SONIC: Bridging the Gap between Different Online Social Network Platforms

Sebastian Göndör*, Felix Beierle†, Senan Sharhan§, Hussam Hebbo‖, Evren Kücükbayraktar‡ and Axel Küpper¶
Telekom Innovation Laboratories, TU Berlin
Service-centric Networking
Ernst Reuter Platz 7, 10587 Berlin, Germany, Tel.: +49 30 835358166
Email: *sebastian.goendoer@tu-berlin.de, †beierle@tu-berlin.de, §senan.mh.sharhan@campus.tu-berlin.de,
§hussam.hebbo@campus.tu-berlin.de, ‡evren.kuecuekbayraktar@campus.tu-berlin.de, ¶axel.kuepper@tu-berlin.de

***Abstract*—Online Social Network (OSN) platforms have become an important part of our everyday online lives. We communicate, share content, and organize meetings and events using social platforms and services. However, even though there is a strong trend towards OSN services to become the main communication medium, most OSN platforms are still proprietary, closed services that keep users from connecting directly and seamlessly to the services of other OSN platforms. The resulting lock-in effects are intentionally created by OSN operators, as their business models are built mostly on targeted advertisement services. While open and decentralized communication protocols exist for most other aspects of digital social interaction, there are only few micro-standards and protocols in existence for the social web. A holistic standard is yet missing. We envision a truly open and decentralized ecosystem of OSN platforms, where users are not cut off from friends using other social platforms and can freely migrate from one OSN platform to another at any time without losing established relationships in the social graph. This would allow users to freely choose an OSN platform of their liking instead of being limited in their choice to the platform used by one's friends. In this paper, we give an overview about the *SOcial Network InterConnect* protocol (SONIC), a novel social protocol for seamless social cross-platform communication. SONIC proposes a decentralized and heterogeneous *Online Social Network Federation* (OSNF), in which platforms are connected via a communication protocol. This allows different OSN platforms to seamlessly communicate with each other, while giving users the ability to migrate social profiles between platforms on demand without losing previously established connections to other user profiles.**

I. Introduction

The rise of the social web, which started with web platforms such as *Classmates.com*, *Sixdegrees*, and *Friendster*, began in the middle of the 1990's by allowing users to link to other individuals in order to model and maintain relationships of former friends or classmates. The idea of modeling social connections between individuals in a web service was extended by *MySpace* in 2003. The service featured detailed and highly personalized profile pages, on which users could not only provide information about themselves, but could also add further content such as images, music files, and videos - or even blogs and user groups.

As of today, Online Social Networks (OSN) have become one of the main communication platforms, allowing users to communicate via text, audio, and video, share content, or just stay in contact with friends and relatives. While a large number of competing OSN platforms with a broad variety of features exist as of today, *Facebook*, which was founded in 2004, managed to overcome its predecessors and competitors by far in terms of number of users and popularity [1]. This forced many competitors to focus on niche markets such as modeling relations to business partners (e.g. *LinkedIn* and *Xing*), or to address different aspects of social interactivity (e.g., communication via *WhatsApp*) or activities (e.g., publishing images via *Instagram*).

OSN platforms are nowadays mostly organized in a centralized manner, which forces their users to not only entrust all personal information to a single platform operator, but also to surrender copyrights. The personal data is then used e.g. for targeted advertisement or other forms of monetization with little or no control for users regarding how and what private information is used. The resulting consequences in terms of privacy and being locked-in by the OSN platform bother most users as control over one's data privacy is lost [2][3][4]. This led to the development of decentralized OSN architectures and systems such as the open source applications *Diaspora*[1] or *Friendi.ca*[2] that allow users to host their own data at an arbitrary server. *Diaspora* is based on a set of open standards and protocols such as *WebFinger*, but still fails to facilitate a seamless communication with arbitrary other OSN platforms [5]. To overcome this problem, many OSN platforms support specialized plugins that connect to OSN accounts on other OSN platforms via their respective - mostly proprietary - APIs and consolidate content such as status updates or friend lists in one web service or application [6]. Anyhow, this requires users to maintain an account at each connected OSN platform. Data, such as status updates, is then automatically synchronized between all connected platforms, which eases the process of managing multiple social accounts, but does not solve the problem of privacy concerns. Furthermore, seamless interaction such as group communication or commenting on content with two or more users from different OSN platforms is not possible [7].

The current trend of OSN platforms becoming the main social communication medium causes severe changes in business models of even big companies such as telecommunication providers [8], since revenue from traditional services such as voice calls and short message service (SMS) are losing significance. Yet unlike for established forms of digital human-to-human communication such as email or instant messaging,

[1]Diaspora: http://joindiaspora.com
[2]Friendi.ca: http://friendica.com/

there is no common and holistic standard for interaction of online social networks, even though features of such services tend to be very similar. As of this, players such as *Facebook* are able to keep their users, following the well known principle of *the rich get richer*: As migrating to other OSN platforms would cut of the digital connections to our friends and relatives, everyone stays with the biggest player. Potentially new users don't have a real choice when choosing a platform, as they are pressured to join the platform their friends are already using. A common and open standard that interconnects all participating OSN platforms would weaken the market position of all too big players such as *Facebook* and therefore allow other platforms to attract more users.

