37 FRAMEWORK FOR OPEN, DISTRIBUTED AND SELF-MANAGED SOCIAL PLATFORMS

Juliana Mitchell-Wong, Suk Keong Goh, Mohan Baruwal Chhetri,
Ryszard Kowalczyk, Bao Quoc Vo
Centre for Information Technology Research
Swinburne University of Technology
Hawthorn, Victoria, AUSTRALIA
{jmitchellwong, sgoh, mchhetri, rkowalczyk, bvo}@ict.swin.edu.au

At present there are no digital social platforms that are open, distributed and self-managed. The openness enables end-users to customize their interactions through their selection of relationships and applications; and application developers to customize an interface for end-users with existing or new services. The distributed architecture ensures the scalability of content and entities, and the resilience to abuse. The self-managed platform provides the entity with control over its relationships; applications with control over the services it provides; and end-users with control over their interactions. These requirements led to the design of the social platform framework described in this paper. The key features of the framework are its modular design, use of open standards, distributed architecture, and policy-based management.

1. INTRODUCTION

A digital social platform is an environment that provides social entities with digital connectivity to other entities, and interactivity with other entities through digital applications. These entities represent individuals, communities, or organizations; and they each have a social network consisting of related entities. The mesh of all entity relationships forms the social ecosystem.

Digital social platforms are often associated with the development of web-based social ecosystems. However, the platform can be used to develop any social system where digital entities interact with others such as a supply chain management system. In this paper, the layers and modules of the social platform framework are abstract and applicable to all social ecosystems but the technologies and examples discussed are specific to web-based social ecosystems.

The paper is structured as follows. Section 2 presents background information on existing social platforms, followed in Section 3 with the motivation for the proposed platform framework. The framework along with the technologies for online communities is presented in Section 4; and the prototype implementation and validation of the framework is described in Section 5. A summary of the features for the proposed platform concludes the paper in Section 6.

2. BACKGROUND

In recent years, a number of social platforms have emerged such as Facebook[1], Bebo[2], Elgg[3] and Krawler[4]. They each have different characteristics including open or closed platform source; centralized or distributed connectivity architecture; ecosystem specific or federated identity; direct, indirect, or group relationships; central or self defined profile access control; and application programming interface (API) or ecosystem specific application development.

Most social platforms are closed source and used to create an open ecosystem of entities such as Facebook and Bebo. There are also platforms for closed ecosystems, such as an organization's intranet, that are also closed source such as Krawler. These closed source platforms have led to the widespread walled garden ecosystems because their platforms are incompatible with one another. Open source social platforms such as Elgg enable social ecosystems to be compatible and integrable.

The architecture of most existing platforms is central where content is stored and managed from a single location. The limitations of this architecture include scalability of content and entities, and resilience to abuse. The distributed architecture addresses these limitations and some new platforms are moving in this direction such as Krawler.

Traditionally, the identity of entities is ecosystem specific. However as platforms move towards a distributed architecture such as Elgg, standards for federated identities are emerging such as OpenID[5]. Federated identities enable entities to use the same identity across garden walled ecosystems. This is a step forward towards enabling an entity to integrate its presence in multiple existing ecosystems, and eventually the integration towards a single ecosystem.

Federated identity is also important when entities form relationships across ecosystems of type direct, indirect or group. Direct relationships are those the entity can discover on its own; indirect relationships are those that require assistance from an existing relation such as in the Elgg platform; and group relationships are those that cluster existing relations. Direct relationship is the main type of relationship an entity forms in an ecosystem; group relationships are common in most ecosystems for organizing relationships; however many shy from indirect relationships because of the difficulty in maintaining the privacy and security of entities.

Privacy and security issues also arise in the access of content such as an entity's profile. In a web-based social ecosystem, it is commonly self-controlled, while intra-network social ecosystems such as those based on the Krawler platform are often controlled centrally. The central manager of an intra-network social ecosystem typically has the trust and agreement of its entities from the real world to publish their profile, and it is held accountable for breaching this trust.

Thus far the platform characteristics that support the end-user are described. Application development is a characteristic that affects developers. Since the launch of the Facebook API, it has been popular with both end-users and application developers. It enabled end-users to customize their interaction environment, and

[1] http://www.facebook.com/
[2] http://www.bebo.com/
[3] http://elgg.org/
[4] http://www.krawlernetworks.com/
[5] http://openid.net/

third party developers to develop applications. However Facebook is proprietary, which led to the development of Google's OpenSocial[6]. It was launched as an open API that is compatible across ecosystems that are developed based on the OpenSocial platform. Unlike Facebook that requires applications to be stored on its server, OpenSocial enable application developers to host their applications on their own server or with Google. Not too long after OpenSocial was released, Bebo released its Open Application Platform. It is open and based on the Facebook API because Bebo wanted to utilize the numerous existing applications developed for Facebook. Before these APIs, applications were custom developed for the ecosystem which was often inadequate to meet all the needs of the end-users. End-users overcame these inadequacies by forming their social networks in multiple ecosystems for the different applications they offer.

