://` or `http://`.
- The Steward services blob access requests from applications. An application is required to obtain a ticket from the Butler owning the resource. The ticket certifies that the application was granted the right to perform specific operations. The ticket contains the URI of the requested PrPI user, a ticket expiration time, a list of authorized operations, and the resource URI.

### 3.3 Interoperability with Existing Data Systems

We envision in the future that there will be storage services that implement the PrPl Semantic Index protocol natively. To enable experimentation, however, it is useful to create PrPl interfaces for existing data sources so they can be made part of our federated storage system. Here are some examples of important data sources:

- Data in Personal Information Management (PIM) systems. Calendar and contact information are particularly relevant in social networking applications. Such information should be exported in their entirety as blobless resources.
- Email messages and especially attachments. Many people resort to mailing data to themselves so the data can be accessed from anywhere. However, it is hard sometimes to figure out where to find the correct version of a file. Meta-information on email messages, and especially the attachments, is an important data source to incorporate into the PrPl Semantic Index.
- Files in personal computers. The PrPl Semantic Index should capture all the meta information associated with files on one's personal computers.
- Online social networking sites. By now, many people have uploaded a lot of interesting information to various networking web sites. Some of these sites have an API so it is easy to build Data Stewards for them. For example, with the Facebook API, not only can we create a semantic index of all the data a user already has on Facebook, we can also write applications that continuously push information from the user's Butler onto Facebook. In this way, it is easy to recall information because it was never uploaded to the site in the first place.

### 3.4 Personal Cloud Console

The Butler provides an interactive web-based management console where a user can login to administer and access his personal cloud of information. The user can create and remove identities and group memberships, view registered devices, and run SPARQL queries directly. It also provides a generic resource browser (with a photo thumbnail), where a user can edit meta attributes, download blobs, and specify access control policies.

## 4. SHARING AND ACCESS CONTROL

Most social networking sites enable users to share data publicly or with their friends using a few selected privacy controls. In contrast, the PrPl infrastructure enables users to share data at fine granularity. User can grant access to other users at a per-resource level. Therefore, instead of specifying policies such as “only friends can access my profile page”, the PrPl system lets users specify fine-grained policies such as “my email address and work phone number are accessible by my co-workers, but my mobile phone number is accessible to all family members”.

A user's PrPl Semantic Index holds not just his personal data but also the shared portions of his friends' Semantic Indices. Typically the Semantic Indices are updated periodically. Suppose Alice wants to get access to data shared

by Bob, Alice's Butler contacts Bob's Butler requesting the PrPl semantic index of all the resources shared with Alice. It can also request Bob's Butler to send updates of the shared index continuously. Bob's Butler computes the semantic index containing the shared meta-data as well as any triples that contain any of the shared resources as the subject. Alice's Butler can also request only resources modified after a certain time, which is usually the time the shared semantic index was last requested.

### 4.1 Access Control

A user can create groups consisting of a set of individuals. The user can designate which individuals and groups have access to which resource. By default, the PrPl system has a special group called *Public*, of which all users of the system are automatically assumed to be a member. A user can share any resource with the group *Public*, thereby making it viewable by everyone.

In addition to groups, more flexible access control is possible by attaching semantic attributes to resources and individuals. This semantic attribute structure allows access control rules to be defined at many levels of granularity. For example, resources with the semantic attribute “vacation photos” can be easily shared with individuals with the semantic attribute “family”. Similarly, a presentation file could be shared with all co-workers in a corporate department. By expressing semantic attributes using standard ontologies such as FOAF [5] and NFO [16], sharing rules can be established between multiple levels of resource and principal hierarchies.

### 4.2 Security Model

Authentication is required before an application can make a request to the Butler on behalf of a user and for all communication between Butlers. A user authenticates to his Butler via a two-way authentication scheme such as Needham-Schroeder-Lowe or a combination of standard username/password and certificate mechanisms [19]. Butlers use shared public keys to ensure two-way authentication. As the representative of a user, the Butler stores the private key that allows it to act on that user's behalf.