In this work, we present the *SOcial Network InterConnect* protocol (SONIC), which proposes an open and decentralized OSN ecosystem, in which users can freely communicate with each other - even when communication partners are hosted on different types of OSN platforms. In order to bridge the gaps between existing OSN platform implementations, the fact that a user's profile is hosted on another server should be kept transparent when communicating. The architectural requirements for the SONIC federation have been defined in [7][9], comprising a decentralized architecture, the use of open protocols and formats, the option for users to migrate their social accounts, seamless communication, the use of a single social profile, and global user identification. The remainder of this paper is organized as follows: The next section provides an overview about existing approaches, protocols, and standards in the area of distributed OSN. Chapter III gives a detailed description of the concept of the SONIC federation and defines the feature set of the SONIC federation. Chapter IV describes the architecture of the SONIC federation, while Chapter V gives an overview about the actual protocol. Chapter VII concludes the paper.

II. Related Work

Faced with the task of addressing privacy issues of centrally organized OSN platforms, Applequist et al. proposed a concept where users maintain a central repository of their data, which can be connected to multiple OSN platforms [10]. Following this approach, users can determine to which connected OSN platform what content should be synchronized, so that multiple types of OSN platforms can be maintained simultaneously via one central service. This allows a logical separation between a professional OSN profile such as *Linked.in* and a more personal one such as *Facebook*. This eases the user experience for users who are active in more than one OSN, yet OSN platform operators still get access to most of your personal information. Furthermore, users are forced to maintain user accounts on each connected platform, to which new content is then pushed. Distributed alternatives such as *Friendi.ca* or *Diaspora* proposed alternative architectures, where social profiles of users are stored on arbitrary servers. Social communication, e.g. messages or status updates, is carried out using special protocols to bridge the gaps between servers called *Pods* [6]. Other approaches tried to eradicate the use of centralized client-server architectures by implementing peer to peer technology. Examples are *PeerSoN*, which uses a *Distributed Hash Table* (DHT) to replicate profiles and related data to other devices in order to maintain availability even when a participant's device is turned off [11][12], or *TRIBLER*, which is based on the popular peer-to-peer software *Bit Torrent* to benefit from its superior content discovery and recommendation mechanisms [13]. *Safebook*, which was proposed by Strufe et al. focuses on privacy and security aspects in peer-to-peer-based social architectures. Here, information about a user's location in the network is replicated to other trusted peers forming logical *shells* around each user referred to as the *Matryoshka*. Requests to a user's data have to pass through these shells, so each user's privacy is enforced using multi-hop communication links [14]. Other research proposed to design OSNs as mobile hosted applications [15], e.g. *POLARIS* [16] or *VEGAS* [17]. Here, a user's personal mobile device is used to store social profiles as well as access social profiles of other users. Even though these approaches mitigate or eradicate the influence and knowledge of a central OSN platform operator about personal information, they all propose again semi-closed solutions and ecosystems, in which users are still hindered from connecting to other OSN platforms.

A. OSN Protocols

Today, most OSN platforms provide an API to allow developers of third party applications and services to connect to the OSN, thus creating well integrated solutions. For example, *Facebook* allows external services to connect to the platform via its *Facebook Graph API*[3], which comprises *nodes* representing items or persons, *edges* connecting nodes, and *fields* that describe attributes of nodes. Most other OSN platforms offer similar functionality but differ in the details. For example, *Google+* defines four resource types to be used in its API, being *people*, *activities*, *comments*, and *moments*[4], while *Twitter* provides an extensive REST-based API to their services. To access the API's functionality, each external application or service needs to register with the platform first and is assigned a unique API key to authenticate itself with each session or request. As most OSN platforms provide their own proprietary API which developers have to use, lock-in effects are created due to the extensive efforts required to adapt an application or service to a certain API. While most OSN platforms provide an API for external services to integrate themselves into the OSN platform, other approaches focus on open and distributed solutions. In 2010, *Facebook* introduced the *Open Graph Protocol* with the intent to include web pages into the social graph. The standard was inspired by *Dublin Core* and makes use of *RDFa* to enrich a web page with semantic information that describes the web page's contents [18][19].

In order to ease the process of adapting applications and services to multiple OSN platforms for developers, *Google* proposed to agree on one standardized social API. The standard would enable application developers to implement against one standard set of APIs, allowing the resulting applications to work with any OSN platform. *OpenSocial*, which was initially released in 2007, comprises a set of interfaces for applications and services to access data of any kind of OSN platform [20]. By implementing *OpenSocial* interfaces, OSN platforms provide a uniform standard API for social applications, so that the same application can be used within multiple OSN

[3]Facebook Graph API Reference: https://developers.facebook.com/docs/graph-api

[4]Google+ API Reference: https://developers.google.com/+/api/latest/

platforms without the need to adapt a new set of APIs to support each new OSN platform. For this purpose, *OpenSocial* defines distinct protocols for *data*, *people*, and *activities*, which allow access to the respective functionality. Even though *OpenSocial* defines a common social API, seamless communication between different OSN platforms is not supported. As *OpenSocial* focuses on APIs for social applications, the standard supports *Gadgets* to render the content. *Shindig*[5] a reference implementation is provided by the Apache Foundation. As of 2015, work on the *OpenSocial* standard has been taken over by the *W3C Social Web Working Group*.

Other approaches focus on more basic details, such as providing a common way to exchange status updates without being bound to a specific platform. One good example is *ActivityStreams 2.0*, which is an open and extensible standard for describing activities comprising *types*, *actors*, and *objects*. This way, *ActivityStreams* are "semantic descriptions of a completed or ongoing action" [21]. The standard is based on JSON and is easily extensible. *OStatus*, which was originally designed for microblogging services is an open standard for exchanging status messages. The standard defines an XML-based data format and implements PuSH as distribution mechanism [22]. Even though its functionality is limited, *OStatus* has been integrated into several existing OSN platforms such as *Friendi.ca*, *Lorea*, *Duuit!*, and *SocialRiver*.

