Distributed Semantic Social Networks: Architecture, Protocols and Applications

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
eingereichte

DISSERTATION

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat)

im Fachgebiet
Informatik

vorgelegt

von **Dipl.-Inf. Sebastian Tramp**

geboren am 29. September 1977 in Leipzig

Leipzig, den 25.4.2014

AUTHOR:
Dipl. Inf. Sebastian Tramp

TITLE:
Distributed Semantic Social Networks: Architecture, Protocols and Applications

INSTITUTION:
Institute of Computer Science, Faculty of Mathematics and Computer Science, University of Leipzig

BIBLIOGRAPHIC DATA:
2013, XX, 136p., 31 illus. in color., 5 tables, 20 listings

SUPERVISORS:
Prof. Dr. Klaus-Peter Fähnrich
Prof. Dr. Sören Auer

ABSTRACT

Online social networking has become one of the most popular services on the Web. Especially Facebook with its 845Mio+ monthly active users and 100Mrd+ friendship relations creates a Web inside the Web. Drawing on the metaphor of islands, Facebook is becoming more like a continent. However, users are locked up on this continent with hardly any opportunity to communicate easily with users on other islands and continents or even to relocate trans-continentally. In addition to that, privacy, data ownership and freedom of communication issues are problematically in centralized environments. The idea of distributed social networking enables users to overcome the drawbacks of centralized social networks. The goal of this thesis is to provide an architecture for distributed social networking based on semantic technologies. This architecture consists of semantic artifacts, protocols and services which enable social network applications to work in a distributed environment and with semantic interoperability. Furthermore, this thesis presents applications for distributed semantic social networking and discusses user interfaces, architecture and communication strategies for this application category.

ZUSAMMENFASSUNG

Soziale Netzwerke gehören zu den beliebtesten Online Diensten im World Wide Web. Insbesondere Facebook mit seinen mehr als 845 Mio. aktiven Nutzern im Monat und mehr als 100 Mrd. Nutzer-Beziehungen erzeugt ein eigenständiges Web im Web. Den Nutzern dieser Sozialen Netzwerke ist es jedoch schwer möglich mit Nutzern in anderen Sozialen Netzwerken zu kommunizieren oder aber mit ihren Daten in ein anderes Netzwerk zu ziehen. Zusätzlich dazu werden u.a. Privatsphäre, Eigentumsrechte an den eigenen Daten und uneingeschränkte Freiheit in der Kommunikation als problematisch empfunden. Die Idee verteilter Soziale Netzwerke ermöglicht es, diese Probleme zentralisierter Sozialer Netzwerke zu überwinden. Das Ziel dieser Arbeit ist die Darstellung einer Architektur verteilter Soziale Netzwerke welche auf semantischen Technologien basiert. Diese Architektur besteht aus semantischen Artefakten, Protokollen und Diensten und ermöglicht die Kommunikation von Sozialen Anwendungen in einer verteilten Infrastruktur. Darüber hinaus präsentiert diese Arbeit mehrere Applikationen für verteilte semantische Soziale Netzwerke und diskutiert deren Nutzer-Schnittstellen, Architektur und Kommunikationsstrategien.

PUBLICATIONS

RELATED IN THE CONTEXT OF THIS THESIS

Some ideas and figures have appeared previously in the following publications. Please note that the author of this thesis is listed with two names (S.Tramp and S.Dietzold).

S. Auer, S. Dietzold, and T. Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749, Berlin / Heidelberg, 2006. Springer. ISBN 3-540-49029-9. doi: 10.1007/11926078_53.

S. Auer, S. Dietzold, J. Lehmann, and T. Riechert. OntoWiki: A tool for social, semantic collaboration. In N. F. Noy, H. Alani, G. Stumme, P. Mika, Y. Sure, and D. Vrandecic, editors, *Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at the 16th International World Wide Web Conference (WWW2007) Banff, Canada, May 8, 2007*, volume 273 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007a.

S. Auer, S. Dietzold, and T. Riechert. Social Software für Kollaborative Wissensarbeit. In C. Müller and N. Gronau, editors, *Analyse sozialer Netzwerke und Social Software - Grundlagen und Anwendungsbeispiele*, pages 235–256. GITO-Verlag – Expertenwissen für die industrielle Praxis, 2007b.

S. Auer, S. Dietzold, and M. Martin. Entwicklung semantischer Webapplikationen: Auf dem Weg vom Dokumenten- zum Daten-Web. *T3N Magazin*, 12:30–33, 2008.

S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify: Light-weight linked data publication from relational databases. In J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 621–630. ACM, 2009. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526793.

S. Dietzold and S. Auer. Access Control on RDF Triple Stores from a Semantic Wiki Perspective. In C. Bizer, S. Auer, and L. Miller, editors, *Proc. of 2nd Workshop on Scripting for the Semantic Web at ESWC, Budva, Montenegro, June 12, 2006*, volume 183 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2006.

S. Dietzold and S. Auer. Realisierung von Sozialen Netzwerken im Semantic Web mit OntoWiki. *i-com - Zeitschrift für interaktive und kooperative Medien*, 3:20–24, 2009. doi: 10.1524/icom.2009.0032.

S. Dietzold, S. Auer, and T. Riechert. Kolloborative Wissensarbeit mit OntoWiki. In *Proceedings of the INFORMATIK 2006 Workshop: Bildung von Sozialen Netzwerken in Anwendungen der "Social Software"*, 2006.

S. Dietzold, T. Riechert, and S. Auer. Semantische Datenintegration mit Hilfe von Semantic Web und Wiki-Technologien. In K.-P. Fähnrich, M. Thränert, and P. Wetzel, editors, *Integration Engineering: Motivation – Begriffe – Methoden – Anwendungsfälle*, Leipziger Beiträge zur Informatik, pages 277–283. Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2007.

S. Dietzold, S. Hellmann, and M. Peklo. Using JavaScript RDFa Widgets for Model/View Separation inside Read/Write Websites. In *Proceedings of the 4th Workshop on Scripting for the Semantic Web*, 2008a.

S. Dietzold, J. Unbehauen, and S. Auer. xOperator - Interconnecting the Semantic Web and Instant Messaging Networks. In *Proceedings of 5th European Semantic Web Conference (ESWC 2008), 1-5 June, 2008, Tenerife, Spain.*, pages 19–33, 2008b.

S. Dietzold, J. Unbehauen, and S. Auer. xOperator - An Extensible Semantic Agent for Instant Messaging Networks. In *Proceedings of 5th European Semantic Web Conference (ESWC 2008), 1-5 June, 2008, Tenerife, Spain.*, pages 787—791, 2008c.

T. Ermilov, N. Heino, S. Tramp, and S. Auer. OntoWiki Mobile — Knowledge Management in your Pocket. In *Proceedings of the ESWC2011*, 2011.

N. Heino, S. Dietzold, M. Martin, and S. Auer. Developing Semantic Web Applications with the OntoWiki Framework. In T. Pellegrini, S. Auer, K. Tochtermann, and S. Schaffert, editors, *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 61–77. Springer, Berlin / Heidelberg, 2009. doi: 10.1007/978-3-642-02184-8_5.

C. Rieß, N. Heino, S. Tramp, and S. Auer. EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, Lecture Notes in Computer Science, Berlin / Heidelberg, 2010. Springer. doi: 10.1007/978-3-642-17746-0_41.

H. Story, A. Sambra, and S. Tramp. Friending On The Social Web. In *Federated Social Web Europe 2011, Berlin June 3rd-5th 2011*, 2011.

S. Tramp, P. Frischmuth, T. Ermilov, and S. Auer. Weaving a Social Data Web with Semantic Pingback. In P. Cimiano and H. Pinto,

editors, *Proceedings of the EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses; 11th October-15th October 2010 - Lisbon, Portugal*, volume 6317 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 135–149, Berlin / Heidelberg, October 2010a. Springer. doi: 10.1007/978-3-642-16438-5_10.

S. Tramp, P. Frischmuth, and N. Heino. OntoWiki – a Semantic Data Wiki Enabling the Collaborative Creation and (Linked Data) Publication of RDF Knowledge Bases. In O. Corcho and J. Voelker, editors, *Demo Proceedings of the EKAW 2010*, October 2010b.

S. Tramp, N. Heino, S. Auer, and P. Frischmuth. Making the Semantic Data Web easily writeable with RDFauthor. In L. A. et al., editor, *Proceedings of 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6089 of *Lecture Notes in Computer Science*, pages 436—440, Berlin / Heidelberg, 2010c. Springer. doi: 10.1007/978-3-642-13489-0_39.

S. Tramp, N. Heino, S. Auer, and P. Frischmuth. RDFauthor: Employing RDFa for collaborative Knowledge Engineering. In P. Cimiano and H. Pinto, editors, *Proceedings of the EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses; 11th October-15th October 2010 - Lisbon, Portugal*, volume 6317 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 90–104, Berlin / Heidelberg, October 2010d. Springer.

S. Tramp, T. Ermilov, P. Frischmuth, and S. Auer. Architecture of a Distributed Semantic Social Network. In *Federated Social Web Europe 2011, Berlin June 3rd-5th 2011*, 2011a.

S. Tramp, P. Frischmuth, N. Arndt, T. Ermilov, and S. Auer. Weaving a Distributed, Semantic Social Network for Mobile Users. In *Proceedings of the ESWC2011*, 2011b.

S. Tramp, H. Story, A. Sambra, P. Frischmuth, M. Martin, and S. Auer. Extending the WebID Protocol with Access Delegation. In A. Harth, O. Hartig, and J. Sequeda, editors, *Proceedings of the Third International Workshop on Consuming Linked Data (COLD2012)*, CEUR Workshop Proceedings. CEUR-WS.org, 2012.

S. Tramp, P. Frischmuth, T. Ermilov, S. Shekarpour, and S. Auer. An Architecture of a Distributed Semantic Social Network. *Semantic Web*, 5(1):77–95, 2014.

J. Unbehauen, S. Hellmann, M. Martin, S. Dietzold, and S. Auer. xOperator - Chat with the Semantic Web, 2008. Poster @ the ISWC 2008.

RELATED IN THE CONTEXT OF SEMANTIC WEB RESEARCH IN GENERAL

In addition to these publications, the following publications of the author are not directly related to the topic of this thesis but nethertheless cover the same research area in a more general sense:

S. Auer, R. Doehring, and S. Dietzold. LESS - Template-Based Syndication and Presentation of Linked Data. In *Proceedings of 7th Extended Semantic Web Conference (ESWC2010) 30 May – 3 June, Heraklion, Greece*, May 2010. doi: 10.1007/978-3-642-13489-0_15.

S. Auer, L. Bühmann, C. Dirschl, O. Erling, M. Hausenblas, R. Isele, J. Lehmann, M. Martin, P. N. Mendes, B. van Nuffelen, C. Stadler, S. Tramp, and H. Williams. Managing the life-cycle of Linked Data with the LOD2 Stack. In *Proceedings of International Semantic Web Conference (ISWC 2012)*, 2012. 22

T. Berger, S. Dietzold, and T. Riechert. Der Einsatz semantischer Daten-Wikis in frühen Phasen des Requirements Engineering. In S. Auer, K. Lauenroth, S. Lohmann, and T. Riechert, editors, *Agiles Requirements Engineering für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder*, volume XVIII of *Leipziger Beiträge zur Informatik*, pages 27–38. Leipziger Informatik-Verbund (LIV), 2009.

S. Dietzold. Generating RDF Models from LDAP Directories. In S. Auer, C. Bizer, and L. Miller, editors, *Proceedings of the SFSW 05 Workshop on Scripting for the Semantic Web , Hersonissos, Crete, Greece, May 30, 2005*, volume 135 of *CEUR Workshop Proceedings*. CEUR-WS, 2005.

S. Dietzold and S. Auer. Integrating SPARQL Endpoints into Directory Services. In S. Auer, C. Bizer, T. Heath, and G. A. Grimnes, editors, *Proceedings of the ESWC'07 Workshop on Scripting for the Semantic Web, SFSW 2007, Innsbruck, Austria, May 30, 2007*, volume 248 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007a.

S. Dietzold and S. Auer. Accessing RDF Knowledge Bases via LDAP Clients. In T. Pellegrini and S. Schaffert, editors, *Proceedings of International Conference Semantics Systems 2007, I-SEMANTICS'07, Graz, Austria; September 5 – 7, 2007*, pages 290–296. Journal of Univeral Computer Science, 09 2007b.

S. Dietzold and T. Riechert. Realisierung einer Web-basierten Plattform für das verteilte Requirements Engineering auf Basis des Application-Frameworks OntoWiki. In S. Auer, K. Lauenroth, S. Lohmann, and T. Riechert, editors, *Agiles Requirements Engineering*

für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder, volume XVIII of *Leipziger Beiträge zur Informatik*, pages 49–58. Leipziger Informatik-Verbund (LIV), 2009.