Figure 2 illustrates how an application belonging to Alice accesses a blob resource  $R$  belonging to Bob. Before accessing the blob, Alice obtains an access ticket for  $R$  from Bob's Butler, along with the location of the relevant Data Steward (1-4). This ticket is similar to a Kerberos [25] client-to-server ticket and allows Alice's application to communicate and authenticate directly with Bob's Data Steward when requesting the resource  $R$  (5). The location of the Data Steward and the ticket can be cached as long as the ticket is valid, allowing the cost of obtaining the ticket to be amortized across multiple accesses to  $R$ . The primary security goals of this model are to provide the ticket-based access described above, while minimizing spoofing and replay attacks.

## 5. PRPL APPLICATION PROGRAMMING INTERFACE

At a high-level, a social networking application consists of two main parts: getting the data of interest, and then allowing the user to interact with it. In some cases, this two-step process may be repeated as users drill down into the results. The PrPl infrastructure described so far provides high-level

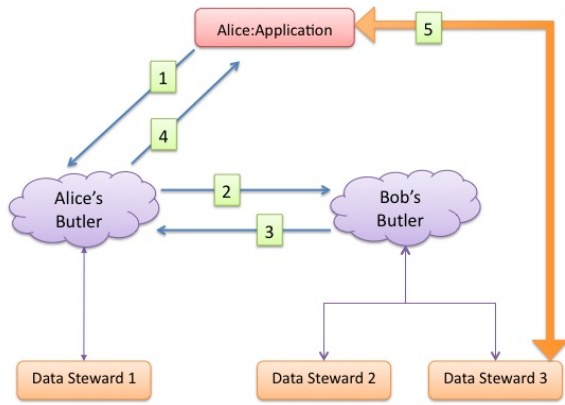


Figure 2: Accessing friend's blob

abstractions to make the first part easy. This section describes how the infrastructure also provides support for the second part after a brief description of a typical PrPI application.

### 5.1 Anatomy of a PrPI Application

PrPI applications run on behalf of a single user. The application can either (a) run directly on a device belonging to the user, in which case the device has authentication credentials that let it communicate directly with the user's Butler, or (b) run on a server which communicates with the user's Butler, with the server interacting with the user via a browser or some other interaction mechanism. In the latter case, the user must have some way of authenticating himself to the server.

Applications typically query the semantic index using the PrPI application API. The application API supports SPARQL [4], a standard query language for RDF graphs. The result of the query is a result set consisting of a set of resources and associated meta-data. The application logic processes the result set and may fetch resource data from its own or others' PrPI Data Stewards as needed. Applications may also request synchronization of the PrPI Semantic index with other users. Finally, applications can create new resources, set meta-data and create or modify resource sharing attributes via the application API.

### 5.2 User Interfaces

Many social networking applications operate on the same kind of data, and as such they can make use of generic user interface methods. Described below are a number of user interfaces available in the PrPI infrastructure. It is important that social applications be made accessible over mobile devices which have small screen real estate. We have selected UI modules that enable users to navigate dynamically through the data rather than provide a passive display.

**Map interface.** The Map view is useful for displaying location information. This view accepts a set of resources, with location information and a preview for each resource. It displays the preview on a map (built on top of Google Maps [6]); clicking on the preview opens up the resource.

**Timeline interface.** Timelines are an important metaphor for displaying data of many kinds. For example, it is useful to display photos according to the time the images are cap-

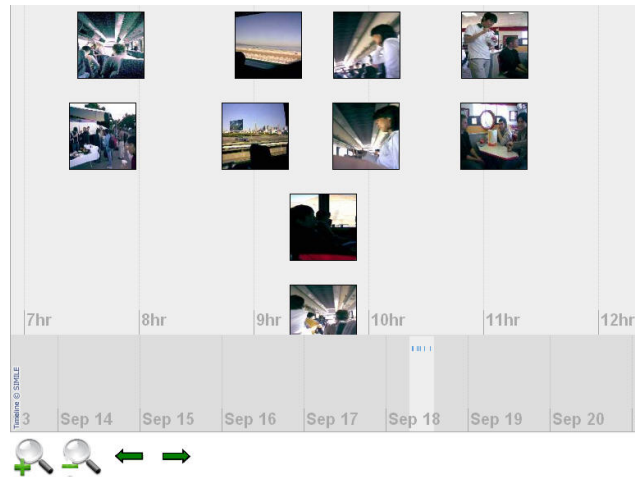


Figure 3: Timeline view of photos

tured, and to display files according to their last modified date.