Several other social protocols aim at connecting social profiles on different servers, but only cover certain aspects. One example is *Tent.io*, which is a protocol for evented data storage that allows users to subscribe to status updates from other users. The architecture proposes a similar subscription mechanism as PuSH and is based on HTTP and JSON [23]. For the domain of text-based real-time communication, the *Extensible Messaging and Presence Protocol* (*XMPP*) facilitates instant message exchange in an open and decentralized manner [24]. While *XMPP* originally focused on instant messaging, hundreds of extensions have been implemented for other uses, including OSN support[6]. *Friend-of-a-Friend* (*FOAF*) is a standard to model relationships between individuals using RDF [25]. The format is used within the protocol *FOAF+SSL* [26] for secure authentication of communication partners. Individuals are identified using an X.509 certificate and a *WebID*. *WebID* is used to publish information related to an individual using a dedicated URL in a profile document [27]. Each *WebID* profile is an RDF document that describes a person or entity associated with this document. For the domain of describing individuals, *Portable Contacts* provides a simple and elegant way to describe address books and persons. The specification supports JSON and XML and also defines authorization and access patterns [28].

A more comprehensive protocol called *Distributed Friends & Relations Network* (*DFRN*) was introduced in 2010 by Macgirvin, which was designed "to provide an open and distributed social communication platform with server requirements comparable to that of a typical hosted blog" [29]. The protocol is based on the *Atom Syndication Protocol* and provides polling and pushing of content between *DFRN* compliant servers. The distributed OSN platform *Friendi.ca* was originally built as a reference implementation for *DFRN*. As the protocol was considered as too "cumbersome" by the *Friendi.ca* developers, development of an alternative protocol called *ZOT* was initiated. *ZOT* was designed to connect different OSN platforms by using publicly available APIs and protocols. In 2012, *ZOT* was discontinued because of tendencies of centralized OSN platforms to apply "strict limits to outside services accessing their proprietary platforms" [7].

Even though social features such as commenting, expressing appreciation, messaging, or sharing content has become an integral part of the social web we know, integration of such features as of today is only possible by implementing against proprietary APIs. Even more severe impairment is the requirement for users to maintain an account on each OSN platform that should be used. For example, *liking* content requires to be logged in with *Facebook*. This requires users to maintain social user accounts on multiple OSN platforms.

B. OSN Featuresets

Even though today's OSN platforms differ in their provided functionality, architectural approaches, privacy and security concepts, or API model, most functions are equivalent or at least similar. This allows to derive a generalized featureset of today's OSN platforms to provide a better definition of social features. Heidemann et al. defined core functionalities of OSN platforms, which are user profiles, subscription to interest groups, and personal contacts [30], while classifying OSN platform based on three criteria: primary usage (private vs. business), focus (general vs. special interest), and access (open vs. closed) [31]. Rohani and Hock proposed a general featureset of social web sites describing a *personal profile*, *communication with peers*, a *personal bill board*, a *thinking room*, an *e-newsletter*, a *friendship network*, and *forums* [32]. Richter and Koch proposed a rather high level approach by defining a list of six basic functions of social networking sites, which can be narrowed down in the two basic categories *keeping in touch* and *identity management*. The basic functionality comprises *identity management* (presenting oneself), *expert search* (searching the network for specified criteria, or receiving recommendations), *context awareness* (discovering common interests), *contact management* (tagging people and defining access restrictions), *network awareness* (receiving status updates about activities of other's), and *exchange* (being direct communication, e.g. instant messaging) [33]. Boyd and Ellison derived a general featureset of social networking sites in 2007, comprising *semi-public profiles*, *connections to other users*, and *traversing this list of connections* [34]. In 2012, Paul et al. defined both a basic and an extended feature set for OSN services from an architectural point of view. Here, basic functionality comprises *profile management*, *relationship handling*, and *interaction*, i.e., messaging and sharing content. Extended features include APIs or search and recommendation functionality. *OpenSocial's* approach to build a common API for arbitrary OSN platforms required a common set of resources application developers could access as well as data formats that would fit to all kinds of OSN platforms. Here, *OpenSocial* defines data formats and RPC calls for *People*, *Groups*, *Activity Streams*, *AppData*, *Albums*, *MediaItems*, and *Messages* [20].

[5]Apache Shindig: http://shindig.apache.org/

[6]OneSocialWeb: http://onesocialweb.org

[7]The Friendi.ca Blog: http://friendica.com/node/24

III. The SONIC OSN Federation

Today's OSN platforms are mostly closed solutions, which disallow users to connect to and communicate freely with other platforms and services. Solutions proposed to address and mitigate these issues mainly proposed either alternative centralized OSN platforms themselves or rely on federated or completely decentralized architectures. All proposed alternatives have in common that users need to create a new user account within the new system, while seamless interaction with other OSN platforms is mostly still not possible. The motivation for users to abandon one closed system for another closed solution is therefore limited. In contrast to the approaches mentioned above, SONIC proposes a different paradigm. Here, a common protocol is used to allow different kinds of OSN platforms to interact directly. Following this approach, OSN platforms support a common API and protocol, which allows to exchange social information across platform borders, while addressing remotely hosted user accounts directly. This way, users can choose the OSN platform they prefer while staying seamlessly connected to all friends using other OSNs. Hence, it is rendered irrelevant whether a user's friends are using the same or a different OSN platform. The result is an *Online Social Network Federation* (OSNF) defined as a *heterogeneous network of loosely coupled OSN platforms using a common set of protocols and data formats in order to allow seamless communication between different platforms* [7].

