A Decentralized Architecture for Sharing and Reusing Life-logs

Johnson Iyilade and Julita Vassileva
Computer Science Department, University of Saskatchewan, 110 Science Place
S7N 5C9 Saskatoon, Canada
{Johnson.Iyilade, Julita.Vassileva}@usask.ca

Abstract. Nowadays, there are growing number of devices and applications that assist user in her day-to-day activities. These devices and applications collect and store enormous amount of information about user's daily lives - clickstreams, events, interests, etc. These life-logs are valuable data source for enriching user modeling when shared and reused across applications and devices. While many frameworks and solutions have been proposed for user model sharing and reuse, most existing solutions follow centralized architectural approach where the data is collected from the various sources and stored on a central server before it is shared with the user-adaptive application. However, centralized server can become a single point of failure and can be a ready target for hackers. In this paper, we describe a decentralized approach for multi-application life-logs sharing and reuse. We outline the essential components of a decentralized user life-logs sharing across applications and services on the web.

Keywords: User modeling, Life-logging, Information Storage, User Data Sharing, Reuse

1 Introduction

The ubiquity of the Internet and advances in mobile and communication technologies have led to a rapid increase in the number of digital devices (e.g. tablets, smartphones, wearable sensors etc) and applications (e.g. blogs, calendars, video, music etc) - designed to assist and automate human tasks and activities, enrich human social interaction and enhance our physical world interaction [1,2]. While interacting with these devices and applications, user leaves enormous amount of digital traces about their daily activities, events, locations, and interests. Although, the idea of keeping a record of our everyday life and activities (so-called *life-logging*) is not new (as it has been explored for the past four decades since the work of Steve Mann in the 1970s [3]), one essential difference today is that while user's life-logs were initially confined to what is gathered by a single application or device in a stand-alone machine or on a local network, user's activities are now being recorded in different

contexts by several independent applications and devices connected to the web. Ability to share, aggregate and reuse information collected by these independent applications are desirable goals for life-logs [4], particularly, to enrich user modeling, since they are valuable sources of information about user's activities, interests and preferences.
Obviously, the idea of sharing, integrating, and reusing user information gathered from many applications or sources is not a new problem in user modeling, many early efforts at addressing this problem, however, follows a centralized approach where the data gathered by the various sources are first integrated and stored in database on a server [5], many servers [6] or on a cloud [7]. Although, it could be argued that a central storage of integrated user model has the advantages of ensuring consistency and availability of data, however, as pointed out in [8], centralized architecture is very restrictive and applications have to adhere to user model representation and language of the central server. In addition, centralized servers often have well-defined points of access which can become the central point of failure. Privacy and security issues are also serious threats to a centralized storage.

In this paper, we describe a decentralized architecture for life-logs sharing, integration and reuse. In decentralized approach, rather than transferring and storing all of user's life-logs to a central repository for later sharing and reuse, each application maintains autonomy in the storage of data and only shares data that is relevant for a particular purpose of adaptation. Integration also occurs at the point of use. The described architecture support basic middleware services such as source selection, semantic mapping, and data integration.

The remainder of the paper is structured as follows. In Section 2, we provide a brief survey of related work. In Section 3, we describe the essential components of decentralized life-logs sharing architecture, and Section 4, concludes the paper.

2. Related Work

Since the 1990s, there have been various efforts towards providing tools and infrastructure that would facilitate the sharing, reuse and aggregation of user information gathered by different independent applications that interacts with the user. An important issue to be considered, however, is the network architecture that could support flexible and secure exchange of user data – i.e. whether the architecture of the user model should be centralized or decentralized [8]. In a centralized approach, the integrated user model is stored in a central server, many servers or on a server in the cloud [7]. The model is then shared across several user-adaptive applications. The first known centralized server for user modeling is the *generic user modeling server* [5] proposed in the 1990s. Subsequently, a number of frameworks for user modeling that follows the centralized architecture have also been proposed. In [9], a framework called IDIUMS was proposed for sharing data between user models in adaptive applications. IDIUMS provides a centralized storage for user model gathered across applications. User adaptive applications can then reuse the information in the central storage through a RESTful Web Service interface. Bielikova and Kuruc [10] also propose a reusable web service called User Model Web Service (UMoWS) for facilitating user model sharing and reuse. In their approach, adaptive applications communicate via SOAP/HTTP with UMoWS which acts as a store of user characteristics for adaptation. In [11], Personis Server, a centralized server was also proposed for storing and reusing user model. User data is stored on a server locally or in the cloud [7]. The central focus of the Personis Server is to allow for scrutability of adaptive systems. In [12], an architecture was proposed for collecting user information into a central Life-long User Model (LLUM) repository which can then be enhanced with information from external sources for recommendation and modeling. As noted earlier, centralized architecture approach guarantees consistency and availability of data, however, centralized approach often require applications to adhere to user model representation and language of the central server. In addition, centralized servers often have well-defined points of access which can become the central point of failure. Maintaining user data privacy and security is a big challenge, particularly with the growing global rise in data security breaches [13] which makes a centralized storage an easy target for hackers. Any successful hack into the server can grant the hackers access to all of the personal life data of the user, thereby exposing the user to unwanted harms.