Each resource optionally has a preview image associated with it, which is created when the resource is indexed and is stored by the Butler. For example, photographs may offer a thumbnail of themselves as a preview and a music file may have its album cover. Given the timeline view, users can scroll across time, spot clusters, click on a resource to access it, zoom in and zoom out quickly, etc. The timeline view is built upon the Simile timeline [7].

Mobile devices like the iPhone do not support some browser interactions such as dragging the timeline and zooming with a scroll wheel. To compensate for this, we have developed single tap buttons which emulate the functionality of dragging and zooming, resulting in the timeline being fully functional on the iPhone.

**Radial graph interface.** A radial graph view is an interactive visualization of connected resources in the RDF graph. It is useful for browsing through one's social network (by traversing edges with the *foaf:knows* URI) and to view friends' shared resources, e.g their photos (by traversing edges with the *prpl:hasPhoto* URI).

To use this view, the application passes in a resource URI which acts as a root node, and a list of property URIs. The view then displays nodes in the RDF graph that are connected to each other using any of the given property URIs. Resources are displayed with their corresponding preview graphic or with text for the resource name if there is no associated graphic. The view is initially centered around the specified root node, and as the user interactively explores the graph and clicks on other nodes, the view is re-centered around the currently selected node. Hovering on the thumbnail of a Photo opens up the full image; it could open up other kinds of resources as well.

The radial graph view is based on the Prefuse toolkit [17]. A demonstration of this interactive view is available at <http://prpl.stanford.edu/demo>.

**Card interface.** For displaying a series of possible options to a user, applications can use the card view described by Dontcheva et al [15]. The card metaphor is particularly suited to mobile phones given the form factor. The card is useful for applications which need to concisely summarize data acquired from multiple sources, and present options to the user.

The card view presents the user with a deck of cards, each containing elements consisting of HTML or images. Users can flip through the deck using the left and right arrows at the bottom of the screen. The user can also re-arrange the layout of the cards easily using drag and drop. Once the layout is changed, the same layout is used for all other cards in the deck, achieving consistency of user experience. The card view is built upon the JQuery toolkit [8].

## 6. IMPLEMENTATION

We have developed a complete working prototype of the system described in this paper. The RDF framework in the PrPI Semantic Index is based on the HP Jena library [20]. Currently, we are using a simple global directory that maps a user’s email address to the corresponding Butlers’ web server address. The Butler is written in Java, and is based on the lightweight Acme web server. It interfaces with applications using XML-RPC and it uses custom protocols wrapped in HTTP to communicate with other Butlers and Data Stewards.

The Data Stewards are also written in Java. To enable experiments with existing data sources, we have created PrPI interfaces for generic file systems on PCs, the file system and GPS location on the iPhone, IMAP contacts and attachments, Personal profile on Facebook, Yelp reviews, and Google contacts in Gdata. Most of these interfaces take only a few hundred lines of code, with the largest being the interface for IMAP attachments and contacts.

Data Source	Description	SLOC
PC file system	Music, photos, GPX or regular file	400
iPhone	Files, GPS, photos and contacts	250
IMAP folder	Contacts and attachments	1050
Facebook profile	Friend list, status, photos	200
Yelp	Reviews	650
Google Gdata	Contacts	200

Table 1: PrPI Data Stewards

## 7. APPLICATIONS

To validate the design of the PrPI platform, we have developed several applications that involve multiple data sources and utilize SPARQL queries to extract data from the Butler.

