A PUBLISH/SUBSCRIBE MODEL FOR PERSONAL DATA ON THE INTERNET

Mark Wallis, Frans Henskens, Michael Hannaford
Distributed Computing Research Group
School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, NSW, Australia
mark.wallis@studentmail.newcastle.edu.au

Keywords: Publish/Subscribe, Distributed, Storage, Personal data, Web 2.0

Abstract: With the recent increase in web application reliance on user-generated content, issues such as data duplication, data age and data ownership are becoming an increasing problem. It is now common to have multiple distinct web applications storing duplicate copies of a user's personal information in distinct storage formats and locations. This paper proposes a change in paradigm that places the ownership of a user's personal data back into their own hands by moving the storage of that data away from web applications and onto private storage nodes exposed by 3rd party providers. Web applications can then subscribe to various pieces of data under electronic contracts that govern the data's usage.

1 INTRODUCTION

In recent years, the design of Internet applications has seen a move away from owner-generated content to user-generated content (O'Reilly, 2005). The Web 1.0 era was dominated by websites in which content was generated by the website owner. Next-generation applications, such as Twitter (Twitter Inc, 2009) (generally viewed as Web 2.0), in comparison, rely heavily on user-generated content (Vickery and Wunsch-Vincent, 2007). With the increase in application dependence on user-generated content numerous issues are becoming prevalent in relation to the maintenance of this data. Problems of data duplication, data age and data ownership occur because it is now commonplace to have multiple websites storing copies of a user's personal information in distinct storage formats and storage locations. This is especially common for personal information such as postal address and credit card number.

This paper proposes a change in paradigm that places the ownership and responsibility for the storage of a user's personal data back into their own hands. The change involves moving the storage of that information onto a private data storage service. The storage service is hosted either locally by the user (perhaps on a device connected to their home LAN) or externally by a data storage provider. A data storage provider makes available storage capacity and software to allow a user to publish and distribute access to their information. 3rd party applications then subscribe to various pieces of data under electronic contracts that define the rules governing the data's usage. Locating a specific user's data storage provider is addressed using extensions to existing Internet protocols. Parallels are drawn showing how such a design aligns with current advancements in Cloud Computing (Boss et al., 2007) and Service-Orientated Architectures (Bell, 2008).

2 PROBLEM DESCRIPTION

As explained below, three main issues were identified with classic Web 2.0 applications in relation to personal data storage: data freshness, data duplication and data ownership.

When the same piece of information is stored by multiple distinct web applications the chance of data becoming stale greatly increases. Take for example a user's postal address. When the user moves house they must currently access each web application to which they have provided their postal address, and manually update it. Before they can complete this

task they need to recall and locate all the relevant web applications. Such a task is increasingly non-trivial because of the growing number of websites that request personal information during registration. Currently there is no system that allows a user to push updates for such information to multiple web applications, so a user is required to remember, locate, access and update for each application in turn.

In addition to the issue of data freshness there is the issue of the amount of data storage wasted by the duplication of information. While the duplication of a small piece of information such as a user's address may appear trivial, when you considering the increasing inclusion of photos and video the gravity of the problem increases. The operators of web applications currently spend considerable money and effort on data storage technologies to store data that duplicates what is stored by other 3rd party systems.

Arguably the most concerning issue is the problem of data ownership. When uploading user information to a 3rd party web application a EULA (end user licence agreement) is generally in-place to describe who owns the information once it has been stored by the web application. It is common for the web application owners themselves to take ownership of the data, and it has been known for users to fight back against such restrictions (AFP, 2009). Passing the responsibility of storage and access of personal data to a 3rd party provider exposes the user to the possibility that 3rd parties could exert restrictions over the usage of that data.

These three key issues can all be tied back to the issue of a single piece of information, which logically belongs to the user, being stored by 3rd party web applications. This paper describes a change in paradigm which moves the storage responsibility to the end-user, and in the process, addresses the above three key issues.

3 RELATED WORK

Detailed below, multiple existing technologies attempt to address a subset of the issues described above by sourcing content from distributed data repositories.

Web application developers are able to use such HTML features as *iframe*s (Raggett et al., 1999) to build a single web page out of multiple components, with each component being potentially provided from a differing location. Such technologies do not scale to large deployments due to the manual way in which the web developer must link the various components. iframe technology is limited to building pages from static components that do not grow in number or relocate dynamically over time.