An alternative approach to information exchange is a decentralized architecture. Here, fragments of user information/model are kept and maintained by each independent application. Only relevant information is exchanged and integrated as needed to fulfill a particular adaptation purpose. Decentralized approach requires integration [14] and semantic mapping [15] techniques provided by a broker for successful exchange.

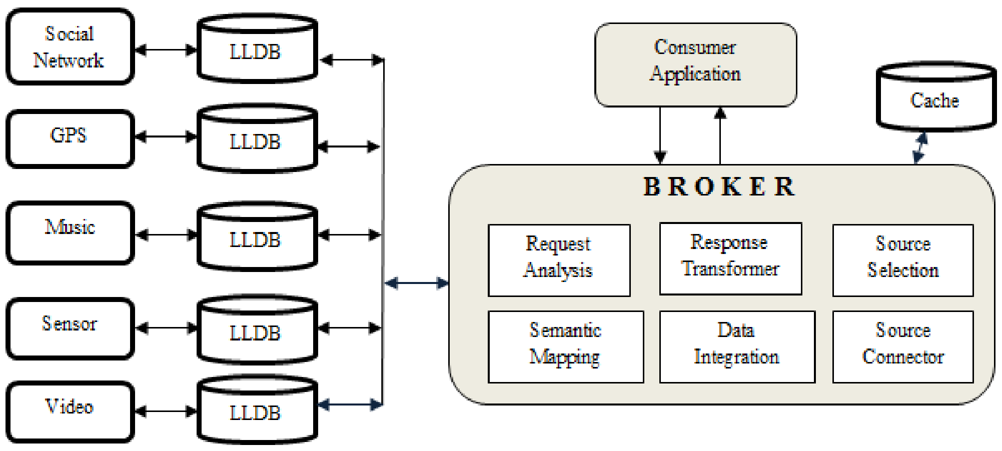

Figure 1: Architecture for Life-logs sharing across applications

3 Life-logs Sharing Across Applications

User Life-logs are currently fragmented in various independent databases, applications and devices. A decentralized architecture that supports the sharing and reuse of the life-logs across applications is presented in this section. The basic components to support decentralized architecture include independent data stores/sources, a broker/aggregator, and the consumer application which needs the data for adaptation purposes. As shown in Figure 1, user interacts with different applications and devices (e.g. social network application, GPS, Music app, Sensor, Video App, etc). These applications and devices collect different kinds of data which are archived in a Life-logs Database (LLDB) maintained by the individual application. Each application might employ various security mechanisms such as encryption to ensure that the data is kept safe. In addition, the application also regulate access to the data using pre-defined privacy policy which determines what part of the data is shared, with whom it is shared and for what purpose.

An important component of this architecture is the data broker. The broker provides middleware services and handles every request for data for personalization by the consumer application. The broker processes the request and communicates with the various data sources to retrieve and aggregate the relevant data. The final aggregated data is not stored by the broker but transmitted to the data consumer for further processing (e.g. reasoning and inference), however, partial results during the aggregation process can be temporarily stored in a cache. The core components of the broker and their functions are briefly described below:

a.) Request Analyzer

The request analyzer takes an incoming request from the consumer application or device in order to determine what the request was and what data is required to fulfill the request. Every request contains the *identity of the consumer application* (for authentication and authorization), the *purpose of adaptation*, the *identity of the user* whose data is being requested. The request can be sent to the broker as HTTP/GET request.

b.) Source Selection

Another important component of the broker is selecting the data sources that have user information relevant to the request based on a specified purpose of use. This is handled by the *source selector*. In an open environment, selection is also important in order to ensure trustworthiness and quality. Hence, trust mechanisms can be employed for determining which provider is to be selected using quality metrics such as *reliability*, *availability*, and *integrity*.

c.) Source connector

After selecting the sources for the data retrieval, different kinds of connection will be required to retrieve the data, particularly when the data is from different sources which requires different application interface for retrieval. For example, for a social network store, a *Social Connector* may be required while for a RESTful resource, an HTTP/GET protocol will be require. The task of the source connector is to determine the appropriate connection mechanism to retrieve data from each source. At this stage also, the source connector may have to perform *user identity mapping* to ensure proper user authentication in order to be able to retrieve the user information from the various sources.

d.) Semantic Mapping

Since the various data sources may use different representation for data it stores, an important task is semantic mapping. The goal of semantic mapping is to resolve the differences in representation from the specific representation of each data sources to a generic representation for the aggregated model and vice versa. In [14], the mapping was done using a hand-crafted mapping rule, while a mediation framework for mapping is presented in [15].

e.) Data Integration

The data integration component is responsible for merging the retrieved data from various sources into user model of proper granularity and resolving all conflicts (e.g differences in data values) associated with the merger.

f.) Response Transformation

The function of the response transformer is to ensure that the data is in the proper format required by the consumer application. For example, this may require the transformation into an RDF format, XML format, FOAF, JSON or any other format required by the consumer application.