P. Frischmuth, T. Riechert, and S. Tramp. Realisierung einer webbasierten Plattform für die verteilte Akquise von Professorendaten auf Basis des OntoWiki-Frameworks. In U. Morgenstern and T. Riechert, editors, *Catalogus Professorum Lipsiensis – Konzeption, technische Umsetzung und Anwendungen für Professorenkataloge im Semantic Web*, volume XXI of *Leipziger Beiträge zur Informatik*, pages 65–75. Leipziger Informatik-Verbund (LIV), 2010.

D. Gerber, M. Frommhold, M. Martin, S. Tramp, and S. Auer. Learning Semantic Web Technologies with the Web-Based SPARQLTrainer. In *Proceedings of the 6th International Conference on Semantic Systems 2010*, ACM, Graz / Austria, September 2010.

N. Heino, S. Tramp, and S. Auer. Managing Web Content using Linked Data Principles — Combining semantic structure with dynamic content syndication. In *Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference (COMPSAC 2011)*. IEEE Computer Society, 2011.

J. Lehmann, S. Auer, L. Bühmann, and S. Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9:71 – 81, 2011.

S. Lohmann, S. Dietzold, S. Auer, and J. Ziegler, editors. *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW 2008)*, volume 417 of *CEUR Workshop Proceedings*, December 2008a.

S. Lohmann, P. Heim, S. Auer, S. Dietzold, and T. Riechert. Semantifying Requirements Engineering – The SoftWiki Approach. In *Proceedings of the 4th International Conference on Semantic Technologies (I-SEMANTICS '08)*, J.UCS, pages 182–185, 2008b.

S. Otto and S. Dietzold. Caucasian Spiders - A faunistic Database on the spiders of the Caucasus - http://caucasus-spiders.info. *Newsl. Brit. Arachn. Soc.*, 108:14, 2007.

T. Riechert, U. Morgenstern, S. Auer, S. Tramp, and M. Martin. Knowledge Engineering for Historians on the Example of the Catalogus Professorum Lipsiensis. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, and B. Glimm, editors, *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, volume 6497 of *Lecture Notes in Computer Science*, pages 225–240, Shanghai / China, 2010a. Springer. doi: 10.1007/978-3-642-17749-1_15.

T. Riechert, U. Morgenstern, S. Auer, S. Tramp, and M. Martin. The Catalogus Professorum Lipsiensis – Semantics-based Collaboration and Exploration for Historians. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, Lecture Notes in Computer Science, Shanghai / China, 2010b. Springer.

C. Rieß, N. Heino, S. Tramp, and S. Auer. EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, Lecture Notes in Computer Science, Berlin / Heidelberg, 2010. Springer. doi: 10.1007/978-3-642-17746-0_41.

ACKNOWLEDGMENTS

The thesis was written within the research network Agile Knowledge Engineering and Semantic Web (AKSW) hosted by the Chair of Business Information Systems (BIS) at the University of Leipzig. I thank my supervisors Prof. Dr. Sören Auer and Prof. Dr. Klaus-Peter Fähnrich for granting me the freedom to develop and pursue my research ideas that have lead to this work.

I want to thank all my colleagues in the AKSW network – I am glad to be part of this group! I thank all my co-authors and committers of the respective open source projects. In particular, I want to thank Philipp Frischmuth, Michael Martin, Jörg Unbehauen, Tim Ermilov, Nathanael Arndt, Norman Heino and Thomas Riechert for their scientific input and implementation work. I want to thank Nadine Jänicke and Amrapali Zaveri for their support in language issues and Daniel for sitting beside me in the library.

Finally, I thank Tina for supporting me in stressful times and managing our family.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

LIST OF USED PREFIX AND NAMESPACES

This document references the following namespaces for datasets, schemas or ontologies by using these corresponding prefixes.

`aair` . . . . . `http://xmlns.notu.be/aair#`
This specification describes the Atom Activity Streams in RDF Vocabulary (AAIR), defined as a dictionary of named properties and classes using W3C's RDF technology, and specifically a mapping of the Atom Activity Streams work to RDF [Minno and Palmisano, 2010] 39

`acontacts` . . . . . `http://ns.aksw.org/Android/`
A native Android system vocabulary which represents the Android contacts database defined by the Android API. This vocabulary is deeply integrated into the Android system since it re-uses class and attribute names from the Android API and represents them as OWL class and datatype properties [Tramp et al., 2011b] 86

`cc` . . . . . `http://creativecommons.org/ns#`
Describing Copyright in RDF – The Creative Commons Rights Expression Language is a small schema to link lincense documents and describe usage rules. 101

`cert` . . . . . `http://www.w3.org/ns/auth/cert#`
Ontology for Certificates and crypto stuff, authored by Henry Story . . 59

`dbpedia` . . . . . `http://dbpedia.org/resource/`
The DBpedia namespace is the root for all extracted concepts from the english Wikipedia [Auer et al., 2007] 42

`dc` . . . . . `http://purl.org/dc/elements/1.1/`
Dublin Core Metadata Element Set, Version 1.1 54

`dct` . . . . . `http://purl.org/dc/terms/`
This document is an up-to-date specification of all metadata terms maintained by the Dublin Core Metadata Initiative, including properties, vocabulary encoding schemes, syntax encoding schemes, and classes. 105

`dssn` . . . . . `http://purl.org/net/dssn/`
Vocabulary to describe some Distributed Semantic Social Network relations [Tramp et al., 2014] 42

`foaf` . . . . . `http://xmlns.com/foaf/0.1/`
The Friend of a Friend (FOAF) RDF vocabulary 38

Part I

INTRODUCTION AND PRELIMINARIES

In Chapter 1 we outline major problems of centralized Social Networks. These issues were the main driving force behind our research and affect all requirements which were constructed for all parts of the thesis. We then lead over to the main reasearch questions which we want to answer in this thesis.

In Chapter 2 we give an overiew on the Semantic Web and its different technologies. In addition to that we list and evaluate the current state of the art regarding distributed and federated Social Web applications and frameworks.

Finally, in Chapter 3 we outline the thesis structure and emphasize specific contributions made to the research field.

1 MOTIVATION AND RESEARCH QUESTIONS

Online social networking has become one of the most popular services on the Web. Especially Facebook with its 845Mio+ monthly active users and 100Mrd+ friendship relations creates a Web inside the Web[1]. The growth of Social Media services seems not to end in a short future. According to Jones [2013], both Facebook and Google+ and have an user base of over 1 Billion registered users with an increasing trend (cf. Figure 1).

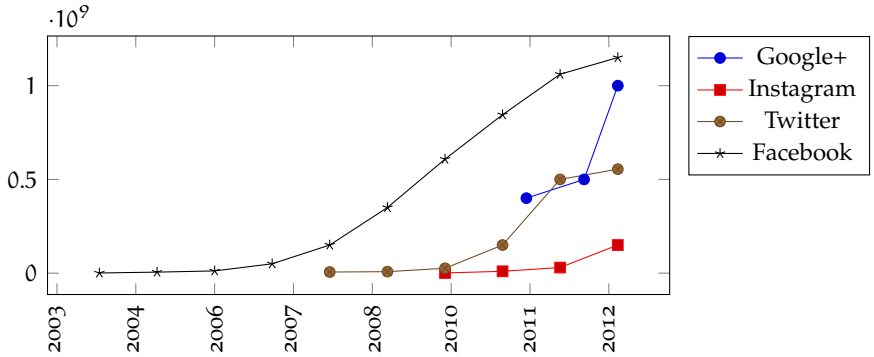

Figure 1: The growth of social networks registered user accounts in the last ten years (image recreated according to the data in Jones [2013])

Drawing on the metaphor of islands, Facebook is becoming more like a continent. However, users are locked up on this continent with hardly any opportunity to communicate easily with users on other islands and continents or even to relocate trans-continentally. Users are bound to a certain platform and hardly have the chance to migrate easily to another social networking platform if they want to preserve their connections. Once users have published their personal information within a social network, they often also lose control over the data they own, since it is stored on a single company's servers. Interoperability between platforms is very rudimentary and largely limited to proprietary APIs. In order to keep data up-to-date on multiple platforms, users have to modify the data on every single platform or information will diverge. Since there are only a few large social networking players, the Web also loses its distributed nature. According to a recent comScore study[2], Facebook usage times already outnumber

1 http://www.sec.gov/Archives/edgar/data/1326801/000119312512034517/d287954ds1.htm

2 http://allthingsd.com/20110623/the-web-is-shrinking-now-what/

traditional Web usage by factor two and this divergence is continuing to increase.

We argue that solutions to social networking should be engineered in distributed fashion so that users are empowered to regain control over their data. The currently vast oceans between social networking continents and islands should be bridged by high-speed connections allowing data and users to travel easily and quickly between these places. In fact, we envision the currently few social networking continents to be complemented by a large number of smaller islands with a tight network of bridges and ferry connections between them. Compared with the currently prevalent centralized social networks such an approach has a number of advantages, which are the most important motivational points for this work.

The following subsections introduce our motivational points more in detail. Each subsection complemented with real-world examples from the last ten years.

PRIVACY

Users of the distributed semantic social network (DSSN) can setup their own DSSN node or chose a DSSN node provider with particularly strict privacy rules in order to ensure a maximum of privacy. This would facilitate a competition of social network operators about the privacy rules most beneficial for users. Currently, due to the oligopoly in the social networking market, which is dominated by big players such as Facebook, Google or Twitter, privacy regulations are often more driven by commercialization interests. Such privacy policies are not subject of individual discussions or agreements. A user can agree on a policies or she is not allowed to use the service. Even partial agreement is not possible. In addition to that, privacy policies can be subject of changes in the future. Also here, the user is able to agree or has to quit using the service.

Considerable examples of privacy issues are:

- In 2010, Facebook was criticized because of a personal information aggregation techniques called *connections*[3]. A *connection* is created whenever a Facebook user *likes* a resource either on the Web or inside the social network (cf. Figure 2). Facebook treats such relationships to other resources as public information, and the identity of the Facebook user may be displayed on the Facebook page of the product or service.

- In 2011, Google launched its own social network service Google+. To join the service, users had to fill mandatory real-name and gender information. Google treated these information as public, using the gender information for adapting the user interface,

3 https://www.eff.org/deeplinks/2010/04/handy-facebook-english-translator

Figure 2: A typical *Like* button for social network integration: clicking the button as well as visiting the page creates a connection between the users account and the website resource.

such as changing words like *he*, *she* and *they*. Even on the most private settings, non-friends could still see names, gender information and profile pictures[4].

DATA SECURITY

Due to the distributed nature it is more difficult to steal large amounts of private data. Also, security is ensured through public review and testing of open-standards and to a lesser extend through obscurity due to closed proprietary implementations. As is confirmed very frequently, centralized solutions are always more endangered of attacks on data security. Even with the best technical solutions in place, insider threats can hardly be prevented in a centralized setting but can not cause that much harm to a DSSN.

The history of social network security breaches include these more or less well known events:

- The PlayStation Network is a social network for online multiplayer gaming which is provided by Sony Computer Entertainment. In 2011, this service had a major security breach and outage which affected 77 million registered accounts[5] (cf. Figure 3). As reported by Sony, user data was stolen in a huge dimension. This included names, adresses, birthdays, purchase history, passwords and other important data items.

- In 2013, Facebook infrastructure was hacked by using a zero day Java exploit deployed on a mobile developer website which was visited by multiple Facebook employees[6]. The compromised website allowed malware to be installed on these employee

4 http://www.pcmag.com/article2/0,2817,2388120,00.asp
5 http://en.wikipedia.org/wiki/PlayStation_Network_outage
6 http://www.cnet.com/news/facebook-says-it-was-hacked-claims-member-data-safe/

Figure 3: The PSN security breach of 2011 affected 77 million registered user accounts and its data.

laptops. In an official statement, Facebook claimed that no user data was compromised[7].

DATA OWNERSHIP

Users can have full ownership and control over the use of their data. They are not restricted to ownership regulations imposed by their social network provider. Instead DSSN users can implement fine-grained data licensing options according to their needs. A DSSN would moreover facilitate a competition of DSSN node providers for the most liberal and beneficial data ownership regulations for users.

- In 2008, the Facebook account of the famous blogger Robert Scoble was blocked because he ran a screen scraper on his Facebook account in order to export his Facebook profile including information about his friends[8]. The story was a starting point for a fascinating debate on who is the owner of the data and can export it. Facebook was criticized for not allowing their users to export their data (a feature which is no available).
- The process of deleting a Facebook account and its data is continuously criticized by multiple organisations. In 2010, Facebook began allowing users to permanently delete their accounts in

7 https://www.facebook.com/notes/facebook-security/protecting-people-on-facebook/10151249208250766

8 http://scripting.com/stories/2008/01/03/scobleAndHisFacebookData.html

order to react on critical articles from important publications such as the New York Times[9].

EXTENSIBILITY

The representation of social network resources like WebIDs and data artefacts is not limited to a specific schema and can grow with the needs of the users. Although extensibility is also easily to realize in the centralized setting (as is confirmed by various APIs such as Open Social), a centralized social network setting could easily prohibit (or censor) certain extensions for commercial (or political) reasons and thus constrain the freedom of its users.