Figure 1 depicts the concept of SONIC. A user (*Alice*) accesses the user interface of her OSN *Platform A* using a web browser, mobile application, or other interface. For *Alice*, it is not important whether her friends use the same OSN *Platform* or not. Her OSN platform server retrieves all relevant information for her in the background directly from her friends' OSN *Platforms* B and C using the SONIC protocol. The retrieved data (e.g., status updates or profile pages) are transmitted in a standardized data format and are integrated seamlessly into the view of the user interface *Alice* is using. This builds a foundation for an open and decentralized OSN ecosystem with no lock-in effects, in which any kind of OSN platform can participate while control over one's personal information is not necessarily given to a central authority.

Prerequisites for an OSNF ecosystem have been defined in [7] comprising a *decentralized architecture*, the use of *open protocols and formats* to facilitate *seamless communication* between platforms, allowing *migration* of user accounts to other OSN platforms [9] while not forcing a user to sign up with more than one OSN platform at once, i.e. having a *single profile*, which is identifiable across platform borders using *global user identification*. By implementing SONIC, barriers between existing OSN platforms will be mitigated by allowing transparent inter-platform communication. Users will still be able to use the OSN platform they are used to in the same way as before.

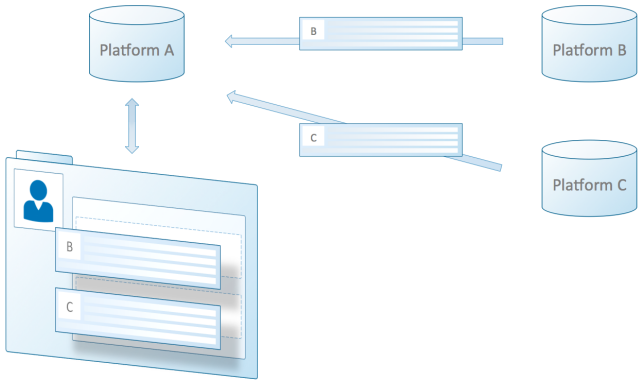

Fig. 1. Federation in SONIC: Information from other user's (remote) *Social Profiles* is directly embedded into the user interface. This closes the gap between different OSN platforms.

A. Core OSN Featureset

The task of defining a common standard for multiple OSN platforms requires to interconnect different features and formats of existing platforms. Even though today's OSN platforms differ in their provided functionality, architectural approaches, privacy and security concepts, or API models, most functionality is semantically equivalent or at least similar. This allows to derive a generalized featureset of today's OSN platforms to provide a better definition of existing OSN functionality. Based on the identified featuresets of other researchers as described in Section II-B, APIs and the general functionality of today's most popular OSN platforms have been analyzed. The analyzed platforms comprise *Facebook*, *Google+*, the distributed and open source approaches *Friendi.ca* and *Diaspora*, the Russian network *vKontakte*, the Chinese platform *RenRen*, the business platforms *Xing* and *LinkedIn*, and the micro-blogging platform *Twitter*. Looking at the provided APIs and user interfaces of these OSN platforms, the following list of general features has been identified[8]:

- **Profile page** A personalized profile page with information about a user.
- **Adding/following friends** Ability to add other users to a list of friends or acquaintances.
- **Instant Messaging** Exchanging text messages with other users.
- **Multiuser Instant Messaging** Engaging in group communication with more than two other users.
- **Posting status updates** Publishing posts about activities in a stream for others to watch.
- **Sharing content** Notifying others about content (e.g. a web page) by posting it in a stream.
- **Liking content** Expressing that you appreciate or value a content object.
- **Commenting on content** Posting a textual comment on other content for others to read.
- **Tagging** Tagging people in content who are associated with the content.
- **Image galleries** Publishing images and managing them in image galleries.
- **Event organization** Inviting other users to events.
- **Groups** Defining specific groups of users.
- **Poking** Sending a *poke* with no fixed semantic meaning to another user.

[8]The analyzed features are limited to features involving exchange of data between two or more users. Features that do not effect other users' profiles are not considered, e.g. categorizing users in local user groups.

Further features can be found in only few, mostly specialized OSNs, comprising for example rating content or voting. Table I gives an overview about what features are supported by which network. By analyzing the features of popular OSN platforms, a list of features can be derived, forming a *core featureset*, i.e. they are supported by almost all OSNs. These features comprise a *profile page*, *adding or following friends*, *(multiuser) instant messaging*, *posting and sharing content* e.g. status updates, *liking content*, *commenting on content*, and *tagging users* in content. The SONIC protocol is designed to support this *core featureset*, while optional protocol extensions will facilitate supporting additional features between OSN platforms.

IV. SONIC Architecture

SONIC promotes an open and decentralized OSNF, in which different OSN *Platforms* are loosely coupled. Users maintain *Social Accounts*, comprising the associated *Social Profile* and a *Social Record*. While *Social Profiles*, which are a collection of all data associated with a user account are hosted on a *Platform*, *Social Records*, comprising information about how to retrieve a certain *Social Profile*. *Social Profile* are stored in a decentralized directory service called *Global Social Lookup System* (GSLS). This section defines and describes the architectural components, roles, and concepts of the SONIC OSNF.

- **Social Account** In SONIC, every user maintains a *Social Account*, comprising the aggregation of all information and resources associated with this user's account, including the *Social Profile* and *Social Record*.
- **Social Profile** A *Social Profile* describes content associated with a *Social Account*, which can be accessed by other users. This comprises data such as name, date of birth, or gender, but also other content such as images, exchanged messages and comments, or connections to other *Social Accounts*.
- **Social Record** A *Social Record* is a dataset that comprises all information that is required to uniquely identify and locate a *Social Account*.