Table 2 briefly describes each application, its data usage and approximate complexity in terms of source lines of code. As the figure illustrates, the PrPI infrastructure facilitates creation of powerful applications using, typically, a few hundred lines of code.

Each application demonstrates one or more features of PrPI as described in previous sections. For example, Smart Phonebook illustrates the unification of data from disparate sources to create an easy-to-use mobile phonebook. Using Collective Memory, a group of people can arrange a collaborative viewing of photographs from a trip. It integrates data from multiple people, and visualizes pictures on a timeline. Personal feeds are an uber news feed, conveniently providing friends with a continuous window into our world, spanning diverse services and data sources. Each friend’s view is customized using PrPI’s access control system. Social Reviews shows off PrPI’s ability to filter along data types and combine private and public data. Here, users can share restaurant reviews with friends and achieve higher relevance during

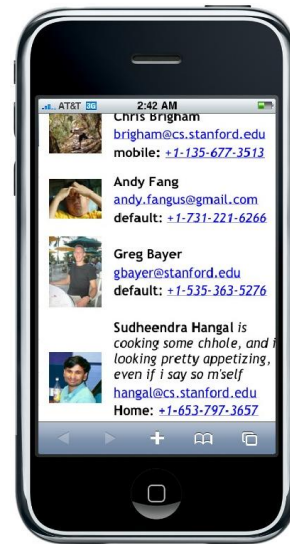


Figure 4: Screenshot of Smart Phonebook

search retrieval due to shared context. Finally, MovieCard combines multiple rounds of information gathering from private and public sources with the goal of finding convenient movie times for a group. It takes us forward to the day when our Butlers can co-ordinate mundane tasks on our behalf.

It is important to note that the applications are mere examples - MovieCard, for instance, can be generalized to think of general appointment scheduling, while the Smart Phonebook is a metaphor for data unification.

### 7.1 Smart Phonebook

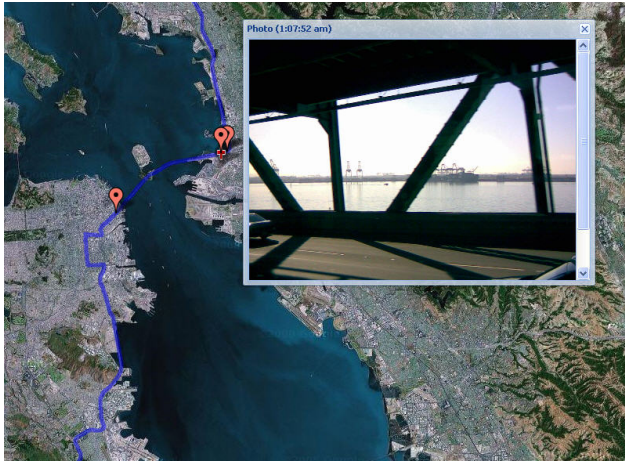
We have built an improved phonebook for mobile phones. As we all know, building up a phonebook on a mobile phone is a cumbersome experience. It involves importing existing contact lists from multiple websites, older phones, and e-mail accounts. Putting together these disparate sources of data would normally involve visiting multiple websites or writing mashups to access different services. Contact duplicate elimination and pruning requires additional effort. Figure 4 shows a screenshot of this application, which uses names and email addresses for contacts harvested from the user’s GMail account, phone numbers from an iPhone phonebook and status messages and profile picture from Facebook. Given that PrPI already unifies the aforementioned information about individuals, writing this application is a simple SPARQL query. The query results are presented using a scrollable list, much like a standard phonebook program.

### 7.2 Collective Memory

The Collective Memory application constructs a complete picture of an individual or group’s day, week, or month, using data from many different sources. Visualizing this data temporally and spatially augments a group’s memories of past events and lets the user make effective use of more information. Consider the usual problem of putting together and viewing the photographs taken by all the people on a trip. Currently, users upload photos into their favourite photo-sharing websites where they are stored under different user accounts, albums, etc. It is almost impossible to get a collective view of all these photos.