Content Delivery Networks (CDN) (Hofmann, 2005) are able to address performance issues surrounding centralised storage of information, but do not address the data duplication, ownership or freshness issues identified by this research. While nodes in a CDN may be viewed similarly to the concept described below, they do not provide the capability for direct end-user management outside of the specific web application in which they are tied.

Content Management Systems (CMS) (Mauthe and Thomas, 2004) are software packages designed to store and present information from a variety of sources under a unified interface. CMS are generally deployed in corporate environments where non-technical users wish to have the ability to update content presented on the web without having to learn markup languages, such as HTML. The CMS backend is capable of sourcing data from multiple locations, but the access method involves aggregating the data on the server before presenting it to the user. The links between various pieces of data are manually configured and the various data sources are non-transparent to the data owner.

Single-signon systems such as OpenID (Foundation, 2009) and Shibboleh (Cantor, 2005) attempt to address data duplication issues in the user authentication space by providing a single repository for user authentication information that can be used by multiple 3rd party applications to authenticate users within single, and across multiple organisations. OpenID currently only focuses on user authentication information. Shibboleh contains extensions to support a subset of user attributes such as address information, but does not scale to large dynamic environments in which there are no concrete relationships established between entities.

Distributed storage has long been important in the distributed computing space, where multiple distributed entities required access to a shared storage solution. Storage Area Networks (SAN) (Corporation, 2009) are capable of providing access for multiple entities to a centralised data storage area and hence are capable of addressing the data duplication and freshness issues. These deployments are currently statically tied to the applications that need to access the SAN and no Internet-based protocol currently exists to allow end users to dynamically locate distributed storage over the Internet.

None of the systems presented above are capable of addressing the three key issues identified in section 2 within a large Internet-scale deployment.

4 PROPOSED SYSTEM

The system proposed by this research plans to address the identified problems by introducing additional technology that allows storage of personal information to be offloaded from the web application and instead handled by 3rd party storage services. These storage services will be leased by individual data owners, and act as a 'single version of the truth' for that users personal data. Enhancements to the standard web browser design will allow this data to be accessed seamlessly. An API will be established that governs communication between the various actors in the system. These web browser enhancements will be a stepping stone between existing browser technologies and a complete Super-Browser implementation as presented in our parallel research (Henskens, 2007).

The first phase of unified personal information storage involves the data owner subscribing to a private storage service (PSS). This service is responsible for storing the data owner's personal information and is located either in-house or outsourced to a specialised data storage provider. The data owner then publishes content to the storage service. Once the data owner has stored shared information in their PSS the next stage is to allow a web application to subscribe to the information. Once a relationship is established the data owner establishes a link between their personal data and the web application. This link is akin to the data owner uploading content to the web application in a standard Web 2.0 scenario, except that in the PSS design the data owner purely provides the unique identifier of the information as opposed to uploading the content itself. Once the link between a piece of data referenced by the web application and the storage location for that data in a PSS is established, the web application is free to request that data directly from the PSS. The final stage in the design is the presentation stage. This will define how the information is transparently presented to end users.

The PSS design addresses the issue of data freshness by ensuring that all web applications present the latest information stored in the PSS. Only the latest information would be made accessible to 3rd party users and applications. Data duplication issues are addressed as the PSS becomes the only system required to store a users personal information. This will take the burden of data storage away from web application hosts and reduce their overall costs.

The PSS design also aids in addressing issues of data ownership by ensuring that the system storing the users personal data has a direct contractual relationship with the owner of the data. Users will be able to dictate the terms of usage for their data before agreeing to use a specific PSS. At the moment, a user is forced to accept the terms and conditions of a web application if they wish to use that specific application. In the PSS design, the user will be free to find another PSS to use if they do not agree with the terms and conditions of a specific provider. Web applications will no longer have any sway over a users personal data as they will not be responsible for collecting, storing and presenting that data. Their responsibility will end with providing a way of 'linking' 3rd party users to data stored in various PSS's using a well-defined API.

5 FUTURE WORK

Work is currently underway to implement a proof-of-concept PSS design and collect performance statistics comparing the proposed solution and a classic Web 2.0 approach. These performance statistics will show that such a solution can be designed with minimal impact to the average user, while still providing solutions to the three key issues presented. A detailed comparison will also be presented comparing the PSS design to the solutions described in section 3.