3. MOTIVATION

Existing platforms demonstrate a wide range of characteristics. However, there is no platform that is open, distributed and self-managed. The openness benefits both the end-user and application developer. End-users are able to customize their interaction environment through their selection of relationships and applications, independent of their community or organization membership; whilst application developers are able to customize an interface for end-users using existing or new services. The distributed architecture of a platform ensures the scalability of content and entities, and its resilience to abuse. The self-management in the platform provides entities with control over their relationships; applications with control over the services they provide; and end-users with control over their interactions.

These requirements led to the proposed platform, VastPark's openSocial[7]. The platform is open source, has a distributed architecture, uses federated identities, supports direct, indirect and group relationships, provides the user with control over their own profile, and has an open API for application development. Open source enable different ecosystems to be developed on the same platform, thus facilitating interactions between these ecosystems. The distributed connectivity architecture enables entities to interact independent of garden walled ecosystems. The federated identity management provides a unified profile, authentication, and authorization of an entity across garden walled ecosystems. The proposed platform enables the entity to define any type of relationships: direct, indirect and group. The privacy and security issues of these relationships are managed through self-defined policies for incoming and outgoing interaction access. Finally, applications are developed using an open API so that developers can create applications for multiple ecosystems and to manage their own service. Table 1 provides a characteristic comparison of the existing and proposed platforms.

[6] http://code.google.com/apis/opensocial/
[7] http://www.opensocial.com. Note that VastPark's openSocial is unrelated to Google's OpenSocial.

Table 1 – Characteristic comparison of existing and proposed social platforms.

	Facebook	Bebo	Elgg	Krawler	openSocial
Platform source	Closed	Closed	Open	Closed	Open
Connectivity architecture	Centralized	Centralized	Centralized	Distributed	Distributed
Identity management	Platform	Platform	Platform, federated	Platform	Federated
Supported relationship	Direct, group	Direct, group	Direct, indirect	Direct, group	Direct, indirect, group
Profile control	Entity	Entity	Entity	Platform	Entity
Application development	Closed API	Open API	Platform	Platform	Open API

4. PLATFORM FRAMEWORK

The framework of the platform in Figure 1 is designed as layers and modules. Layers separate the different areas of the framework whilst modules separate the different features within the layers. The layers are connectivity, social relationship, social tool, end-user application, management policy, and end-user or software agent. An interface connects the different layers and modules, thus eases the development and maintenance of the platform such that existing technologies are used and future technologies can replace the old. To enable the replacement of a module without affecting the others on the platform, a specification for the interfaces of the modules are required.

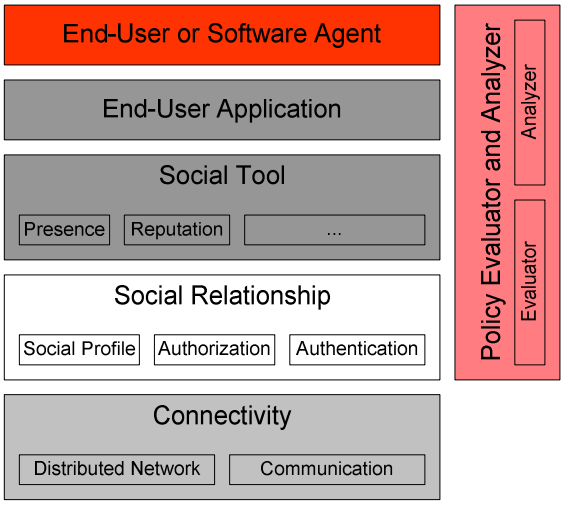

Figure 1 – Framework of a social platform

The connectivity layer form links to other entities on the web for interaction. It consists of two modules: distributed network and communication. The distributed or peer-to-peer (P2P) network is where entities discover and form links to other entities

on the web. Discovering and linking entities at this low-level removes the dependence on applications. Some existing P2P protocol include Kademlia (Maymounkov, 2002) and Tapestry (Zhao, 2004). Communication between entities at the network level is commonly Transmission Control Protocol (TCP)[8] or User Datagram Protocol (UDP)[9]. Other available protocols include the Datagram Congestion Control Protocol (DCCP)[10] and the Stream Control Transmission Protocol (SCTP)[11].

The social relationship layer establishes relationships with other entities. It consists of three modules: social profile, authentication, and authorization. The social profile defines its direct, indirect and group relationships. This profile is stored locally or by a third party. The authentication and authorization modules verify the identity and acknowledge the relationship of an entity. Relationships are mostly formed in a centralized architecture where authentication and authorization are performed by a central authority. However as relationships begin to form in a distributed architecture; open standards for authentication and authorization have emerged such as Security Access Markup Language (SAML) for the exchange of authentication and authorization between entities; and OpenID for the authentication of identities.