Application	Data used	Lines of code	UI format
Smart Phonebook	Phone contact list, Facebook status, profile photos, email address	100	List
Collective Memory	Photos, friends, Location	550	Map/Timeline
Personal Feeds	Various e.g. photos, movies, status update	480	Various
Social Reviews	Facebook friends, Reviews, Location	580	Custom/Map
MovieCard	Movie preferences, movie showtimes, Facebook friends, profile photos	620	Card

**Table 2: Some Applications built on the PrPl platform. Lines of code includes only non-blank, non-comment lines.**



**Figure 5: Collective Memory with Map Interface**

The PrPl Collective Memory application queries all of a user’s friends who were on the trip and obtains a list of all the photographs accessible to the user via a SPARQL query. It then displays the pictures on the timeline or map interface. The user can also tweak the query to generate a view of only the photos that were taken at the same location as himself during the trip. Figure 5 shows the screenshot of the application using the map interface. Our group used this application on our annual retreat to collate all the photos taken by various people during the trip; we also used the map layout to display a trail of our trip along with a display of all photos taken on the trip.

### 7.3 Social Reviews

The Social Reviews Service (SRS) lets a user ask the question: “What are interesting places around me that my friends recommend?” The application helps him identify what is popular among his circle of friends near his current location. This yields results that are of higher relevance due to shared, social context rather than querying generic public services such as Yelp. It illustrates the PrPl platform’s ability to easily filter data, and combine public and private review sources.

SRS utilizes combines the user’s current location with nearby points of interest (e.g. restaurants) that have been recommended by friends. The application gets its Butler to synchronize with friends’ Butlers to collect up-to-date reviews, which come from both public and private sources. It then issues a SPARQL query for restaurants near the current location that are rated highly by their friends. Figure 6 illustrates a search using this application. One of the locations appears multiple times in the results; it is popular among multiple friends. The application can present results to the user using either a map view with recommended lo-

cations plotted on the map, or with a list view ordered by a composite scoring function.



**Figure 6: PRPL Social Reviews:** a) User prpl is logged in and searches for nearby places recommended by friends. b) User publishes his current location in the form of zip code (or address, or city name). SRS searches reviews published by his and friends. c) and d) show a popular location recommended by multiple friends. Results are ranked by score (f) taking into account distance (e) from the user, number of visits and average rating.

This application highlights the benefits of using implicit contextual information to aid in searches. Once again, a simple query suffices to generate results that otherwise could be achieved only by looking up or calling friends one by one. An added benefit this application offers is that users may be willing to share reviews more openly within a limited circle of their friends, thereby promoting information flow among friends.

### 7.4 Personal Feeds

Personal Feeds enable a user to have friends subscribe to his activities across the entire web using a single feed URI. Currently, they need to track down the user on each social networking site and subscribe to his action news feed individually. A powerful, unified feed can be established virtually automatically using the PrPl platform. To subscribe, the feed reader synchronizes continuously with the user’s Butler and fetches updates when a resource is created or modified. The feed can fetch various resources using relevant SPARQL queries depending on the data type such as photographs, video, contact information, friends list changes, etc.

The Personal Feed application converts the information to a standard syndicated feed format such as RSS [9] or MediaRSS [10], which can be viewed by a standard reader on a desktop, web browser or mobile device. Figure 7 is a screenshot of a RSS feed displayed on the iPhone; selecting



Figure 7: Personal feeds of friends on an iPhone-based feed reader. On the right screen, user prpl views his friend’s new latest photos.

an item in the feed opens it up in its entirety.

While all friends can subscribe using the same feed URI, each gets a customized view of user A’s world. The PrPI personal feed is automatically customized per user to receive only the items that are visible to the receiver using PrPI’s access control mechanism. It has the advantage that it captures user activity across diverse sites and applications, without needing a central portal to host all data for all users. Personal feeds is an illustration of PrPI’s person-centric world view.

Syndicated feeds enable PrPI to present a well-understood format on which third-party tools can innovate upon. For example, geo-coded data from a trip can be represented using a GeoRSS extension. Loose coupling allows us to connect our feeds with external data visualizers e.g. web service [11] offerings of timeline and map views.