- In 2013, Google disabled the Google+ feature to embed videos from Vimeo in its posts. Vimeo is a video-sharing website with emphasis on high definition playback and professional users and a competitor to Googles own video-sharing platform Youtube.

RELIABILITY

Again due to the distributedness the DSSN is much less endangered of breakdowns or cyberterrorism, such as denial-of-service attacks.

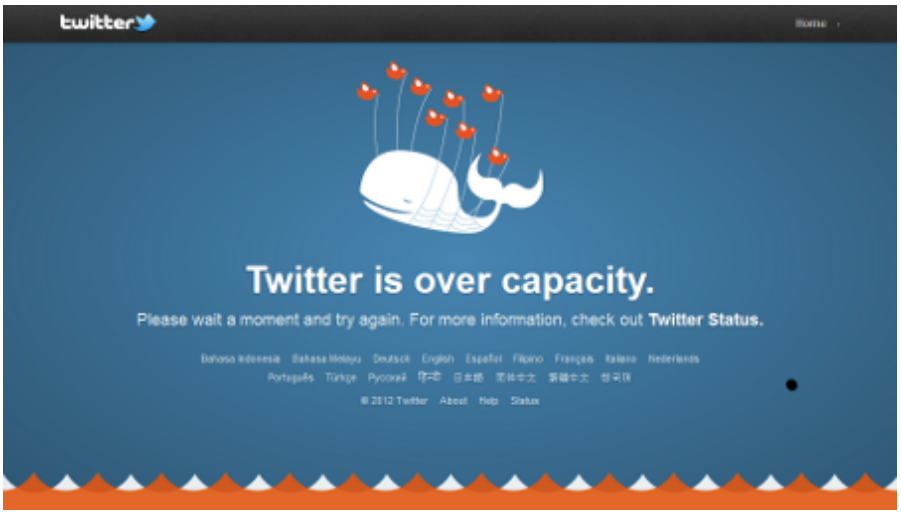

Figure 4: Twitters fail whale is a pictorial synonym for unreliability of a service.

- In 2012, Facebook was not available in Germany and other European countries for some hours due to an error in the DNS

9 http://www.nytimes.com/2013/04/18/technology/personaltech/how-to-sever-ties-to-social-networks-and-other-web-sites.html

resolution. In 2013, technical problems in Facebooks infrastructure were the reason for worldwide access problems to the social network and its payment service.

- The social networking and microblogging service *Twitter* has a long history of outages due to an overloading amount of users and messages. The service was long time very unstable due to the scale of growth of the user base. When the service experiences an outage, users see the *fail whale* error message (cf. Figure 4) which is by now a well know internet meme.

FREEDOM OF COMMUNICATION

As we observed recently during the Arab Spring where social networking services helped protesters to organize them self, social networks can play a crucial role in attaining and defeating civil liberties. A centralized infrastructure can be more easily controlled and suppressed by powerful governmental and non-governmental organizations in order to hinder people to communicate and organize them self. This is not an exception but rather the norm, as Reporters without Borders document yearly in their *Enemies of the Internet* report[10] (cf. Figure 5). A DSSN with a vast amount of nodes is much less endangered to these attacks.

- The Chinese government frequently blackouts social networking services such as Twitter and Flickr[11] because of political commentary and self organization[12].
- In 2014, Twitter was blackouted by the Turkish government a few days ahead of local elections. This ban was a direct reaction of the Turkish prime minister Recep Tayyip Erdoğan after audio recordings of his alleged conversations suggesting corruption were leaked[13].

All these points motivate our research towards distributed social networks based on semantic technologies. More in detail, the folowing research questions will be addressed.

10 Downloaded from http://12mars.rsf.org/2014-en/ and used under terms of Creative Commons CC BY-NC-SA 3.0

11 http://usatoday30.usatoday.com/tech/news/2009-06-02-china-twitter-tiananmen-protests_N.htm

12 http://news.smh.com.au/breaking-news-technology/internet-activists-discuss-online-democracy-20091122-isc9.html

13 http://www.theguardian.com/world/2014/mar/21/turkey-blocks-twitter-prime-minister

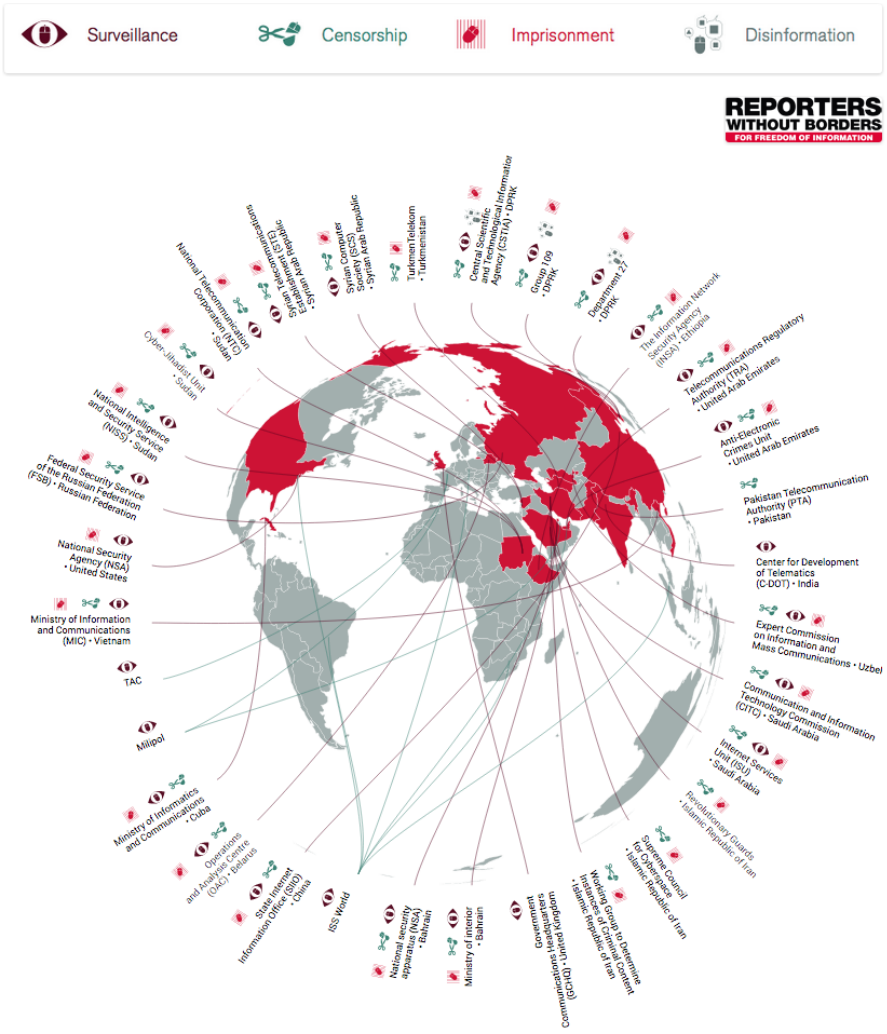

Figure 5: Map of the Institutions in Enemies of the Internet by *Reporters without Borders*

RESEARCH QUESTIONS

The research reported in this thesis addresses the following principal question:

> To what extent can Semantic Web technologies be deployed to support the structure, the maintenance as well as the usage of Distributed Social Networks on the World Wide Web?

This principal question can be broken down into a number of specific research questions:

Q1: Which overall architecture based on Semantic Web technologies is suitable in a distributed environment to build Social Networks?

The overall architecture of a proposed Distributed Semantic Social Network is the minimal core on which all participants have to agree to be interoperable in a global scale. Which building blocks are used to form such an architecture and how do they interact with each other?

Q2: Which protocols and data definition standards are needed to provide a robust framework for social interactions?

This question goes into detail with protocols and data definition: Which specific protocols need to be established to form a network which is extensible and robust? Which schemata and ontologies should be used to describe user generated content with Semantic Web standards?

Q3: Which services are needed in such an environment and which infrastructure is needed to provide these services?

Since centralized social networks pool different services to a combined interface, we have to think about a minimal infrastructure of services which need to be established in a way that they can handle similar requests but as a distributed team.

Q4: To what extent can we use and extend the existing architecture of the World Wide Web which has proved to be scalable and stable?

The World Wide Web is growing constantly and up to now there seems to be no limitation for this process. The Architecture of the Web is a major reason for this development and a proposed DSSN architecture should emphasize the scalable parts of the more general Web architecture in order to reuse it. In addition to that we need to clarify, which parts need to be extended.

Q5: How can we enable existing social applications to be part of a DSSN?

The best way of growing is to integrate existing resources. For our proposed architecture we need to develop methods and best practice on integrating existing social web applications such as wiki, blog and forum applications.

Q6: Which types of applications can provide user interfaces to a DSSN?

A distributed environment has certain drawbacks compared to a centralized environment. DSSN applications need to hide these drawbacks from the user and should provide similar used interfaces in terms of performance, usability and productivity. This goal rises different requirements for the underlying semantic technology which should be tackled too.

Q7: How can we support the transformation of social data from existing social applications?

Part of the integration of existing applications is the transformation of a social web applications database in order to allow other applications access these data in the same way as they can access native DSSN data resources.

2

BACKGROUND AND STATE OF THE ART

In this Chapter we give an overiew on the Semantic Web and its different technologies. In addition to that we list and evaluate the current state of the art regarding distributed and federated Social Web applications and frameworks.

2.1 THE SEMANTIC WEB

> The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. — W3C Semantic Web Activity Homepage[14]

This quotation captures the essence and goals of all activities around the well known buzz word *Semantic Web*. The Semantic Web is based on the idea of preparing content and data so as to allow machines to read and process it in a way which goes far beyond the simple presentation and indexation of dump data such as plain text and plain images today. In Berners-Lee et al. [2001] the authors define it as

> ... not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

This extension allows for an evolution of the Web from a document-centric Web to a data- and information-centric Web with a strong emphasis on data integration by using globally shared semantic concepts. In Shadbolt et al. [2006] the progress of this evolution is outlined as a process of *specifying, developing and deploying languages for shared meaning*. These languages provide a foundation for semantic interoperability and, together with protocols, the framework for sharing machine-readable content on the Web.

The Semantic Web is based on the World Wide Web (WWW) as described in Jacobs and Walsh [2004] and re-uses or re-defines key concepts of the WWW such as Universal Resource Identifiers (URI, Berners-Lee et al. [2005]) or the Hypertext Transfer Protocol (HTTP, Fielding et al. [1999]).

In the context of this thesis the important parts of the Semantic Web framework include: the Resource Description Framework (RDF) with its core concepts and serializations (Section 2.1.1), the concept of

14 http://www.w3.org/2001/sw/

Linked Data (Section 2.1.3), and an introduction to the query language SPARQL (Section 2.1.4).

A more comprehensive introduction to the topic is provided by Hitzler et al. [2007] and Heath and Bizer [2011].

2.1.1 *Resource Description Framework*

2.1.1.1 *Concepts*

According to Schreiber and Raimond [2014] the RDF data model consists of the following basic concepts:

RESOURCE is a term to denote all things in the Web as well as in the real world which we can identify. This range of things includes documents, images and applications from the web as well as physical objects, abstract concepts and even fantasy heroes from the real world. The term is used in the broadest sense imaginate.

IRI is an abbreviation for International Resource Identifier (*IRI*, Duerst and Suignard [2005]) which is a sequence of Unicode characters. IRIs are used to identify resources. A subset of IRIs are URLs, which provide not only an identifier but also a locator for a resource in order to allow to access this resource with an application.

QNAME IRIs can be written abbreviated by using a *QName* (qualified name). Qnames were introduced by Bray et al. [2009] in order to be used as IRI references. They consist of a prefix part and a local part in the form `prefix:local`. While the prefix part identifies a local abbrevation for a namespace, the local part together with the prefix identifies a resource inside that namespace. This thesis uses hence a list of prefix / namespace abbreviations, which are are listed on Page xviii.

LITERAL is a sequence of Unicode characters which are associated with a datatype and an optional language tag.

DATATYPE is a resource identified by an IRI. Datatype definitions enable application to parse and interpret a literal correctly. RDF re-uses many of the XML Schema datatypes from Peterson et al. [2012] as normative, such as `xsd:string` and `xsd:date`, and defines two additional non-normative datatypes: `rdf:HTML` and `rdf:XMLLiteral`. In addition to the normative datatypes, application-specific datatypes can be used and expressed.

LANGUAGE TAGS can be associated only if the literal is of type `rdf:langString` (the datatype of language-tagged string values). Language tags identify a language or sub-language as defined in Phillips and Davis [2009].

BLANK NODES represent resources in the same way as IRIs do, however they do not identify them globally (but locally). They are used if the IRI of a resource is not (yet) known or if it is not needed to identify a resource in a global scope.

Based on these fundamental concepts, the following advanced concepts form the foundation of RDF.