A. Roles

In the SONIC OSNF users maintain *Social Accounts*, which they use to access or create content. Content is encapsulated in data objects with a standardized data format. Here, the following roles have been defined:

- **Author** The *Author* of any content is the user who initially created it. Information about authorship is indicated for every content object by a globally unique identifier, the *Global ID*.
- **Owner** Content is stored within the *Social Profile* of a user, who gains ownership for this content. In some cases, content is stored within the *Social Profile* of a user who is not the content's *Author*. Comments on user *Bob's* status updates for example are stored within *Bob's Social Profile*, although he is not the *Author* (see Section IV-C).
- **Viewer** A *Viewer* is a user accessing any content object. Depending on the access permissions set by the content's *Owner*, different *Viewers* will be provided with different content. For example, as a close friend of *Bob*, *Alice* might see different content when accessing his profile.
- **Provider** The *Provider* runs a *Platform*, on which one or more *Social Profiles* are hosted.

B. Architectural Components

The proposed SONIC protocol connects different OSN servers via loose coupling. The RESTful request/response protocol is based on HTTP and uses JSON to exchange content and messages. To verify the authenticity of content from a remote OSN platform, public key cryptography is used to digitally sign exchanged content as well as requests. In SONIC, befriending another user involves the exchange of IDs as well as public keys. The process is carried out using a central directory service that stores the *Social Records* of all users. *Social Records* contain all relevant information for connecting to another user's *Social Profile*. The protocol does not cover server-to-client communication, i.e. how content is delivered to a user's social application. The following architectural components are defined in SONIC:

- **Platform** A *Platform*, as depicted in Figure 2, is a web service that runs the business logic, i.e. provides OSN functionality as described in Section V. The platform hosts and manages *Social Profiles* and allows users to access their *Social Accounts* via a user interface, e.g. a mobile application or web page. A *Platform* is required to support the *core featureset* and provides an API for the SONIC protocol to allow cross-platform communication. Each *Platform's* API must be accessible for other *Platforms*. In addition to that, *Platforms* support an interface to connect to the GSLS.
- **Client** A *Client* is an application that is used to access a *Platform's* user interface, e.g. a website or mobile application.
- **GSLS** The *Global Social Lookup System* is a peer-to-peer-based directory service that stores and manages the *Social Records* of all users in SONIC. The GSLS provides a RESTful interface for retrieving, creating, updating and deleting *Social Records*, while cryptographic signatures prevent forged data records [7].

As stated in Section III, one requirement of SONIC is the possibility to migrate user accounts between different OSN

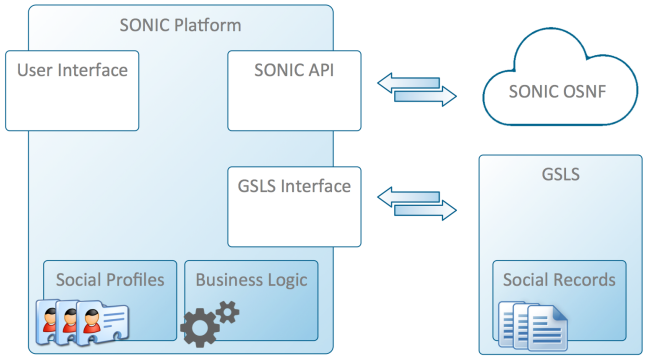

Fig. 2. A SONIC *Platform* hosts *Social Profiles* of users and provides OSN functionality. Users can access this functionality via the user interface of the *Platform*. The *Platform* is connected to the SONIC OSNF via the SONIC API, while discovery of other user's *Social Profiles* is facilitated by the GSLS

TABLE I. COMPARISON OF FEATURES OF POPULAR OSN PLATFORMS

Feature description	Facebook	Google+	Friendi.ca	Diaspora	vKontakte	RenRen	Xing	LinkedIn	Twitter
Profile page	●	●	●	●	●	●	●	●	●
Adding/following friends	●	●	●	●	●	●	●	●	●
Instant Messaging	●	●	●	○	●	●	●	●	●
Multiuser Instant Messaging	●	●	○	○	●	●	●	○	◐
Posting status updates	●	●	●	●	●	●	●	●	●
Sharing content	●	●	●	●	●	●	●	●	●
Liking content	●	●	●	◐	●	●	●	●	●
Commenting on content	●	●	●	●	●	●	●	●	●
Tagging	●	●	●	●	●	●	○	●	●
Image galleries	●	●	●	○	●	●	○	○	○
Event organization	●	●	●	○	○	●	●	○	○
Groups	●	●	○	○	●	●	●	●	○
Poking users	●	○	○	○	○	○	○	○	○

● feature is supported ◐ feature is supported with limitations ○ feature is not supported

platforms. This requires a common standard for social user accounts, which is logically decoupled from the business logic of the OSN *Platform*. Therefore, in SONIC all data associated with a user account is encapsulated in *Social Accounts*, which comprises both the actual profile data as well as further information for e.g. profile lookup.

C. Privacy concept

All content in SONIC is encapsulated in content objects with a standardized format and stored within the *Author's Social Profile*. Hence, content is stored at the user's *OSN Platform* giving him full control over the distribution of his content. For access control, each *OSN Platforms* needs to implement access control schemas. This allows users to specify who is permitted to access what parts of their *Social Profiles*. If anyhow content is related to another content object, which is stored in another *Social Profile*, it is attached to this object and stored alongside it. As of this, depending on the location of the referenced content object, content can be stored in a remote *Social Profile*. For example, a comment by a user *Alice* on a post of a user *Bob* is stored as an attachment to the post of *Bob* within *Bob's* account. In this scenario, even though *Alice* is the *Author* of the comment, *Bob* becomes it's *Owner* and is responsible for storage and delivery of the data. This allows users to maintain full control over who can access what part of their personal information while giving authors full control over what content is published related to their content (e.g. a discussion on a status update).