The social tool layer consists of tools that utilize relationships to obtain information. Presence and reputation are two examples of such tools. Presence is a tool that determines the state of a user represented by an entity on the web. The simplest presence tool has two user states: online and offline. In recent years, the eXtensible Messaging and Presence Protocol (XMPP)[12] also known as Jabber has gained popularity as the protocol for presence awareness. The reputation of an entity is the aggregated observations from other entities. It has been used effectively on the web in centralized communities where observations are reported to the central authority, such as e-Bay[13]. In distributed models, observations are obtained from other entities as required. Existing distributed reputation models include PeerTrust (Xiong, 2003) and Trust Model for Mobile Agent Systems Based on Reputation (TRUMMAR) (Derbas, 2004).

Applications provide end-users with their interaction environment interface. They are custom made for the ecosystem, or developed by a third-party. Custom made applications are often proprietary. However, if open technologies such as Internet Message Access Protocol (IMAP) are used, interactions can cross ecosystems. Applications can also be a mashup (aggregation) of services or widgets (small applications), making it easier for users to create their own application. Bebo's Open Application Platform and Google's OpenSocial are two APIs that enable third-parties to develop applications for use across different platforms.

The policies in the social platform controls access authorization to resources, and sets obligatory actions which can be a reward, penalty or neutral action when specified conditions are met. These policies are specified and evaluated against a request in a language such as Ponder (Damianou, 2001), Rei (Kagal, 2002) and

[8] http://tools.ietf.org/rfc/rfc793.txt
[9] http://tools.ietf.org/rfc/rfc768.txt
[10] http://tools.ietf.org/rfc/rfc4340.txt
[11] http://tools.ietf.org/rfc/rfc4960.txt
[12] http://www.xmpp.org/
[13] http://www.ebay.com/

eXtensible Access Control Markup Language (XACML)[14]. The completeness and consistency of policies are often analyzed using logic. The two most common types of logic for this are description logic (DL) and logic programming. They are both mature research areas, thus either representation of the policy leads to readily available analysis tools such as Selected Linear Without contrapositive clause Variants (SLWV) a theorem prover for logic programming (Pereira, 1993), and Pellet a DL reasoner (Sirin, 2007). There are three types of resources: entity, application, and user interaction. The entity's policy controls access to its profile that consists of relationship and personal information, and its policy that explicitly specifies its access control. This policy is used for organizing the entity's relationships. The application's policy controls access to its profile that consists of service and attribute information, and its policy that explicitly specifies its access control. This policy is used for managing the application's quality of service. The user's interaction policy controls the access of incoming and outgoing messages according to the sender's relationship and the application used. This policy is used for managing user-to-user interactions.

The end-user initiates an action or responds to the action of others according to its interaction goals. Its interaction behavior is based on its knowledge of the relevant policies of its own, other users, and applications. When a planned action conflicts with a policy, the user decides if it is willing to bear the consequences of performing the planned action or to change its plan. The framework enables software agents to be used for automating the user interactions according to a specified goal. Existing agent toolkits include Java Agent DEvelopment (JADE)[15].

5. IMPLEMENTATION AND VALIDATION

VastPark's openSocial prototype is implemented according to the platform framework with basic functionality for each layer as shown in Figure 2. Although the modules are basic, they illustrate the benefits and validate the framework.

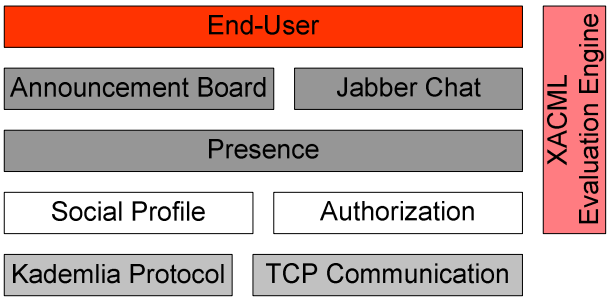

Figure 2: Prototype implementation

Two modules of the connectivity layer are implemented: distributed network and communication. The distributed network module is implemented with the Kademlia P2P protocol for discovering and linking entities in the social ecosystem and the

[14] http://www.oasis-open.org/committees/xacml/
[15] http://jade.tilab.com/

communication module is implemented with TCP to provide reliable in-order delivery of messages which is suitable for file transfer but not for streaming video. One problem the prototype faces with the distributed network is the prevention of incoming connections to systems behind different organization's firewalls. This is one of the problems that policy-based management of interactions aims to alleviate by providing the option to manage based on relationships rather than applications.