TRIPLE is the most granular knowledge item which can be managed by RDF. The terms *triple* and *statements* are equivalent. Statements always have the following structure, that consists of three components (hence the name *triple*):

```
<subject> <predicate> <object>
```

Statements express a relationship between the `<subject>` and the `<object>`. The type (or nature) of this relationship is stated by the `<predicate>`. The relation is always unidirectional from the `<subject>` to the `<object>`.

On the `<subject>` position of a triple, IRIs and blank nodes can be used. On the `<predicate>` position of a triple, IRIs can be used. On the `<object>` position of a triple, IRIs, blank nodes and literals can be used.

Here is an example triple which uses all available concepts:

```
<[]> <rdfs:label> <"DSSN"^^rdf:langString@en>
```

In this example the three components have the following values:

- The `<subject>` is a blank node resource.
- The `<predicate>` is a resource from the `rdfs` namespace (cf. Section 2.1.2) with the local name `label`.
- The `<object>` is a literal with the value of `DSSN`, an associated datatype and a language tag for the English language.

RDF statements are identified by their components only. That is, two statements with the same `<subject>`, `<predicate>` and `<object>` are regarded as identical.

GRAPHS are used to manage different collections of triples. A graph can be identified by an IRI which means graphs are also resources as well and can be subject and object of an RDF triple.

DATASET A new concept in RDF 1.1 is the term of a *dataset*. An RDF dataset is a collection of RDF graphs, and comprises:

- Exactly one default graph, being an RDF graph. The default graph does not have a name and may be empty.
- Zero or more named graphs. Each named graph is a pair consisting of an IRI or a blank node (the graph name), and an RDF graph. Graph names are unique within an RDF dataset.

[Cyganiak et al., 2014]

2.1.1.2 *Serializations*

In order to allow machines to read and write RDF data, the following RDF serializations are standardized:

RDF/XML [Beckett, 2004] was the first syntax recommended by the W3C. It defines a syntax for RDF graphs in XML. RDF/XML is criticized for performance and complexity issues. Nevertheless RDF/XML has a broad tool support and is required by the Web Ontology Language (OWL, cf. Section 2.1.2) as an interchange format.

TURTLE [Beckett and Berners-Lee, 2004] started as a small syntax to list RDF in scientific papers. It is very famous because of its simplicity and is designed to be written by humans. Turtle supports Qnames / namespace declarations as well as some abbreviations to avoid double content. All examples in this thesis are written in turtle.

TRIG [Bizer and Cyganiak, 2007] is an extension to Turtle which supports RDF 1.1 datasets.

N-TRIPLES [Carothers and Seaborne, 2014] are the easiest way to write RDF triples. Each line represents one triple. Neither Qnames / namespaces nor datasets and newlines have to be encoded. N-Triples started as a line-based format to represent the correct answers for parsing RDF/XML test cases. It is known to be the fastest syntax available for parsing.

N-QUADS [Carothers, 2014] is an extension to N-Triples which supports RDF 1.1 datasets.

JSON-LD [Sporny et al., 2014] is, rather, a method to transform JSON documents to RDF graphs, than a new syntax. It is based on the concept of a context where JSON structures are mapped to RDF properties and classes.

Apart from these common serializations, RDF can be represented in HTML with RDFa [Adida et al., 2008] or as plain JSON [Davis et al., 2013].

2.1.2 *Schema and Ontology Languages*

The described basic RDF data model with triples, graphs and datasets is far away from being a universal framework to share and re-use data across application, enterprise, and community boundaries. To be able to do this, one needs to specify not only the structure and syntax of the data but also the semantic in terms of a logical language. The

common understanding of the Semantic Web community is, to use description logics to describe semantics in RDF. To achieve this goal, two standards are recommended by the W3C:

RDF SCHEMA (RDFS, Brickley and Guha [2014]) represents the first level of terminology languages created on top of plain RDF. RDFS provides a vocabulary for describing simple class and property hierarchies and is used with the prefix `rdfs` for qnames. Apart from the definition of fundamental top terms[15], the most important terms which are provided by RDFS are terms to describe classes and properties[16].

`rdfs:Class` represents the class of all classes. To define an instance of a class resource, the property `rdf:type` is used. Classes can be defined in a hierarchy using the `rdfs:subClassOf` property. All instances of a subclass will also be instances of the superclass too.

`rdfs:Property` represents the class of all properties. All resources which are used at the `<predicate>` position of a statement are an instance of this class. Properties can be defined in a hierarchy using the `rdfs:subPropertyOf` property. All resources which are in relation to each other with a sub property will be in relation with each other with the super property too. Domain and range restrictions can be defined by the `rdfs:domain` and `rdfs:range` properties which relate a property with a class.

In addition to these terms, RDFS provides some properties for schema documentation (`rdfs:label`, `rdfs:comment`), linking (`rdfs:seeAlso`, `rdfs:isDefinedBy`) and container description (`rdfs:member`, `rdfs:Container`, `rdf:ContainerMembershipProperty`).

WEB ONTOLOGY LANGUAGE (OWL) is, compared to RDFS, a huge step forward to provide a rich language for defining ontologies. While RDFS provides 16 terms, OWL (Version 2) provides 77 terms. To name and describe all these terms is far beyond the scope of this section. The following overview introduces just some of the most important language elements:

- OWL provides terms for ontology management and versioning (prior version, version info, deprecated terms, ...).
- OWL provides terms to describe complex classes (union, complement, ...) and cardinality restrictions on properties (min, max, ...).

15 `rdfs:Resource` represents all RDF resources (everything), `rdfs:Literal` represents the class of all literal, `rdfs:Datatype` represents the class of all datatypes

16 A best practice for publishing RDF vocabularies is to create local names in camel case as well as start property local names with lower case and class local names with upper case. Other design patterns for modeling, consuming and publishing vocabularies can be found in Dodds and Davis [2014].

- OWL provides terms to annotate properties with a logic feature (functional, transitive, ...).

This brief selection of terms is just to show the expressiveness of OWL. For a complete overview of OWL features, see Hitzler et al. [2012].

2.1.3 *Linked Data*

The term *Linked Data* refers to a method for publishing and consuming structured and linkable data on the web in order to build something more powerful based on small parts. The term was coined and defined by Berners-Lee [2006] in a way that he required four rules which should the web of data allow for growing:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information using the standards (RDF*, SPARQL).
- Include links to other URIs, so that they can discover more things.

These simple rules were clarified by Bizer et al. [2007] to form some kind of protocol which allows for publishing and consuming RDF data based on HTTP. The main problem which had to be discussed was the issue on how this protocol should distinguish between information resources (web documents such as HTML files, JPEG images, ... everything which is on the web and can be retrieved) and non-information resources (concept, persons, ... everything which is not on the web).

The main reason to distinguish between these types of resources is based on the requirement to allow for annotation of both types of resources esp. in cases where one resource represents a concept (non-information resource) and another resource represents a document which is *about* this concept. To make this clearer: the time of creation of the Berlin Wall is not the same as the time of creation of the Wikipedia page about the Berlin Wall (but both data items are important).

There exist two solutions to solve this problem, both of which are heavily in use:

- Use an IRI with a fragment identifier to represent the non-information resource so that the IRI without that fragment identifier is automatically the corresponding information resource. This solution is often used for small RDF documents such as vocabularies.
- Use an HTTP re-direction with a status code 303 to provide a re-direction from the non-information resource to the information

resource. In combination with the HTTP `Access` header field, this solution provides better support for different representations of the same data (e.g. Turtle and RDF/XML). Another advantage of the HTTP `Access` header is the capability of publishing RDF data in combination with classic HTML pages which describe a concept for human consumption. Due to a more complicated deployment process, this solution is mostly in use with support of Linked Data enabled tools such as the data wiki OntoWiki (Chapter 9, Auer et al. [2006]) or the RDF triple store Virtuoso [Erling and Mikhailov, 2007].

2.1.4 *SPARQL Query Language*

The query language SPARQL [Harris and Seaborne, 2013] is a further key technology and important building block for the Semantic Web. SPARQL 1.1 defines a query and update language together with result syntax specifications in XML and JSON and a HTTP protocol. With SPARQL, an application can retrieve and manipulate RDF data, graphs and datasets managed by a SPARQL endpoint. SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, optional patterns as well as add-ons such as filter, order, slices and other query features.

The basic query feature of SPARQL is a triple pattern which consists of a triple with potential variables at the `<subject>`, `<predicate>` and `<object>` position.

```
1 SELECT DISTINCT ?person ?bday
2 WHERE {
3   ?person a foaf:Person.
4   ?person foaf:birthday ?bday.
5   FILTER (xsd:string(?bday) >= xsd:string("01-29"))
6 }
7 ORDER BY ASC(?bday)
8 LIMIT 5
```

Listing 1: Example SPARQL query to demonstrate the language features

Listing 1 demonstrates multiple query features of SPARQL such as projection (line 1), triple patterns (line 3 an 4), a `FILTER` clause (line 5) as well as clauses for ordering and limiting the output (line 7 and 8). The English language explanation of this query is something like: *Give me the next 5 persons whose birthday is on January 20 or after and sort them by birthday*.

2.2 SOCIAL NETWORKS

2.2.1 *Types of Social Network based websites*

While Facebook, Youtube and Twitter might be the first social media services that come to mind, these examples do not represent the full scope of social networks that exist on the Web. According to Kaplan and Haenlein [2010] six different types of social media exist:

COLLABORATIVE PROJECTS such as *Wikipedia*[17] and *OpenStreetMaps*[18]. In this projects, users collaborate in a certain way on a resource and connect to each other in order to organize this collaboration. Users are typical interested in changes to resources or changes from specific users.

Figure 6: Microblogging service Twitter: A famous tweet from the Oscar award show 2014

BLOGS AND MICROBLOGS such as *Twitter*[19] and *WordPress.com*[20]. In this projects, the communication between users and the publication of a users posts is the most important feature. Beside the reception of their own posts by the community (e.g. via re-tweets, see Figure 6) users are interested in new posts from other users or regarding specific topics.

CONTENT COMMUNITIES such as *Youtube*[21] (Videos) and *Flickr*[22] (Images). Organizing and sharing their content is the most important

17 http://wikipedia.org
18 http://openstreetmap.org
19 http://twitter.com
20 http://wordpress.com
21 http://youtube.com
22 http://flickr.com

feature here. Beside that, the communication about content is important as well (bookmarks, bookmarks, likes, see Figure 7).

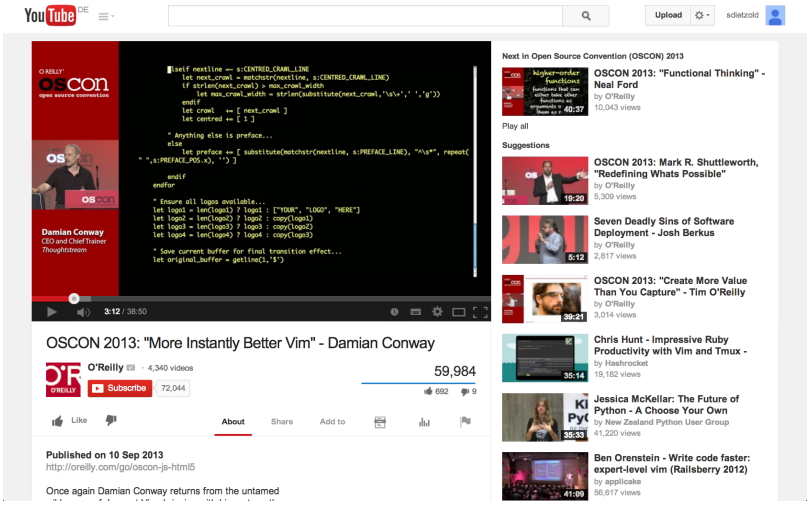

Figure 7: Content sharing site Youtube: likes, dislikes, follower count as well as visitor count support users to interpret the quality and reputation of a video

SOCIAL NETWORKING SITES such as *Facebook*[23] and *Google+*[24]. These services enable their users connect to each other by creating personal information profiles as well as inviting friends and colleagues to have access to those profiles. In addition to that, sending emails and instant messages between each other is an important feature. Social networking sites exist in many flavours and sub types regarding their target community or there main connection relation (friends, co-worker, fellow student).

VIRTUAL GAME WORLDS such as *World of Warcraft*[25] and *Eve Online*[26]. In addition to playing with or against each other, virtual game worlds support their players in finding new opponents or allies. The game world is most often complemented by a social network site where users can communicate outside the game as well as inspect game statistics and news.

VIRTUAL SOCIAL WORLDS such as *Second Life*[27] and *Habbo Hotel*[28]. Similar to game worlds, virtual social worlds enable users to act in a virtual world with an customizable avatar. In difference to game worlds, there is no primary game target to achieve. Instead of that, virtual social worlds enable their users to create virtual

23 http://facebook.com
24 http://plus.google.com
25 http://www.warcraft.com
26 http://eveonline.com
27 http://secondlife.com
28 http://habbo.com

objects and interact with them (e.g. to equip a users virtual apartment in the Habbo Hotel with virtual furniture).