Sonic distinguishes between local and remote content objects. While local content is always stored at the *Author's Platform*, remote content may be stored in a remote *Social Profile*. To ensure that remote content is not altered by the *Owner*, remote content objects are signed by the *Author*. To uniquely identify content across *Platforms*, a unique *Object ID* is assigned to every content object. Remote content object are attached to other content objects by specifying a *Target ID* referring to the *Object ID* of the associated content object.

- **Local Content** Content which is stored within the *Social Profile* of its *Author* is classified as *Local Content*. As the content's *Owner* is also its *Author*, he may edit or delete it when necessary. Integrity and origin of the content can be verified by a signature created by the *Author*.
- **Remote Content** Content, which is associated with content stored within the *Social Account* of another user. It is stored as an attachment to its target content object, i.e. as part of the *Social Profile* of the *Owner* of the content. As this content is then owned by somebody other than the content's *Author*, a digital signature is created using the *Author's Account KeyPair*. This way, the content's integrity can be verified.

D. Social Record

As migrating *Social Profiles* to other *Platforms* would result in a change of the URL of a profile, links to other users' *Social Profiles* would break. Hence, SONIC uses a domain independent, globally unique identifier for *Social Accounts* - the *Global ID* - which is kept intact during the migration process. The *Global ID* is stored with information about a *Social Profile's* current location and proof of authenticity and integrity in the *Social Record*. The GSLS, which stores the *Social Records* of all users serves as a central directory service. The *Global ID* is derived directly from an RSA public key using the key derivation function PBKDF2 [35] and a salt of fixed length. The RSA key pair is also used to digitally sign the *Social Record*, rendering forgery of a *Social Record* is prevented. This way, *Global IDs* are directly bound to the public key of a *Social Account*. SONIC makes use of two different key pairs, which are used to sign content, requests, and responses to verify the authenticity and integrity of messages.

- **Account KeyPair** The *Account KeyPair* is used and managed by the *Platform* to sign content created by the account owner as well as requests and responses. This requires that the private part of the *Account KeyPair* is entrusted to the platform.
- **Personal KeyPair** The *Personal KeyPair* is used to create the *Global ID* of a *Social Account* and sign the *Social Record*. As it is required for revocation and exchange of the *Account KeyPair*, it should be kept secret and shouldn't be made accessible to the *Platform*.

As the *Account KeyPair* is used to sign content, requests, and responses in the SONIC OSNF, the *Platform* is required to store and manage the key pair. Both the personal and the account public key are included in the *Social Record*. The Personal key is used to create the *Global ID*. In order to allow revocation of the *Account KeyPair*, the *Personal Keypair* is used to both create the *Global ID* and sign the *Social Record*. This way, the *Global ID* remains unchanged when the *Account KeyPair* needs to be revoked. Revocation information is stored in the *Social Record* and signed using the *Personal KeyPair*.

V. The SONIC Protocol

The Sonic API provides access to the relevant resources of a social profile using a REST-like design. The protocol covers *Platform*-to-*Platform* communication for retrieving or pushing content to remote *Social Accounts*. As SONIC does not define any constraints how to display or utilize the exchanged information, *Platforms* are free to deliver content (e.g. a list of status updates received from multiple connected *Social Accounts*) to their users in any way. SONIC only defines a protocol and data formats defining how social information is exchanged between *Platforms*. This way, *Platforms* are free to provide any user experience they want, while allowing their users to stay connected to other users on different *Platforms*.

For authentication purposes, a set of HTTP request and response headers are specified that are used to transfer additional information required to correctly process a request. This includes the targeted API version of SONIC, a timestamp, the *Global ID* of the *Social Account* that initiated the request, and a digital signature. Content transferred with a request is required to be formatted according to the specifications of the SONIC API.

A. SONIC API

The SONIC protocol provides access to business logic of a *Platform*. Here the *core featureset* as defined in Section III-A, comprising resources *link* (adding friends), *profile* (profile page), *stream* (posting and sharing content), *comment* (commenting on other content), *like* (liking content), *tag* (tagging people in posts), and *conversation* (instant messaging).

1) LINK: The resource *link* provides access to all data associated with links to other *Social Accounts*. This includes accessing the friends roster, which lists the *Social Accounts* a user is linked to. Furthermore, new connections to other *Social Accounts* can be initiated by sending a signed *link request* object. Here, each *Platform* is responsible for defining access permissions, i.e. allowing or denying access to different resources of a user's *Social Profile* once the request has been accepted. *Platforms* may implement different authorization schemes. In SONIC, *links* between two *Social Profiles* are unidirectional.

2) PROFILE: The resource *profile* allows access to other users' *Social Profiles*. Similar to *OpenSocial's Person*, a SONIC *profile* object comprises an extensive list of information related to the *Social Profile's* owner. While only a few attributes such as *Global ID* or *displayName* are mandatory, an extensive list of other optional attributes can be specified. This includes a user's date of birth, gender, or relationship status.

3) STREAM: The *stream* resource comprises reading, creating, updating, and deleting activities such as status updates or shared content. Activities are *Activity Streams 2.0* objects with additional mandatory attributes describing *signature*, and information identifying the content's *Author*. Users may post activity objects to another user's *stream* resource, while the targeted *Social Profile Owner* may define rules to allow or deny the incoming request. This allows *Platforms* to assemble a stream of activities from linked *Social Profiles*.