The initial PSS concept is targeted at information stored by web applications for the purpose of direct return on behalf of its users. For instance, the initial prototype suits applications such as image storage where the images are not altered by the web application. This style of information has been termed as 'inline'. Additional solutions are possible to support 'cacheable' data in addition to 'inline' data. Cacheable data is defined as data of which web applications store a local copy. The publish/subscribe model proposed will allow for the PSS to 'push' updates of such data to the linked web applications whenever updates are made. This will allow the issues of data freshness to be addressed for cacheable data, although it does not address the issue of data duplication and data ownership.

6 CONCLUSION

This paper presents three concerns identified with the growing popularity of Web 2.0 applications:

- Data freshness is addressed using a publish/subscribe model and single version of the truth. The data presented to the end user is always the freshest version because it is sourced directly from the user's PSS.

- Data duplication is addressed by removing the need for data to be stored by the web application. Appropriate web application registration and linking reduces the number of copies of a piece of data to a single instance stored on the PSS.
- Data ownership is addressed by ensuring that storage is the responsibility of the personal storage service directly engaged by the end user. PSS providers are liable to users, not to web applications, and hence the user has control over use of their data. Data ownership is clear-cut because the user is responsible for both the storage of, and access to, the data.

The presented solution ties directly into the realms of Cloud Computing (Boss et al., 2007), Service-Orientated architectures (Bell, 2008) and SAAS (Software-as-a-service) (Bennett et al., 2000). In a sense, a PSS can be seen as a SSP (Storage Service Provider) in a Storage-as-a-service (Foley, 2009) cloud component that allows other web applications to publish and subscribe to data within the cloud. The PSS system, however, will provide the necessary additional access and presentation layers on-top of the storage to ensure that the user experience is seamless. It makes sense in a Cloud Computing landscape that each application in the cloud is able to access a shared storage repository rather than having to replicate the same information for each web application deployed in the cloud.

The primary output of the next stage of this research will be a PSS API. A well defined API will allow multiple vendors to implement not only their own PSS's, but also the required web browser enhancements that are key to providing a end-to-end seamless solution.

REFERENCES

AFP (2009). About-facebook: backflip on data ownership changes. *The Sydney Morning Herald*.

Amazon Web Services (2009). *Amazon Elastic Compute Cloud Technical Guide*. Amazon.

Bell, M. (2008). *Introduction to Service-Oriented Modeling, Service-Oriented Modeling: Service Analysis, Design, and Architecture*. Wiley and Sons.

Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay, L., and Munro, M. (2000). Service-based software: the future for flexible software. In *Seventh Asia-Pacific Software Engineering Conference (APSEC'00)*, volume 17th, page 214.

Boss, G., Malladi, P., Quan, D., Legregni, L., and Hall, H. (2007). Cloud computing. Technical report, IBM Corporation.

Cantor, S. (2005). *Shibboleth Protocol Specifications*. internet2.

Cockburn, C. and Wilson, T. (1995). Business use of the world-wide web. *Information Research*.

Corporation, E. (2009). *Information Storage and Management: Storing, Managing, and Protecting Digital Information*. EMC.

Feldt, K. (2007). *Programming Firefox: Building Rich Internet Applications with XUL (Paperback)*. O'Reilly Media, Inc.

Foley, J. (2009). How to get started with storage-as-a-service. *InformationWeek Business Technology Network, http://www.informationweek.com/cloud-computing/blog/archives/2009/02/how_to_get_-star.html*.

Foundation, O., Openid authentication 2.0 - final. http://openid.net/specs/openid-authentication-2_-0.html.

Henskens, F. (2007). Web service transaction management. *International Conference on Software and Data Technologies (ICSOFT)*.

Hofmann, M. (2005). *Content Networking: Architecture, Protocols and Practice*. Morgan Kaufmann Publishers.

Mauthe, A. and Thomas, P. (2004). *Professional Content Management Systems: Handling Digital Media Assets*. Wiley.

OASIS (2007). Security assertion markup language (saml) v2.0 technical overview. Technical report, Working Group.

O'Reilly, T. (2005). What is web 2.0. *O'Reilly Net*.

Raggett, D., Hors, A. L., and Jacobs, I. (1999). *HTML 4.01 Specification*. W3C, w3c recommendation december 1999 edition.

Twitter Inc, Twitter. www.twitter.com.

Vickery, G. and Wunsch-Vincent, S. (2007). *Participative Web And User-Created Content: Web 2.0 Wikis and Social Networking*. Organization for Economic.