White [2014] complements this classification with seven major social network categories. This classification overlaps partially with Kaplan and Haenlein [2010] and is more strictly social network site based:

SOCIAL CONNECTION networks such as *Facebook* and *Google+*. Users want to keep in touch with friends and family members. The shared content is more or less private and the social network interface is centered around activity feeds and status messages.

MULTIMEDIA SHARING networks such as *Vimeo*[29] and *Picasa*[30]. Users primarily want to share content on the Web, either public or restricted to some or all contacts in the social network. The social network interface is centered around the content (comments, likes, activity feeds).

PROFESSIONAL networks such as *LinkedIn*[31] and *Xing*[32]. These social networks are primarily used for recruiting and finding business partners. Profile descriptions emphasizes the business related points in the curriculum vitae of a user in order to allow for recruiters to better match with their requirements. In addition to that, business related groups with interests in specific topics can be started.

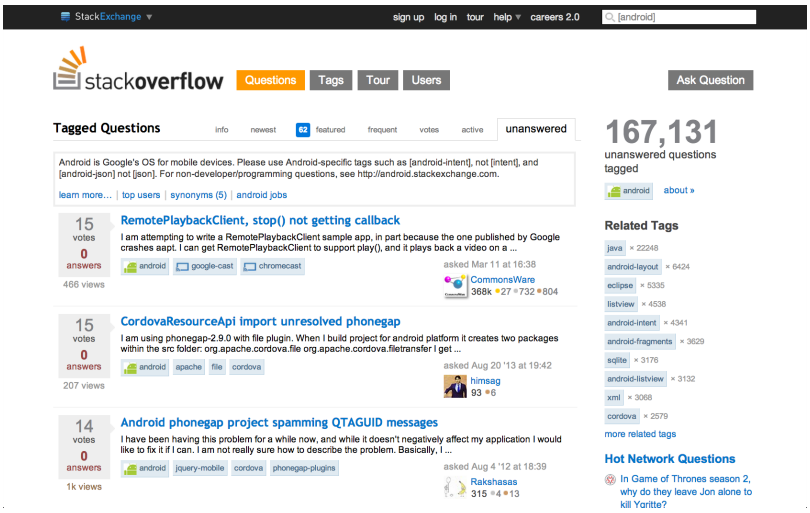

Figure 8: Question / Answer network StackOverflow: Users can up-vote unanswered questions in order to place them more prominent in the list

29 http://vimeo.com
30 http://picasa.google.com
31 http://linkedin.com
32 http://xing.com

INFORMATIONAL networks such as *StackOverflow*[33] and the *Do-It-Yourself Chatroom*[34]. These sites represent informational communities where users seeking answers to common questions on a certain topic. Informational social networks often use a forum-based site concept. A special sub category of informational social networks are question-answering platforms such as StackOverflow where users create and answer questions from other users as well as vote for and against questions or answers (see Figure 8). These sites have a strict page model including categories and tags for questions in order to maximize the value of the user-generated content. In addition to that, users gain reputation points by answering questions or other activities.

EDUCATIONAL social networks such as *The Student Room*[35] and *The Math Forum*[36]. These social networks have mostly student users which seek for other students in order to collaborate in projects.

HOBBIES based social networks such as *YardShare*[37] and *DigTheDirt*[38] esp. for gardening enthusiasts. These social networks connect people which are interested in a specific topic or hobby as well as provide specific content for this group.

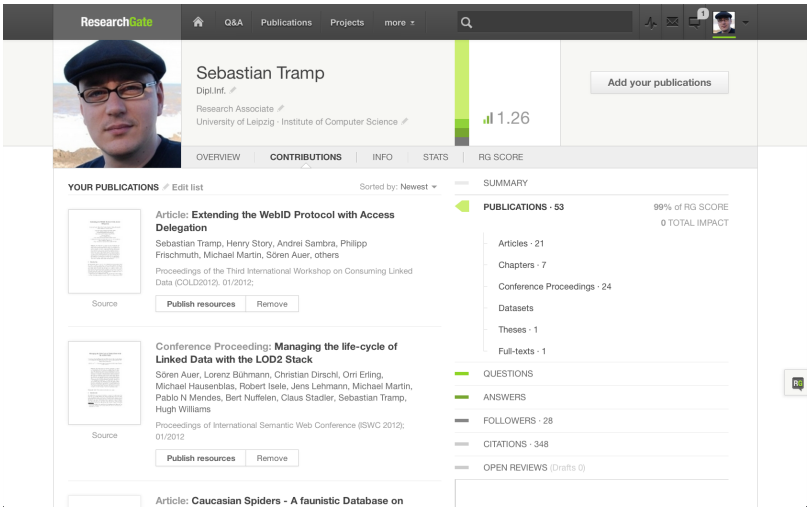

Figure 9: Academic social network ResearchGate: A user profile with detailed publication description

33 http://stackoverflow.com
34 http://www.diychatroom.com
35 http://www.thestudentroom.co.uk
36 http://mathforum.org
37 http://www.digthedirt.com
38 http://www.digthedirt.com

ACADEMIC social networks such as *Academia.edu*[39] and *ResearchGate*[40]. Users of these social networks are interested in possible co-authors and project partners. In order to achieve this, academic network profiles include publications and research topics (see Figure 9).

2.2.2 *Federated and Distributed Social Networks*

Most Social Web applications operate as silos of information. This model poses some drawbacks such as the lack of interoperability between applications, having a one-single ownership model, the inability of fully exporting data as well as the opacity in using or transmitting private data. These challenges are the reasons for addressing a distributed model. Various architectures for achieving a federated social network have been proposed and numerous projects based on those architectures have been developed to provide convenient functionality to the users. With the advent of the Semantic Web, research in this field was adapted to taking advantage of machine-readable data and ontologies. We can roughly divide the related work into distributed social networks on the Web 2.0 and distributed social networks on the Semantic Web.

2.2.2.1 *Web 2.0 Approaches*

The distributed social network model has emerged to overcome shortcomings attributed to centralized models. The foundation of the distributed model lies on a set of standards and technologies. This set of standards and protocols, which together are referred to as *Open Stack*, contains *RSS*[41], *PubSubHubbub*[42], *Pingback*[43] *Webfinger*[44], *ActivityStreams*[45], *Salmon*[46], *OAuth authorization* (Hammer [2010] and Hardt [2012]), *OpenID*[47], *XMPP* [Saint-Andre, 2004], *OpenSocial* [OpenSocial Spec], *OStatus* [Prodromou et al., 2010] and *DSNP* [Thurston, 2011].

39 http://academia.edu
40 https://www.researchgate.net
41 http://www.rssboard.org/rss-specification
42 https://code.google.com/p/pubsubhubbub/
43 Pingback [Langridge and Hickson, 2002] is one of three approaches which allow the automated generation of backlinks on the Social Web. Pingback supports the propagation of untyped links only.
44 Webfinger, https://code.google.com/p/webfinger/, enables users to make email addresses valuable by adding meta data.
45 http://activitystrea.ms/
46 http://www.salmon-protocol.org/
47 http://openid.net/

Some of the projects which were developed using these technologies include: StatusNet[48], DiSo[49], GNU Social[50], The Mine Project[51], OneSocialWeb[52], BuddyPress[53], and Diaspora[54]. These projects differ in the employed protocols and federation policy. The focus in projects such as *DiSo*, *The Mine Project*, *Appleseed* and *BuddyPress* is on equipping people with tools and functionalities. This approach allows users to build their own networks which enable them to manage and share data and relations. For instance, the *DiSo* project releases WordPress plugins which build up on OpenID, microformats and OAuth. It creates open and interoperable building blocks for launching decentralized social networks. In contrast, another strategy is to bridge between social networks to make a joint network. In a nut shell, *StatusNet* is a microblogging platform which extends OStatus for providing federated status updates; *GNU Social* is designed above *StatusNet* as a decentralized social network; *OneSocialWeb* aims at connecting social networks by employing XMPP [Saint-Andre, 2004] for instant messaging.

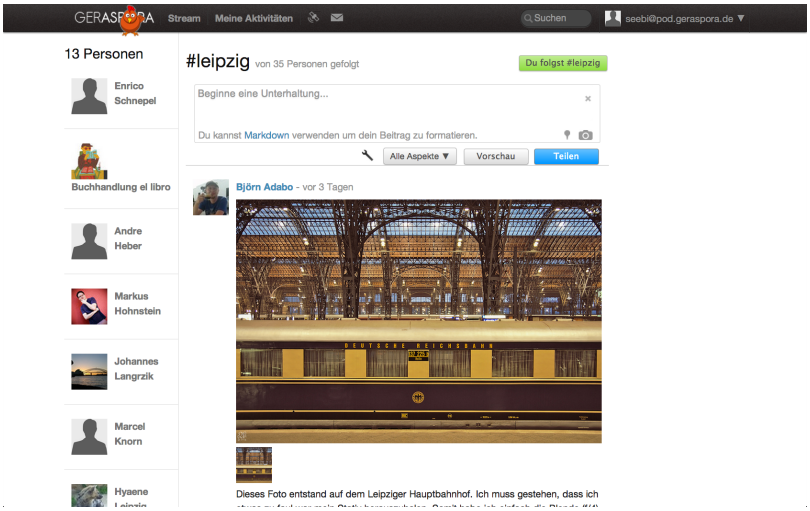

Figure 10: Diaspora*: Activity stream of the hashtag `#Leipzig`

Diaspora is a distributed social network which gained much awareness thanks to an article in the New York Times[55]. It uses Activity Streams / PubSubHubbub for public federation of posts and Webfinger to discover remote users (identified by addresses such as `user@server`). To communicate with remote users, Diaspora instances (so called *pods*) send Salmon messages which are signed and

48 http://status.net/, now http://pump.io
49 http://diso-project.org/
50 http://gnu.io, now a continuation of the StatusNet project
51 http://themineproject.org/, see also Lukas [2008]
52 http://onesocialweb.org/
53 http://buddypress.org/
54 https://www.joindiaspora.com/, see also Bielenberg et al. [2012]
55 http://www.nytimes.com/2010/05/12/nyregion/12about.html

encrypted. Additionally OStatus 2 and OAuth are used. Diaspora features privacy enhanced Status messages, blogging and photo sharing (see Figure 10).

Centralized models benefit from short development time, no required routing, central control and storage as well as data mining capabilities. In addition to these features, using distributed models is solely subject to disadvantages e.g. reachability of interlinked data or the maintenance of many peers. While a federated model as a hybrid model improves some of those disadvantages (e.g. reachability maybe higher, since the number of peers may be smaller), some of them still remain (e.g. full control over your owned data). There are many different views for tackling distributed social network challenges on the way to a federation in the Web 2.0. A well-known view is the *network of networks* which employs existing protocols and standards for providing foundations on the basis of which networks can easily communicate with one another.

Another view is using mashups. Mashups are Web applications that combine data from more than one service provider to create new services. An example is the buddycloud project[56], running an inbox server as a centralized manager over federated social networks. This server aggregates all the posts and updates from social networks to which a user has subscribed. Here again, the XMPP protocol [Saint-Andre, 2004] is used for messaging and federation.

A user-centric architecture, used in Danube [Sabadello, 2011], relies on individuals for maintaining personal data and relations. It allows them to manage their relationships with each other and with vendors. A similar view has been proposed by PrPl [Seong et al., 2010] which builds on a person-centric, social networking infrastructure, where a person's data is logically collected in one place, and social networking applications can be executed in a distributed manner without a central service.

2.2.2.2 *Semantic Web Approaches*

Primarily, Semantic Web based attempts have concentrated on the conformation of traditional technologies such as *microblogging*, *instant messaging* or *pingback* for Semantic Web. Thus, these adapted technologies can form the basis for a new generation of social networks. SMOB is a semantic and distributed microblogging framework introduced by Passant et al. [2010]. It presents some main requirements for using microblogging at a large scale, i.e. machine-readable meta data, a decentralized architecture, open data as well as re-usability and interlinking of data. These challenges have been addressed in SMOB by using RDF(a)/OWL data, distributed Hubs for exchanging information and a sync protocol (based on SPARQL/Update over HTTP)

56 http://buddycloud.com/

and interlinking components. sparqlPuSH [Passant and Mendes, 2010] utilizes the PubSubHubbub protocol to broadcast RDF query result updates. In this project, a SPARQL query which is associated with the agent is at first created and monitored in an RDF triple store. The registered user is notified whenever the result set changes. The PSI BackLinking Service for the Web of Data[57] supports the manual creation of backlinks on the Data Web by employing a number of large-scale knowledge bases, as for example, data of the UK Public Sector Information domain. Since it is based on crawling a fixed set of knowledge bases, it cannot be applied for the entire Data Web. Another service that amongst others is integrated with the PSI BackLinking Service is SameAs.org[58] [Glaser et al., 2009]. This service crawls the Web of Data in order to determine URIs describing the same resources. OKKAM [Bouquet et al., 2008] is a system that aims at unifying resource identifiers by employing meta data about resources in order to match them on entities. The approaches above support interlinking of resources and thus building Social Networks by employing centralized hubs, but do not support decentralized, on-the-fly backlinking, since they are based on crawling the Data Web on a regular basis.