4) COMMENT: The resource *comment* allows to create, retrieve, update, and delete comment objects. Comment objects contain a textual comment message, the date of creation, data identifying the comment's *Author*, and a *Target ID* that specifies the content commented on. Comments are posted in relation to content objects (e.g., a status update) and are linked to this content object via a *Target ID*. A comment is stored within the *Social Profile* of the *Owner* of the targeted content. As comments are usually linked to content from other users, they need to be signed by the *Author*. This way, any *Viewer* of the comment is able to verify its integrity and authenticity.

5) LIKE: The resource *like* allows to retrieve, create, or delete like objects. Similar to comment objects, likes are posted related to other content objects (e.g., a status update) and are linked to this content object via a *Target ID*. Likes are stored within the *Social Profile* of the *Owner* of the targeted content and need to be signed by the *Author*.

6) TAG: The resource *tag* allows to link users to content, indicating that the linked user is somehow related to the content. *Tag* objects can be created by any user and are stored within the *Social Profile* that contains the content the person is tagged in.

7) CONVERSATION: The resource *conversation* comprises textual messages exchanged between two or more individuals. A *conversation* encapsulates a list of *messages*, which are broadcast by their *Author* to all members of the same *conversation*, thus facilitating multiuser chat as well as private conversations. Further control messages are used to control the *conversation* state, e.g., users joining or leaving the conversation. This way, all participants of a *conversation* receive the same *messages*, allowing them to synchronize the state of the *conversation* between all communication participants.

VI. Implementation

The described protocol has been implemented in the open source OSN *Friendi.ca*, effectively replacing the native DFRN protocol. While the user interface and user experience of *Friendi.ca* remains unchanged, requests to other social profiles hosted on compatible platforms are performed using the SONIC protocol. Furthermore, a SONIC prototype has been implemented using the PHP framework Laravel. For evaluation purposes, the approach has been set up in a testbed with three servers, of which two run the adapted *Friendi.ca* version and the third the SONIC prototype. User accounts have been set up on all three servers to evaluate the effectiveness of the SONIC approach. Results show that the described protocol allows OSN platforms to interact with any other OSN platform given the assumption that all participating platforms support the SONIC protocol. Hence, users are no longer forced to maintain user accounts in more than one OSN and are able to create and maintain connections to other users across platform borders. The features supported by the SONIC protocol represent the *core featureset*, which has been derived by analyzing features and APIs of today's popular OSN platforms. Analysis of requests to other user accounts showed that by integrating the SONIC protocol into *Friendi.ca*, the average response time or requests between user accounts on different *Friendi.ca* servers was reduced by 54.92% compared to DFRN-based requests. At the same time, the average payload size of DFRN-based requests (6.8KB on average) was reduced by 52.8%.

VII. Conclusion

With the ongoing trend towards communicating via OSNs, there is a strong need for open protocols and standards in this area. Yet, current OSN platforms are mostly closed and proprietary solutions, which create well-calculated lock-in effects to bind users to the service. The majority of existing social APIs is not standardized and most of them offer only limited access to the platform's full functionality. We believe that OSNs can only become the main communication medium with open protocols and standards, which allow seamless inter-platform communication. While several open solutions and protocols have emerged over the last decade, they cover only certain aspects of decentralization of OSN functionality, e.g., exchanging status messages. SONIC proposes a holistic approach that facilitates seamless connectivity between different OSN platforms, thus building the foundation for an open and heterogeneous *Online Social Network Federation*. In this paper, we presented the architecture of the SONIC OSNF and provided an overview of the federation protocol. The protocol supports a *core featureset* of today's popular OSN platforms, while protocol extensions may be implemented to support additional features. By implementing the proposed protocol, OSN platforms can open their services and allow users to communicate with other OSN platforms. This would solve current problems of closed and proprietary OSN platforms such as lock-in effects or privacy issues. Using SONIC, users can freely choose an OSN platform of their liking and even migrate to another platform at a later time without losing any data or existing connections to other profiles.

Acknowledgment

The work described in this paper is based on results of ongoing research as part of the research project SONIC (grant number `01IS12056`). The project's web page is accessible at http://sonic-project.net. *SONIC* is funded as part of the *SoftwareCampus* initiative by the *German Federal Ministry of Education and Research (BMBF)* in cooperation with *EIT ICT Labs Germany GmbH*.

References

[1] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The Anatomy of the Facebook Social Graph," *arXiv preprint arXiv:1111.4503*, 2011.

[2] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca, "Decentralized Online Social Networks," in *Handbook of Social Network Technologies and Applications*. Springer, 2010, pp. 349–378.

[3] B. Fitzpatrick and D. Recordon, "Thoughts on the Social Graph," 2007, http://bradfitz.com/social-graph-problem/.

[4] W3B, "Trends im Nutzerverhalten," 2013, http://www.fittkaumaass.de/reports-und-studien/trends/nutzerverhalten-trends.

[5] A. Bleicher, "The Anti-Facebook," *IEEE Spectrum*, vol. 48, no. 6, pp. 54–82, 2011.

[6] T. Paul, A. Famulari, and T. Strufe, "A Survey on Decentralized Online Social Networks," *Computer Networks*, vol. 75, Part A, pp. 437–452, 2014.

[7] S. Göndör and H. Hebbo, "SONIC: Towards Seamless Interaction in Heterogeneous Distributed OSN Ecosystems," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*. IEEE, 2014, pp. 407–412.

[8] BITKOM, "Press release: Gezeitenwechsel bei Kurznachrichten," 2014, http://www.bitkom.org/de/presse/8477_79536.aspx.