### 7.5 MovieCard

The Movie Card application takes us closer to the day when our Butlers will solve cumbersome activity scheduling and co-ordination tasks within a group. Consider the logistics involved in picking out a movie for a group of people. This involves putting together movie preferences of group members and their calendars, along with publicly available movie schedules and ratings. Ordinarily, this task would be performed with a sequence of phone or email exchanges between the whole group. In the PrPI system, the user has access to a semantic index of data for himself and friends (assuming his friends have shared calendar and movie preferences), as well as publicly available data such as movie showtimes. The MovieCard application performs a SPARQL query to generate a list of potential movies and showtimes that are convenient for all or some members of the group, and sorts them by movie rating and the number of people who can attend. We import publicly available movie showtime, location and ratings data from an online web service at regular (say, weekly) intervals.

MovieCard uses the card interface described in Section 5.2 for displaying the movie-going options to a user. Figure 8

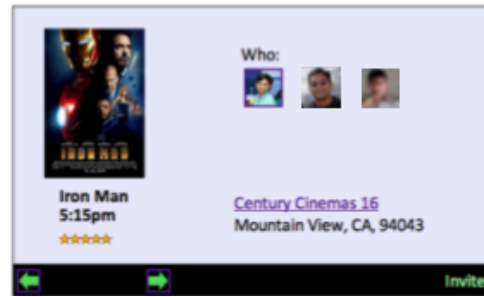


Figure 8: Screenshot of a MovieCard

shows an example of a card displayed to the user. The card lists the option for movie, showtime and theater, its rating, and pictures of the people who are available during that time. Since the card has type information associated with the objects it contains, it can offer type-specific actions. For example, selecting the movie poster brings up a full review of the movie and selecting the name of a theatre brings up a map centered around its location. Similarly, selecting a person’s pictures brings up a range of communication options with that user. The users can easily rearrange the layout of the card; the new layout automatically gets applied to all cards in the deck of results for consistency. Users flip through the deck using the forward and back buttons at the bottom of the card.

The MovieCard application illustrates the convenience of having structured and readily available data in a semantic index. This enables the application focus on the application logic, without needing to worry about the mechanics of obtaining the data.

## 8. RELATED WORK

There has been much interest recently in building a semantic desktop. The Haystack project developed a semantically indexed personal information manager [22]. IRIS [13] and Gnowsis are semantic desktops which currently target a single user [24]. Social semantic desktop [14] and its implementations [18][16] envision connecting semantic desktops for collaboration.

Our work differs from existing semantic desktop work in several important ways. First, PrPI permits social networking applications involving data from multiple users. Social semantic desktops only focus on collaboration among knowledge workers whereas we claim to be distributed social network infrastructure. Second, our work has a focus on ordinary consumers who may utilize multiple online services and applications, and own a multitude of computers and mobile devices.

Ensemble is a distributed file system for consumer devices [21]. While Ensemble is targeted at consumer appliances and managing media files, it does not have semantic index of resources and does not maintain semantic relationships between different data items, or support collaboration between users.

Desktop search applications such as Google desktop [12] create an index over personal data including documents, emails and contacts. While the desktop searches allow users to quickly search over several data types, they are limited not only to personal data, but also to single device mostly.

Mashups are web applications that combine data from

more than one service providers to produce new data tailored to users' interest. Although mashups can provide unified view of data from multiple data sources, they are limited in many ways compared to our work. First, data sources are limited to service providers: users have to upload their data to each individual service provider. Second, they do not let users share their data across different sources: users have different social network built around each service, and it is almost impossible to synchronize all of them.

## 9. CONCLUSIONS

This paper proposes an infrastructure that offers consumers a different way to manage their data and share information with their friends. Each user has an active Personal Cloud Butler that organizes his personal cloud data. Once the user informs the Butler about the various storage devices he owns and how he wishes to share information with his friends, the data is available to him and his friends, without having to explicitly upload the information anywhere. We demonstrated the utility of this approach by developing a good number of applications from sharing contact information, real-time photos, personal news feeds, restaurant reviews to even scheduling a movie together.