Two important prerequisites for establishing a distributed social network on the Semantic Web is firstly to transform social network data into RDF and secondly, to aggregate the exported datasets by linking between person instances in different datasets. For the former prerequisite, an appropriate ontology is essential for representing social data. *FOAF* (Friend of a Friend) [Brickley and Miller, 2004], which specifies how to describe personal information and relationships with other people in a social network, is well-suited for this purpose. The *SIOC* project [Breslin et al., 2006] also extends FOAF in order to describe rich social data. The work presented in Bojars et al. [2008] uses *SIOC* for the representation of blog and bookmark content. For the second prerequisite, a graph matching model is needed to provide linkages. The work which has been carried out by Rowe [2009] describes a method for interlinking user profiles from different social networks such as *Facebook*, *MySpace* and *Twitter*. The core idea is to generate RDF graphs which can then be interlinked based on the corresponding user identifiers in each graph. Beyond these activities, an infrastructure is necessary to which these technologies can be employed.

57 http://backlinks.psi.enakting.org
58 http://sameas.org

3

STRUCTURE AND CONTRIBUTIONS OF THIS THESIS

In this Chapter, we outline the thesis structure and emphasize specific contributions made to the research field.

STRUCTURE

This thesis consists of four parts which we describe in the following subsections.

Part i – Introduction and Preliminaries

In Chapter 1 we outline major problems of centralized Social Networks. These issues were the main driving force behind our research and affect all requirements which were constructed for all parts of the thesis. We then lead over to the main reasearch questions which we want to answer in this thesis.

In Chapter 2 we give an overiew on the Semantic Web and its different technologies. In addition to that we list and evaluate the current state of the art regarding distributed and federated Social Web applications and frameworks.

Finally, in Chapter 3 we outline the thesis structure and emphasize specific contributions made to the research field.

Part ii – Architecture and Protocols

In this part of the thesis we describe the main technological ingredients for a DSSN as well as their interplay. The semantic representation of personal information is facilitated by a *WebID profile*. The WebID protocol allows for using a WebID profile for authentication and access control purposes. *Semantic Pingback* facilitates the first contact between users of the social network and provides a method for communication about resources (such as images, status messages, comments, activities) on the social network. Finally, *PubSubHubbub*-based subscription services allow for obtaining near-instant notifications of specific information as WebID profile change sets and activity streams from people in one's social network. Together, these standards and protocols provide all necessary ingredients to realize a distributed social network having all the crucial social networking features provided by centralized ones.

After providing a big picture overview of the overall architecture in Chapter 4, we give a detailed insight into two specific protocol extensions in Chapter 5 (Semantic Pingback) and Chapter 6 (Access Delegation).

Part iii – Applications

This part is structured in three chapters each describing an application which is to an extend integrated into the proposed architecture of Part ii. xOperator (Chapter 7) is an Instant Messaging agent which allows for querying the content of the DSSN data layer by using natural language templates. The MSSW client (Chapter 8) is deeply integrated Android DSSN client which allows for Social Network contact management by using the DSSN data layer and the Semantic Pingback service. OntoWiki (Chapter 9) is a semantic data wiki massively extended to be well connected in the DSSN architecture both as a consumer and a producer of Social Network data.

All three applications use a different subset of the proposed DSSN architecture as well as provide fundamental different user interfaces and usage concepts. xOperator provides query capabilities by using a natural language chat interface to "talk" to the Social Network. The MSSW client is a backend service provider of the Android plattform and is seamless integrated to bridge the DSSN data layer and the local phones contact data. OntoWiki is a browser based client which allows for sending and receiving activities as well as managing Social Network contacts.

Part iv – Evaluation, Conclusion and Future Work

In this part of the thesis, we evaluate the proposed architecture in a different way than providing prototypes such as the application in Part iii. Instead, we simulate a DSSN based on real data from the well known Twitter service and measure triple distribution as well as query performance for different DSSN node types. Finally, we conclude this thesis and provide some directions for future work.

CONTRIBUTIONS

This thesis proposes approaches to tackle each of the aforementioned research questions. The main contributions of this thesis respective the research questions (cf. Chapter 1) are:

- A Semantic Web enabled but backward compatible Pingback protocol to tackle some pressing obstacles of the emerging Linked Data Web, namely the quality, timeliness and coherence of data,

which are prerequisites in order to provide direct end user benefits. (Q2, Q3)

- The Demonstration of its usefulness by showcasing use cases of the Semantic Pingback implementations. (Q2, Q3)
- An adoption of the publish/subscribe communication pattern in order to use this as a fast and reliable transport protocol in a Linked Data enabled Social Network infrastructure. (Q2, Q3)
- An extension of the WebID authentication protocol to tackle an important issue on how users can use and describe software agents as their trusted secretaries. This allows for personal trusted machine to machine communication between user agents and rises software agents to a level of first class citizens of the Linked Data Web. (Q2, Q3)
- An overall architecture to build Distributed Semantic Social Networks based on the previous contributions and further third party ingredients (Q1–Q4).
- A Social Network simulation framework in order to evaluate triple distribution as well as query performance of selected DSSN environments and an evaluation of the proposed architecture on the basis of the simulation framework and three different DSSN enabled application prototypes. (Q1–Q3)
- The concept and development of a DSSN user interface, which shows how a deeply and synergistic coupling of Distributed Semantic Social Networks and Instant Messaging networks can be achieved. This interface approach naturally combines the well-balanced trust and provenance characteristics of IM networks with semantic representations and query answering of the Semantic Web. (Q5, Q6)
- The concept and development of a mobile DSSN client which showcases how different (social) Semantic Web standards, technologies and best practices can be integrated into a comprehensive architecture for social networking (on mobile devices). (Q6)
- The extension of a data wiki in order to showcase the integration of existing Social Web applications into the DSSN based on the proposed architecture. This full featured user interface and personal information management also demonstrates, which interface features are needed in the distributed context of a DSSN and which query strategies are suitable to drive such a rich featured UI. (Q5–Q7)

Part II

ARCHITECTURE AND PROTOCOLS

In this part of the thesis we describe the main technological ingredients for a DSSN as well as their interplay. The semantic representation of personal information is facilitated by a *WebID profile*. The WebID protocol allows for using a WebID profile for authentication and access control purposes. *Semantic Pingback* facilitates the first contact between users of the social network and provides a method for communication about resources (such as images, status messages, comments, activities) on the social network. Finally, *PubSubHubbub*-based subscription services allow for obtaining near-instant notifications of specific information as WebID profile change sets and activity streams from people in one's social network. Together, these standards and protocols provide all necessary ingredients to realize a distributed social network having all the crucial social networking features provided by centralized ones.

After providing a big picture overview of the overall architecture in Chapter 4, we give a detailed insight into two specific protocol extensions in Chapter 5 (Semantic Pingback) and Chapter 6 (Access Delegation).

4

ARCHITECTURAL OVERVIEW

The results presented in this chapter were primarily published in Tramp et al. [2014] as well as in Tramp et al. [2011a] and Story et al. [2011].

In this chapter, we describe the DSSN reference architecture. After introducing a few design principles on which the architecture is based, we present its different layers, i.e. the data, protocol, service and application layers.

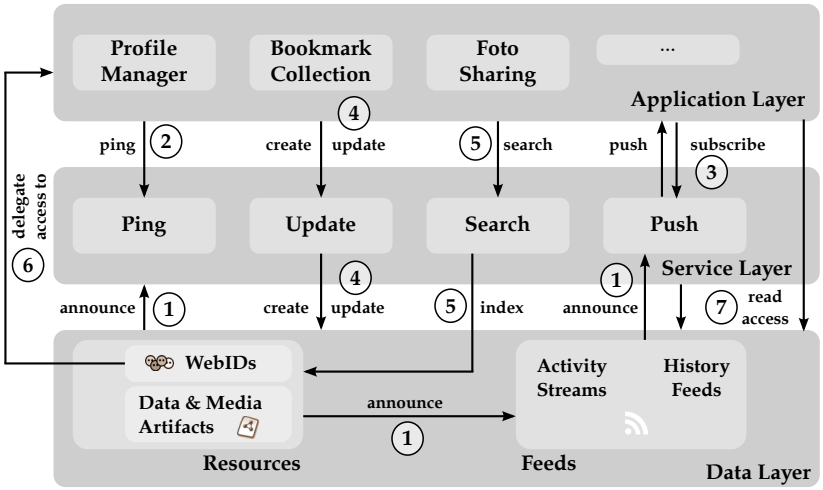

Figure 11: Architecture of a Distributed Semantic Social Network

As a first introduction, we describe the depicted box elements in Table 1 as well as the depicted arrows in the following enumeration:

(1) Resources announce services and feeds via links or header fields, feeds announce services — in particular a push service.

(2) Applications initiate ping requests to spin the Linked Data network.

(3) Applications subscribe to feeds on push services and receive instant notifications on updates.

(4) SPARQL Update services are able to modify resources (e.g. on demand of an application, cf. Section 2.1.4).

(5) Personal and global search services can index resources and are used by applications.

(6) Access to resources and services can be delegated to applications by a WebIDs, i.e. the application can act in the name of the WebID owner (cf. Access Delegation, Chapter 6).

(7) The majority of all access operations is executed through standard Web requests (cf. Linked Data, Section 2.1.3).

Table 1: Overview on DSSN concepts with description and references

Concept	Description and Reference
WebID	RDF document which describes an user or agent in the Social Network (Section 4.2.1)
Data artifact	RDF document which describes a resource such as a comment, an event or any other concept which is linked from the DSSN (Section 4.2.1)
Media artifact	Media objects such as images together with an RDF representation (Section 4.2.1)
Activity stream	A feed document which describes recent activities regarding a subject agent or object resource (Section 4.2.2)
History feed	A feed document which describe recent changes in terms of added or deleted RDF statements of a resource (Section 4.2.2)
Ping Service	A service which receives Semantic Pingback requests (Chapter 5)
Update Service	A service which receives SPARQL update requests (Section 4.4)
Push Service	A service which handles subscriptions to feeds as well as the distribution of published feed items (Section 4.4)
Search Service	A services which answers search request based on a resource index (Section 4.4)

In the next sections we introduce basic design principles as well as discuss each DSSN layer in detail.

4.1 BASIC DESIGN PRINCIPLES

Our DSSN architecture is based on the following three design principles.

Linked Data.

The main protocol for data publishing, retrieval and integration is based on the Linked Data principles Berners-Lee [2006]. All of the information contained in the DSSN is represented according to the RDF

paradigm, made dereferencable and interlinked with other resources. This principle facilitates heterogeneity as well as extensibility and enables the distribution of data and services on the Web. The resulting overall distributive character of the architecture fosters reliability and freedom of communication and leads to more data security by design.

Service Decoupling.

A second fundamental design principle is the decoupling of user data from services as well as applications Krohn et al. [2007]. It ensures that users of the network are able to choose between different services and applications. As a result, this principle enables an even more distributed character of the social network which stresses the same issues as distributed Linked Data. In addition, this principle helps users of the DSSN to distinguish between their own data, which they share with and license to other people and services, and foreign data, which they create by using these services and which they do not own. This turns the unbalanced power structure of centralized social networks upside down by strictly settling the ownership of the data to the user side and allowing access to that data in an opt-in way, which leads to more privacy.

Protocol Minimalism.

The main task for social networking protocols is to communicate RDF triples between DSSN nodes, not to enforce a specific work flow nor an exact interpretation of the data. This constraint ensures the extensibility of the data model and keeps the overall architecture clean and reliable.

4.2 DATA LAYER

The data layer comprises two main data structures:

- *resources* for the description of static entities and
- *feeds* for the representation and publishing of events and activities.

4.2.1 *Resources*

We distinguish between three main categories of DSSN resources: *WebIDs* for persons as well as applications, *data artefacts* and *media artefacts*. The properties, conditions and roles in the network of these resources are described in the next paragraphs.