[9] S. Göndör, F. Beierle, E. Kücükbayraktar, H. Hebbo, S. Sharhan, and A. Küpper, "Towards Migration of User Profiles in the SONIC Online Social Network Federation," in *ICCGI*. IARIA, 2015.

[10] D. Appelquist, D. Brickley, M. Carvahlo, R. Iannella, A. Passant, C. Perey, and H. Story, "A Standards-Based, Open and Privacy-Aware Social Web," *W3C Incubator Group Report*, vol. 6, 2010.

[11] S. Buchegger and A. Datta, "A Case for P2P Infrastructure for Social Networks - Opportunities & Challenges," in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*. IEEE, 2009, pp. 161–168.

[12] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "PeerSoN: P2P Social Networking: Early Experiences and Insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*. ACM, 2009, pp. 46–52.

[13] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. Epema, M. Reinders, M. R. Van Steen, and H. J. Sips, "TRIBLER: A Social-Based Peer-to-Peer System," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 2, pp. 127–138, 2008.

[14] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: A Privacy-Preserving Online Social Network Leveraging on Real-Life Trust," *Communications Magazine, IEEE*, vol. 47, no. 12, pp. 94–101, 2009.

[15] Y.-J. Chang, H.-H. Liu, L.-D. Chou, Y.-W. Chen, and H.-Y. Shin, "A General Architecture of Mobile Social Network Services," in *Convergence Information Technology, 2007. International Conference on*. IEEE, 2007, pp. 151–156.

[16] C. Wilson, T. Steinbauer, G. Wang, A. Sala, H. Zheng, and B. Y. Zhao, "Privacy, Availability and Economics in the Polaris Mobile Social Network," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*. ACM, 2011, pp. 42–47.

[17] M. Dürr, M. Maier, and F. Dorfmeister, "Vegas - A Secure and Privacy-Preserving Peer-to-Peer Online Social Network," in *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, 2012, pp. 868–874.

[18] Facebook, "The Open Graph Protocol," 2010, http://ogp.me/.

[19] Facebook Developers, "OpenGraph," 2013, https://developers.facebook.com/docs/opengraph/.

[20] M. Häsel, "OpenSocial: An Enabler for Social Applications on the Web," *Communications of the ACM*, vol. 54, no. 1, pp. 139–144, Jan. 2011.

[21] James Snell, "Activity Streams 2.0," 2015, http://www.w3.org/TR/activitystreams-core/.

[22] W3C, "OStatus," 2013, http://www.w3.org/community/ostatus/.

[23] Tent.is, "Tent.io," 2013, https://tent.io/.

[24] P. Saint-Andre, K. Smith, and R. Tronçon, *XMPP: The Definitive Guide: Building Real-Time Applications with Jabber Technologies*. OReilly, 2009.

[25] D. Brickley and L. Miller, "FOAF Vocabulary Specification 0.98," *Namespace document*, vol. 9, 2010.

[26] H. Story, B. Harbulot, I. Jacobi, and M. Jones, "FOAF+SSL: Restful Authentication for the Social Web," in *Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, 2009.

[27] W3C, "WebID 1.0 Web Identity and Discovery," 2013, http://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html.

[28] J. Smarr, "Portable Contacts 1.0 Draft C," 2008.

[29] M. Macgirvin, "DFRN - The Distributed Friends & Relations Network," 2011.

[30] J. Heidemann, M. Klier, and F. Probst, "Online Social Networks: A Survey of a Global Phenomenon," *Computer Networks*, vol. 56, no. 18, pp. 3866 – 3878, 2012.

[31] J. Heidemann, "Online Social Networks - Ein sozialer und technischer Überblick," *Informatik-Spektrum*, vol. 33, no. 3, pp. 262–271, 2010.

[32] V. A. Rohani and O. S. Hock, "On Social Network Web Sites: Definition, Features, Architectures and Analysis Tools," *Journal of Computer Engineering*, vol. 1, pp. 3–11, 2009.

[33] A. Richter and M. Koch, "Functions of Social Networking Services," in *Proc. Intl. Conf. on the Design of Cooperative Systems*, 2008, pp. 87–98.

[34] D. M. Boyd and N. B. Ellison, "Social Network Sites: Definition, History, and Scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[35] B. Kaliski, "PKCS# 5: Password-based cryptography specification version 2.0 (RFC 2898)," 2000.

Additional Information

Bibliographic Data S. Göndör, F. Beierle, S. Sherhan, H. Hebbo, E. Küçükbayraktar, and A. Küpper, “SONIC: Bridging the Gap between Different Online Social Network Platforms,” in *Proceedings of the 8th IEEE International Conference on Social Computing and Networking (SocialCom)*. IEEE, 2015, pp. 399-406.

Pre-print from https://beierle.de

Online at http://dx.doi.org/10.1109/SmartCity.2015.104

Authors Sebastian Göndör

Felix Beierle

Senan Sherhan

Hussam Hebbo

Evren Küçükbayraktar

Axel Küpper

BibTeX

```
@inproceedings{Goendoer2015SocialCom,
title = {{SONIC: Bridging the Gap between Different Online Social Network Platforms}},
author = {G\"ond\"or, Sebastian and Beierle, Felix and Sherhan, Senan and Hebbo, Hussam and K\"u\cc\"ukbayraktar, Evren and K\"upper, Axel},
booktitle = {{Proceedings of the 8th IEEE International Conference on Social Computing and Networking (SocialCom)}},
publisher = {IEEE},
year = {2015},
pages = {399-406},
doi = {10.1109/SmartCity.2015.104}
}
```