We found that many social applications can be described as a navigation of the result of a semantic query among friends' shared data. Like the well-known map-reduce construct from Google, which hides the complexity of running a task over a large number of parallel servers, the PrPI infrastructure makes it easy to create distributed social networking applications. The queries can be expressed in just a few hundred lines of code, and the infrastructure abstracts away the complexity of dealing with different people's data on different storage devices. This ease of development may give rise to applications we cannot yet imagine. Furthermore, without any central server that can access all the users' private data, this infrastructure can give rise to new applications that operate on data that individuals and groups are unwilling to divulge to some third party.

What we have described in this paper is potentially the beginning of a paradigm shift. This model currently assumes that a highly available, robust, secure Butler exists that runs on behalf of the user. This model can be realized, for example, by having an ISP provide a virtual Butler appliance as part of a home gateway service. We plan to incorporate caching techniques into our model so that our applications can continue to operate even when they are disconnected from the network or if the Butler becomes unavailable.

## 10. REFERENCES

[1] <http://www.ietf.org/rfc/rfc4122.txt>.  
 [2] <http://www.w3.org/RDF/>.  
 [3] <http://www.w3.org/2004/OWL/>.  
 [4] <http://www.w3.org/TR/rdf-sparql-query/>.  
 [5] <http://foaf-project.org>.  
 [6] <http://maps.google.com>.  
 [7] <http://code.google.com/p/simile-widgets/>.  
 [8] <http://jquery.com>.  
 [9] <http://en.wikipedia.org/wiki/Rss>.  
 [10] <http://search.yahoo.com/mrss>.  
 [11] <http://timeline.to/>.  
 [12] <http://desktop.google.com>.

[13] A. Cheyer, J. Park, and R. Giuli. Iris: Integrate. relate. infer. share. In *Proceedings of the Semantic Desktop Workshop at ISWC*, pages 738–753, 2005.  
 [14] S. Decker and M. Frank. The social semantic desktop. In *DERI Technical Report 2004-05-02*, 2004.  
 [15] M. Dontcheva, S. M. Drucker, D. Salesin, and M. F. Cohen. Relations, cards, and search templates: user-guided web data integration and layout. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 61–70, New York, NY, USA, 2007. ACM.  
 [16] T. Groza, S. Handschuh, K. Moeller, G. Grimnes, L. Sauer mann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjonsdottir. The nepomuk project - on the way to the social semantic desktop. In T. Pellegrini and S. Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 201–211. JUCS, 2007.  
 [17] J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM.  
 [18] JRichter, M. Vl, and H. Haller. Deepamehta - a semantic desktop. In S. Decker, J. Park, D. Quan, and L. Sauer mann, editors, *1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference, Galway, Ireland, 01.11.05*, 2005.  
 [19] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *In Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer-Verlag, 1996.  
 [20] B. McBride. Jena: A semantic web toolkit. In *Internet Computing, IEEE*, pages 55–59, Nov/Dec 2002. ISSN 1089-7801.  
 [21] D. Peek and J. Flinn. Ensemble: integrating distributed storage and consumer electronics. In *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 16–16, Berkeley, CA, USA, 2006. USENIX Association.  
 [22] D. Quan, D. Huynh, and D. R. Karger. Haystack: a platform for authoring end user semantic web applications. In *Proceedings of the ISWC*, pages 738–753, 2003.  
 [23] D. Recordon and D. Reed. Openid 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, New York, NY, USA, 2006. ACM.  
 [24] L. Sauer mann, G. A. Grimnes, M. Kiesel, C. Fluit, H. Maus, D. Heim, D. Nadeem, B. Horak, and A. Dengel. Semantic Desktop 2.0: The Gnowsisis Experience. In *Proc. of the ISWC Conference*, pages 887–900, 2006.  
 [25] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *In Proceedings of the Winter 1988 Usenix Conference*, 1988.