The social relationship layer consists of the entity's profile and authorization of other entities. The profile is managed locally by the entity and includes information on its relationships and personal details such as age and interests. The local management of the profile provides a single point for information about the entity. However, it requires the entity to be always present to respond to requests. The use of federated identity providers will eliminate this limitation. The authorization of other entities depends on the relationship policies of the entity.

The social tool layer consists of the presence tool which has been custom designed for the prototype. It compiles the status information of users by querying the entities directly or by receiving the information from a common relation.

The application is an announcement board that enables end-users to chat using Jabber. The application enables users to post announcements to the social ecosystem. This action utilizes the presence tool to determine the status of its relations, and their connectivity and communication channel. To respond to the announcements, users use the Jabber service. The announcement board is an application which integrates an external service and a custom built service.

The management policy is specified using the XACML language for controlling interactions to and from the entity and end-user. Authorizing interactions to and from the entity involves forming relationships whilst for the end-user it involves satisfying relationship and application conditions. The policies enable the entity and end-user to self-manage their incoming and outgoing interactions.

The end-user manually interacts with other users through the application interface with constraints set by the relevant policies it defines. This user to user interaction can be automated with software agents in the future.

The layers and modules of the framework such as the connectivity layer enable the replacement of the P2P or communication protocol module without affecting other modules as the interface between layers and modules have a specified format. Furthermore, it simplifies the development of each layer and module.

The prototype validates the framework and a summary is presented in Table 2. To find a plumbing service, an end-user customizes its interaction environment by establishing an interaction network consisting of entities that are plumbers and the application announcement board. This application is composed of a customized messaging service and the Jabber chat service. The entities are represented by peers in a P2P connectivity network; and they each manage their own profile and policy locally on their peer and not on a central server. Policies are used to manage an entity's relationship, the application's services and the end-user's interactions. Unlike other platforms where policies can only be written with conditions based on pre-defined attributes, the policies of this framework have the flexibility to be written with multiple conditions consisting of attributes defined by different sources. For example, the policy can combine the entity's trust measure of an entity with the number of announcements it receives a day as measured by the announcement board application to determine the result of the request.

6. CONCLUSION

Although the prototype implemented is rather basic, it still illustrates the characteristics of the platform framework: connectivity and communication between entities in a distributed network, relationship organization, aggregation of information from relations, aggregation of services or widgets to form an application, and self-management of relationships, applications and interactions.

The proposed framework combines the best characteristics for achieving an open, distributed and self-managed platform which no previous platforms have done. The key features are its modular design, use of open standards, distributed architecture, and policy-based management. The modular design eases the development and maintenance of the framework such that existing technology can be easily replaced with newer technology, and the failure of one component does not lead to the collapse of the system. Open standards ensure interoperability and widespread adoption and uptake of the platform because they are developed publicly by the collaboration of users and developers. The distributed architecture ensures scalability in content and entities, and resilience to abuse. The policies can be used to selectively share information rather than adhering to a fixed mechanism. This offers the entity, application, and end-user a more complete control over its information and interaction.

There are still research problems to tackle before the end-user's web-based social network reflects those in the real world, and are accepted by end-users. Among them are the monitoring and enforcement of policies, and integration of policies between entities such as those between an individual and a community. These are some current work of our group.

7. ACKNOWLEDGEMENTS

This work has been supported by the Agent-Enabled Social Networks project in collaboration with VastPark under the Australian Research Council's Linkage funding scheme (ARC Linkage Project LP0562500).

8. REFERENCES

1. Damianou N, Dulay N, Lupu E, Sloman M. The Ponder Policy Specification Language. In Proceedings of the Workshop on Policies for Distributed Systems and Networks 2001; 1995: 18-39.
2. Derbas G, Kayssi A, Artail H, Chehab A. TRUMMAR – A Trust Model for Mobile Agent Systems Based on Reputation. In Proceedings of the IEEE/ACS International Conference on Pervasive Services 2004; 113-120.
3. Kagal L. Rei: A Policy Language for the Me-Centric Project. HP Labs Technical Report 2002.
4. Maymounkov P, Mazieres D. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In Proceedings of the 1st International Workshop on Peer-to-Peer Systems 2002; 53-65.
5. Pereira LM, Caires L, Alferes J. SLWV – A Theorem Prover for Logic Programming. In Lecture Notes in Computer Science 1993; 660/1993:1-23.
6. Sirin E, Parsia B, Grau B C, Kalyanpur A, Katz Y. Journal of Web Semantics 2007; 5(2): 51-3.
7. Xiong L, Liu L. A Reputation-Based Trust Model for Peer-to-Peer E-Commerce Communities. In Proceedings of the IEEE International Conference on E-Commerce 2003; 275-284.
8. Zhao BY, Huang L, Stribling J, Rhea SC, Joseph AD, Kubiatowicz JD. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications 2004; 22(1): 41-53.