WebID

WebID [Story et al., 2013][59] recently conceived in order to simplify the creation of a digital ID for end users. Since its focus lies on simplicity, the requirements for a WebID profile are minimal. In essence, a WebID profile is a de-referenceable RDF document (possibly even an RDFa-enriched HTML page) describing its owner[60]. That is, a WebID profile contains RDF triples which have the IRI identifying the owner as subject. The description of the owner can be performed in any mix of suitable vocabularies and FOAF [Brickley and Miller, 2004] as the fundamental 'industry standard' which can be extended[61]. An example WebID profile comprising some personal information (lines 8–12) and two `rel:worksWith`[62] links to co-workers (lines 6–7) is shown in Listing 2.

```
9  @prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
10 @prefix foaf:<http://xmlns.com/foaf/0.1/> .
11 @prefix rel:<http://purl.org/vocab/relationship/> .
12 <http://philipp.frischmuth24.de/id/me> a foaf:Person;
13   rdfs:comment "This is my public profile only, more information available with
         FOAF+SSL";
14   rel:worksWith <http://sebastian.tramp.name>,
15     <http://www.informatik.uni-leipzig.de/~auer/foaf.rdf#me>;
16   foaf:depiction <http://img.frischmuth24.de/people/me.jpg>;
17   foaf:firstName "Philipp"; foaf:surname "Frischmuth";
18   foaf:mbox <mailto:frischmuth@informatik.uni-leipzig.de>;
19   foaf:phone <tel:+49-341-97-32368>;
20   foaf:workInfoHomepage <http://bis.informatik.uni-leipzig.de/PhilippFrischmuth>.
```

Listing 2: A minimal WebID profile with personal information and two `rel:worksWith` relations to other WebIDs.

Apart from the main focus on representing user profiles, our architecture extends the WebID concept by facilitating two additional tasks: *service discovery* and *access delegation*.

Service discovery is used to equip a WebID with relations to trusted services which have to be used with that WebID. The usage of the WebID itself ensures that an agent can trust this service in the same way as she trusts the owner of the WebID. The most important service in our DSSN architecture is the Semantic Pingback service, which we describe in detail in Section 4.3.2.

In addition to this service, we introduce access delegation for the WebID protocol. WebID access delegation is an enhancement to the current WebID authentication process in order to allow applications

59 Formerly know as the FOAF+SSL best practice [Story et al., 2009], the latest specification is available at `http://webid.info/spec/`.

60 The usage of an IRI with a fragment identifier allows for indirect identification of an owner by reference to the (FOAF) profile document.

61 In theory, FOAF can be replaced by another vocabulary but as a grounding for semantic interoperability, we suggest to use it.

62 Taken from *RELATIONSHIP: A vocabulary for describing relationships between people* at `http://purl.org/vocab/relationship`.

to access resources and services on behalf of the WebID owner, but without the need to introduce additional application certificates in a WebID. We describe the WebID protocol as well as our access delegation extension in detail in Section 4.3.1.

Agents and Applications play an important role in today's social networks[63]. They have access to large parts of the profile data and can add or change some of the profile information, e.g. create activity descriptions or create and link images. Applications on the DSSN are also identified by using WebID profiles, but are not described as a person but as an application. They can act on behalf of a person but rely on delegated access rights for such an activity. This process is described in Section 4.3.1.

Data Artefacts

Data artefacts are resources on the Web which are published according to the Linked Data principles. Data artefacts includes posts, comments, tag assignments, activities and other Social Web artefacts which have been created by services and applications on the Web. Most of them are described by using specific Web ontologies such as SIOC [Breslin et al., 2006], Common Tag[64] or Activity Streams in RDF [Minno and Palmisano, 2010].

Media Artefacts

Media artefacts are also created by services and applications but consist of two parts — a binary data part which needs to be decoded with a specific codec or decoding application, and a meta-data part which describes this artefact[65]. Usually, such artefacts are audio, video and image files, but office document types are also frequently used on the Social Web. Media artefacts can be easily integrated into the DSSN by using the Semantic Pingback mechanism, which is described in Section 4.3.2, and a link to a push-enabled activity stream. An example photo-sharing application is described in Section 4.5.

4.2.2 *Feeds*

Feeds are used to represent temporally ordered information in a machine-readable way. Feeds are widely used on the Web and play a crucial role in combination with the *PubSubHubbub protocol* to enable near real-time communication between different services. In the context of the DSSN architecture, two types of feeds are worth considering:

63 Social network games such as FarmVille can have more than 80 million users (according to `appdata.com`), which constantly create activity descriptions.

64 `http://commontag.org/Specification`

65 Typically, the user uploads the binary part and the service creates the meta-data part based on additional form data and extracted meta-data from the binary part.

Activity Feeds

Activity feeds describe the latest social network activities of a user in terms of an `actor -- verb -- object` triple where activity verbs are used as types of activities (e.g. to post, to share or to bookmark a specific object)[66]. Activity feeds can be used to produce a merged view of the activities of one's own social network.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <feed xmlns="http://www.w3.org/2005/Atom"
3      xmlns:activity="http://activitystrea.ms/schema/1.0/">
4    <title>Activity Feed of Sebastian Tramp</title>
5    <id>http://xodx.aksw.org/?c=person&amp;id=seebi</id>
6    <link rel="hub" href="http://pubsubhubbub.appspot.com"/>
7    <link rel="self" type="application/atom+xml" href="..."/>
8    <updated>2014-04-22T18:34:45+02:00</updated>
9    <entry>
10     <title>&quot;Sebastian Tramp posted a Post&quot;</title>
11     <id>http://xodx.aksw.org/?c=activity&amp;id=aa9eb16a86a18f5a6cbb</id>
12     <link href="http://xodx.aksw.org/?c=activity&amp;id=aa9eb16a86a18f5a6cbb" />
13     <published>2014-04-22T18:34:45+02:00</published>
14     <updated>2014-04-22T18:34:45+02:00</updated>
15     <author>
16       <name>Sebastian Tramp</name>
17       <uri>http://xodx.aksw.org/?c=person&amp;id=seebi</uri>
18     </author>
19     <activity:verb>http://xmlns.notu.be/aair#Post</activity:verb>
20     <activity:object>
21       <id>http://xodx.aksw.org/?c=resource&amp;id=b8273f2a97a4c47313ee</id>
22       <content>Just test the awesome xodx!</content>
23       <published>2014-04-22T18:34:45+02:00</published>
24       <activity:object-type>http://rdfs.org/sioc/ns#Post</activity:object-type>
25     </activity:object>
26     <content>New Activity: &quot;Sebastian Tramp posted a Post&quot;: Just test
           the awesome xodx!</content>
27   </entry>
28 </feed>
```

Listing 3: Activity feed with a single example activity entry: The activity is defined from line 9–27 and the feed has an attached PubSubHubbub service (line 6).

In our DSSN architecture, each activity is created as a Linked Data resource (i.e. a DSSN data artefacts), which links to the actor and object of the activity (cf. Listing 3). In addition, each activity is equipped with a Pingback Server in order to allow for receiving reactions on this activity (called pingbacks) and thus to spin a content network between these artefacts.

History Feeds

History feeds are used to allow syndication of change sets of specific resources between a publisher of a resource and many subscribers

66 Activity streams (http://activitystrea.ms) are Atom format extensions to describe activity feeds. It is extensible in a way that allows publishers to use new verb- or object-type IRIs to identify site-specific activities.

of the resource. History feeds describe changes in RDF resources in terms of added and deleted statements which are boxed in an Atom feed entry. A subscriber's social network application can use this information to maintain an exact copy of the original resource for caching and querying purposes. History feeds are in particular important for the syndication of changes of WebID profiles (e.g. if a contact changes its phone number).

4.3 PROTOCOL LAYER

The protocol layer consists of the WebID identity protocol and two networking protocols which provide support for two complete different communication schemes, namely resource linking and push notification.

4.3.1 *WebID (protocol)*

From a more technical perspective, the WebID protocol [Story et al., 2013] incorporates authentication and trust into the WebID concept. The basic idea is to connect an *SSL client certificate* with a WebID profile in a secure manner and thus allowing owners of a WebID to authenticate against 3rd-party websites with support for the WebID protocol. The WebID (i.e. a dereferencable URI) is, therefore, embedded into an *X.509 certificate*[67] by using the Subject Alternative Name (SAN) extension. The document, which is retrieved through the URI, contains the corresponding public key. Given that information, a relying party can assert that the accessing user owns a certain WebID. Furthermore, the WebID protocol can provide access control functionality for social networks shaped by WebIDs in order to regulate access to certain information resources for different groups of contacts (e.g. as presented with *dgFOAF* in Schwagereit et al. [2010]). An example of a WebID profile, which is annotated with a public key, is shown in Listing 4. This WebID profile contains additionally a description of an *RSA public key* (line 15), which is associated to the WebID by using the `cert:identity` property from the *W3C certificates and crypto ontology* (line 19).

Nevertheless, the described approach requires the user to access a secured resource directly, e.g. through a Web browser which is equipped with a WebID-enabled certificate. However, in the scenario of a distributed social network this arrangement is not always the case. If, for example, the software of user A needs to update its local cache of user B's profile, it will do so by fetching the data in the background and not necessarily when user A is connected. An obvious solution would be to hand out a WebID-enabled certificate to the software (agent), but then the user needs to create a dedicated certificate for

67 http://www.ietf.org/rfc/rfc2459.txt

```
@prefix rsa:  <http://www.w3.org/ns/auth/rsa#>.
@prefix cert: <http://www.w3.org/ns/auth/cert#">.
[] a rsa:RSAPublicKey;
  rdfs:comment "used from my smartphone ...";
  cert:identity <http://philipp.frischmuth24.de/id/me>;
  rsa:modulusc "C41199E ... 5AB5"^^cert:hex;
  rsa:public_exponent "65537"^^cert:int.
```

Listing 4: An extension of the minimal WebID from Listing 2: Description of an RSA public key, which is associated to the WebID by using the `cert:identity` property from the W3C certificates and crypto ontology.

all tools that have to access secured information and simultaneously allows all participating tools to "steal" her identity, which is not the preferred solution from a security perspective.

To resolve this dilemma, we have extended the WebID protocol by adding support for *access delegation*. By delegating access to an agent, a user allows a particular agent to deputy access-secured information resources. The agent itself authenticates against the relying party by using its own credentials, e.g. by employing the WebID protocol, too. Additionally, it sends a informational HTTP header, which indicates that a resource is accessed on behalf of a certain WebID user.

The access delegation WebID protocol extension is discussed in detail in Chapter 6.

4.3.2 *Semantic Pingback*

The purpose of *Semantic Pingback* (in detail presented in Chapter 5 and Tramp et al. [2010a]) in the context of a DSSN architecture is twofold:

- It is used to facilitate the first contact between two WebIDs and establish a new connection (*friending*).
- It is used to ping the owner of different social network artefacts if there are activities related to these artefacts (e.g. commenting on a blog post, tagging an image, sharing a website from the owner).

The Semantic Pingback approach is based on an extension of the well-known Pingback technology [Langridge and Hickson, 2002], which is one of the technological cornerstones of the overwhelming success of the blogosphere in the Social Web. It enables bi-directional links between WebIDs and RDF resources as well as weblogs and websites in general. It facilitates contact / author / user notifications in case a link has been newly established. It is based on the advertisement of a lightweight RPC service[68] in the RDF document, HTTP

68 In fact, we experimented with different service endpoints. Based on the results, which are described in more detail in Story et al. [2011], we now prefer simple HTTP post requests which are not compatible with standard XML-RPC pingbacks.

or HTML header of a certain Web resource, which should be called as soon as a (typed RDF) link to that resource is established. The Semantic Pingback mechanism allows casual users and authors of RDF content, of weblog entries or of an article in general to obtain immediate feedback when other people establish a reference to them or their work, thus *facilitating social interactions*. It also allows to publish backlinks automatically from the original WebID profile (or other content, e.g. status messages) to comments or references of the WebID (or other content) elsewhere on the Web, thus *facilitating timeliness and coherence* of the Social Web.