4 Conclusion

Our goal in this paper is to describe a generic architecture for sharing and reusing life-logs gathered by many applications and devices. We particularly focus on application-to-application user data sharing environment. A very important issue is whether the architecture will be centralized or not. Although, many of the existing approaches follows the centralized architecture where user data is first integrated and stored on a central repository before sharing. This paper presents an alternative approach to user life-logs sharing and reuse based on decentralized architecture. Each application maintains fragments of user life-logs collected in a particular context while interacting with the user. User modeling, in this case, is then a series of process involving tasks such as request analysis, source selection, semantic mapping, data integration, and response transformation performed by the broker.

References

[1] Poslad, S (2009). "Ubiquitous Computing: Smart Devices, Environments and Interactions". John Wiley & Sons Publishing. ISBN: 978-0-470-03560-3

[2] Seng, L (2007). "Context-aware Pervasive Systems: Architectures for a New Breed of Applications". Auerbach Publications. ISBN: 0-8493-7255-0

[3] Kieron, O., Tuffield, M. M., Shadbolt, N. (2009). "Lifelogging: Privacy and empowerment with memories for life". Identity in the Information Society (Springer) 1 (1): 155. doi:10.1007/s12394-009-0008-4

[4] Kay J. and Kummerfield B. (2009). "Life Long User Modeling: Goals, Issues and Challenges". 1st Workshop on Lifelong User Modelling Workshop held in conjunction with UMAP.June 22-26, 2009, Trento, Italy. Online at: http://sydney.edu.au/engineering/it/~llum/2009_UMAP09_llum/Kay_Kummerfeld.pdf

[5]Kobsa, A.2001. Generic User Modeling Systems. User Modeling and User-Adapted Interaction. Vol. 11, pp. 49-63

[6] Fink, J. and Kobsa, A. 2000. A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. User Modeling and User-Adapted Interaction. vol 10. pp 209-249.

[7] Dolog, P., Kay, J. Kummerfeld B. (2009). Personal Lifelong User Model Clouds. Proceeding of the Lifelong User Modeling Workshop at UMAP09, June, 2009, Trento, Italy, Pp: 1-8

[8] Cena F. and Gena C. (2012). Designing and evaluating new generation user models (NewGUMs). Tutorial at UMAP 2012. Slides Accessed Online at: http://umap2012.polymtl.ca/en/workshops-and-tutorials/84-workshops-and-tutorials/104

[9] Prince R. and Davis H. (2011). IDIUMS: sharing user models through application attributes. Poster presentation, Proc. User Modeling, Adaptation and Personalization, UMAP 2011, Girona, Spain, July 2011. Springer Verlag. Berlin-Heidelberg, ISBN 978-3-642-22362-4. pp 40-42

[10] Bielikova M. and Kuruc J. 2005. Sharing User Models for Adaptive Hypermedia Applications. Proc. 5th Int. Conf. Intelligent Systems Design and Applications. Washington DC, USA. pp. 506 - 513.

[11] Kay J., Kummerfeld B., Lauder P. 2002. Personis: A server for user models. In P. de Bra, P. Brusilovsky, and R. Canejo, editors. Proc. Adaptive Hypermedia and Adaptive Web-based Systems, Springer Verlag. ISBN:3-540-43737-1. pp. 203-212.

[12] Elliott D., Hopfgartner, F., Leelanupab, T., Moshfeghi, Y. and Jose J.M. (2009). An Architecture for Life-long User Modelling. Proceedings of the Lifelong User Modelling Workshop, at UMAP'09 User Modeling Adaptation, and Personalization. pp 9-16

[13] Hartwig, R.P. and Wilkinson C. (2013). "Cyber risks: The Growing Threats". Accessed Online at: http://www.iii.org/assets/docs/pdf/paper_CyberRisk_2013.pdf. Last accessed date: 08-April-2013

[14] Fabian Abel, Nicola Henze, Eelco Herder, Daniel Krause. Linkage, Aggregation, Alignment and Enrichment of Public User Profiles with Mypes. Proc. 6th Int. Conf. Semantic Systems (I-SEMANTICS), Graz, Austria, September 2010, Article No. 11. ISBN: 978-1-4503-0014-8.

[15] Berkovsky S., Kuflik T., Ricci F. 2007. Mediation of User Models for Enhanced Personalization in Recommender Systems". User Modeling and User-Adapted Interaction. Vol. 18, Issue 3., pp 245-286