```
36 @prefix ping: <http://purl.org/net/pingback/>.
37 <http://philipp.frischmuth24.de/id/me> ping:to <http://pingback.aksw.org>.
```

Listing 5: Extension of the minimal WebID profile from Listing 2: Assignment of an external Semantic Pingback service which can be used to ping this specific resource.

As a result, the distributed network of WebID profiles, RDF resources and social websites can be much more tightly and timelier interlinked by using the Semantic Pingback mechanism than conventional websites, thus rendering a network effect, which is one of the major success factors of the Social Web. Semantic Pingback is completely downwards compatible with the conventional Pingback implementations, thus allowing the seamless connection and interlinking of resources on the Social Web with resources on the DSSN. An extension of our example profile with Semantic Pingback functionality making use of an external Semantic Pingback service is shown in Listing 5. In line 23, the subject resource is linked with the `ping:to` relation to the Semantic Pingback service.

As requested by our third DSSN design paradigm (protocol minimalism), Semantic Pingback is a generic data networking protocol which allows to spin relations between any two Social Web resources. In the context of the DSSN Architecture, Semantic Pingback is used in particular for friending, commenting and tagging activities.

FRIENDING is the process of establishing a symmetric `foaf:knows` relation between two WebIDs. A relationship is approved when both persons publish this relation in their WebIDs. A typical friending work flow can be described by the following steps:

- Alice publishes a `foaf:knows` relation to Bob in her WebID profile.
- Alice's WebID hosting service pings Bob's WebID to inform Bob about this new statement.
- Bob receives a message from his Pingback Service.

- Bob can approve this relation by publishing it in his WebID profile, which sends again a ping back to Alice[69].

This basic model of communication can be applied to different events and activities in the social network. Table 2 lists some of the more important pingback eventsIn any case, the owner of these resources can be informed about the event and in most cases specific actions should be triggered. However, a specific reaction is not enforced by the protocol.

Table 2: Typical RDF statements which can cause ping activities.

source resource	object property	target resource
foaf:Person	foaf:knows	foaf:Person
		(friending)
sioct:Comment	sioc:about	foaf:Image
	(commenting an image or any other resource)	
sioc:Post	sioc:reply_of	sioc:Post
		(replying to a post)
tag:Tagging	tag:taggedResource	*
	(any resource which is tagged by a user)	
aair:Activity	aair:activityObject	*
	(any resource is object of an activity)	

4.3.3 *PubSubHubbub*

PubSubHubbub[70] is a web-hook-based publish/subscribe protocol, as an extension to Atom and RSS, which allows for near instance distribution of feed entries from one publisher to many subscribers. Since feed entries are not described as RDF resources, PubSubHubbub is not the best solution as a transport protocol for a DSSN from a Linked Data perspective. However, PubSubHubbub with atom feeds is widely in use and has good support in the Web developer community which is why we decided to use it in our architecture. Similar to Semantic Pingback, it is agnostic to its payload and can be used for all publish/subscribe communication connections.

Figure 12 depicts the workflow of publication, subscription and notification of the protocol:

1. The feed publisher creates or updates a feed.

69 Please note that the Semantic Pingback protocol does not enforce any specific reaction on a certain relation type or any reaction at all.

70 http://code.google.com/p/pubsubhubbub/

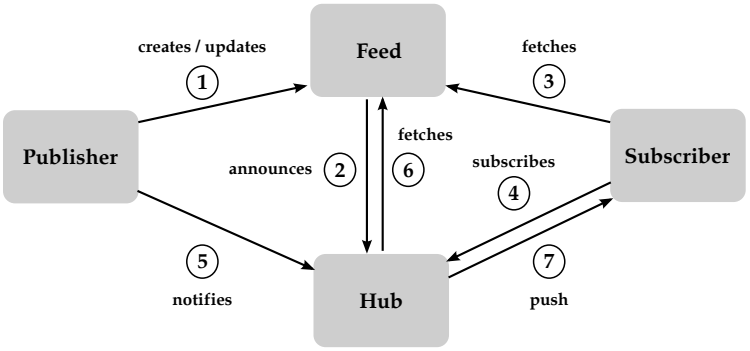

Figure 12: Publication / Subscription workflow of PubSubHubbub

2. This feed includes a link the responsible subscription hub in the head of the feed.
3. The possible subscriber fetches the feed for the first time and discovers the subscription hub.
4. The subscriber subscribes to the feed on the subscription hub.
5. The publisher notifies the hub regarding new content in the feed.
6. The hub fetches the content in order to distribute it.
7. The hub pushes new content to the subscribers.

The main advantage of this communication model is to avoid frequent and unnecessary pulls of all interested subscribers from this feed and to allow a faster broadcast to the subscriber.

In the DSSN architecture, two specific feeds are important and interlinked with a WebID to allow for subscriptions:

- *activity feeds* which are used for activity distribution and
- *history feeds* which are used for resource synchronization.

Activity Description Distribution

The distribution of activity descriptions is a fundamental communication channel for any social network. A personal activity feed publishes the stream of all activities on social network resources (artefacts and WebIDs) with a specific user as the actor. These activity descriptions can and should be created by any application which is allowed to update the feed (ref. access delegation Chapter 6). In addition, activity feeds should be created for data and media artefacts in order to allow object-centered push notification. Typically, activity feed updates are pushed to a personal search / index service of a subscribed user (see next section).

Resource Synchronization

The synchronization of resource descriptions between publisher and consumers is an additional communication scheme based on feeds. It is required, especially in distributed social networks, to take into account that relevant data is highly distributed over many locations and that access to and querying of this data can be very time-consuming without caching. A properly connected resource synchronization tackles this problem by allowing users to subscribe to changes of certain resources over PubSubHubbub. For a WebID, this process can be included into the *friending* process, while for other resources a user can subscribe manually (e.g. if a user is member of a certain group, then she may subscribe to the feed of a group resource to receive updates). Resource synchronization is a well-known topic when dealing with distributed resources. We have designed our data model as Linked Data update logs [Auer et al., 2009] based on the work previously published by the Triplify project[71].

4.4 SERVICE LAYER

Services are applications which are part of the DSSN infrastructure (in contrast to applications from the application layer). WebIDs can be equipped with different services in order to allow manipulation and other actions on the user's data by other applications. As depicted in Figure 11, we have defined four essential services for a DSSN.

Ping Service

The ping service provides an endpoint for any incoming pingback request for the resources of a user. First and foremost, it is used with the WebID for friending but also for comment notification and discussions.

One single ping service application instance can provide its services for multiple resources. In a minimal setup, a ping service provides only a notification service via email. In a more complex setup, the ping service has access to the update service of a user (via access delegation) and can do more than sending notification.

A ping service can be announced with the `ping:to` relation as shown in Listing 5.

Push Service

The push service is used for activity distribution and resource synchronization. Both introduced types of feeds announce its push service in the same way as using the `rel="hub"` link in the feed head. Since

71 http://triplify.org

both types of feeds are valid atom feeds, a DSSN push service can be a standard PubSubHubbub-based instance.

To equip social network resources with its corresponding activity and history feeds, we have defined two OWL object properties which are sub-properties of the more generic `sioc:feed` relation from the SIOC project [Breslin et al., 2006]: `dssn:activityFeed` and `dssn:syncFeed`.

In addition to these RDF properties, DSSN agents should pay attention to the corresponding HTTP header fields `X-ActivityFeed` and `X-SyncFeed`, which are alternative representations of the OWL object properties to allow the integration of media artefacts without too much effort.

Search and Index Service

Search and index services are used in two different contexts in the DSSN architecture.

- They are used to search for public Web resources, which are not yet part of a user's social network. These search services are well-known semantic search engines such as Swoogle [Ding et al., 2004] or Sindice [Tummarello et al., 2007]. They use crawlers to keep their resource cache up-to-date and provide user interfaces as well as application programming interfaces to integrate and use their services in applications.

  In our architecture, these public services are used to search for new contacts as well as other artefacts in the same way as people can use a standard Web search engine. The main advantage in using Semantic Web search engines lies in their ability to use graph patterns for a search activity[72].

- In addition to public search services, we want to emphasize the importance of private search services in our architecture. Private search services are used in order to have a fast resource cache not only for public, but also for private data which a user is allowed to access.

  A private search service is used for all users and queries from applications which act on behalf of the user. The underlying resource index of a private search service is used as a callback for all push notifications from feeds to which the user has subscribed. That is, she is able to query over the latest up-to-date data by using her private search service. In addition, she can query for data which has never been public and is published for a few people only.

72 A motivating example in our context is the search for resources of type `foaf:Person`, which are related to the DBpedia topic `dbpedia:DataPortability` (e.g. with the `foaf:interest` object property).

Since private search services are used by applications which act on behalf of the user, they must be WebID-protocol-enabled. That is, they accept requests from the user and her delegated agents only. In addition, applications need to know which private search service should be accessed on behalf of the user. This mode is again made possible by providing a link from the WebID to the search service[73]

In our architecture we assume that search services accept SPARQL queries.

Update Service

Finally, an update service provides an interface to modify and create user resources in terms of SPARQL update queries. In the same way as private search services, update services are secured by means of the WebID protocol and accept requests only by the user itself and by agents in access delegation mode[74].

Typical examples of how to use this service are the creation of activities on behalf of the user or the modification of the user's WebID, e.g. by adding a new `foaf:knows` relation.

In the next section, we give a detailed description of the service interplay and usage by applications.

4.5 APPLICATION LAYER

Social Web applications create and modify all kinds of resources for a user. In our architecture, they have to use the trusted services which are related to a WebID instead of their own. Since access to these services is exclusively delegated by the user to an application, the user has full control over her data[75]. To illustrate how DSSN applications work with a WebID and its services, we will describe a simple photo-sharing application work flow:

1. When a user creates an account on this service, she uses her WebID for the first login and delegates access to this application.

73 In our prototypes we use a simple OWL object property `dssn:searchService`, which is a sub-property of `dssn:trustedService`. We assume that such an easy vocabulary is only the first step to a fully featured service auto-discovery ontology and consider all `dssn` terms as unstable.

74 We defined `dssn:updateService` as a relation between a WebID and an update service.

75 At the moment we distinguish only between access and no access to a service. As an extension, we can imagine that a private search service can handle access on parts of the private Social Graph differently (an online game does not need to know which other activities you pursue on the Web). Access policies for RDF knowledge bases is a topic of ongoing research and we hope that the results of this research area can be adapted here.

2. The application analyses the WebID and discovers the trusted services and some meta-data of the user (e.g. name, short bio and depiction).

3. The user then uploads her first image to the application.

4. The application creates a new image resource and an activity stream for that resource.

5. After that, the application creates two activities for the user: one in the stream of the image resource and one in the personal stream of the user, employing the recently delegated access right

6. These activities are instantly pushed to all of the user's friends and are not part of the data of the image publishing service.

7. Furthermore, the photo-sharing application equips the newly uploaded image with the pingback service of the user; thus enabling the image for backlinks and comments.

8. New comments can arrive from everywhere on the Web, but the application also provides its own commenting service (integrated in the image Web view).

9. If another user writes a comment on this image, a data artefact is created in the namespace of the application and a ping request is sent to the user's pingback service (since this service is related to the image).

This simple example demonstrates the interplay and rules of the DSSN service architecture. A more complex social network application is described in the next section.

5

SEMANTIC PINGBACK

The results presented in this chapter were primarily published in Tramp et al. [2010a] and influenced other related publications of the author such as Story et al. [2011]; Tramp et al. [2011b,a, 2014].

In this Chapter, we introduce the Semantic Pingback protocol as a part of the proposed DSSN architecture.

As described briefly in Section 4.3.2, *Semantic Pingback* is used to facilitate the first contact between two WebIDs and establish a new connection (*friending*) as well as to to ping the owner of different social network artifacts if there are activities related to these artifacts (e.g. commenting on a blog post, tagging an image, sharing a website from the owner). In addition to that, the initial motivation for Semantic Pingback as a main building block for a Linked Data infrastructure goes far beyond these DSSN use-cases:

Recently, the publishing of structured, semantic information as Linked Data, not only data from Social Networks, has gained much momentum. A number of Linked Data providers meanwhile publish more than 200 interlinked datasets amounting to 13 billion facts[76]. Despite this initial success, there are a number of substantial obstacles, which hinder the large-scale deployment and use of the Linked Data Web as well as Linked Data enabled architectures such as Distributed Social Semantic Networks.

These obstacles are primarily related to the *quality, timeliness and coherence* of Linked Data. In particular for ordinary users of the Internet, Linked Data is not yet sufficiently visible and (re-) usable. Once information is published as Linked Data, authors hardly receive feedback on its use and the opportunity of realizing a network effect of mutually referring data sources is currently unused.

In this chapter we present an approach for complementing the Linked Data Web with a social dimension. The approach is based on an extension of the well-known Pingback technology [Langridge and Hickson, 2002], which is one of the technological cornerstones of the overwhelming success of the blogosphere in the Social Web. The Pingback mechanism enables bi-directional links between weblogs and websites in general as well as author/user notifications in case a link has been newly established. It is based on the advertising of a lightweight RPC service, in the HTTP or HTML header of a certain Web resource, which should be called as soon as a link to that resource is established. The Pingback mechanism enables authors of a weblog entry or article to obtain immediate feedback, when other people reference their work, thus *facilitating reactions and social interactions*. It also allows to automatically publish backlinks from the original article

76 http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics

to comments or references of the article elsewhere on the Web, thus *facilitating timeliness and coherence* of the Social Web. As a result, the distributed network of social websites using the Pingback mechanism (such as the blogosphere) is much tighter and timelier interlinked than conventional websites, thus rendering a network effect, which is one of the major success factors of the Social Web.

With this work we aim to apply this success of the Social Web to the Linked Data Web. We extend the Pingback mechanism towards a Semantic Pingback, by adding support for typed RDF links on Pingback clients, servers and in the autodiscovery process.

When an RDF link from a Semantic Pingback enabled Linked Data resource is established with another Semantic Pingback enabled Linked Data resource, the latter one can be automatically enriched either with the RDF link itself, with an RDF link using an inverse property or additional information. When the author of a publication, for example, adds bibliographic information including RDF links to co-authors of this publication to her semantic wiki, the co-authors' FOAF profiles can be enriched with backlinks to the bibliographic entry in an *automated or moderated* fashion. The Semantic Pingback supports *provenance* through tracking the lineage of information by means of a provenance vocabulary. In addition, it allows to implement a variety of measures for *preventing spam*.

Semantic Pingback is completely downwards compatible with the conventional Pingback implementations, thus allowing to seamlessly connect and interlink resources on the Social Web with resources on the Data Web. A weblog author can, for example, refer to a certain Data Web resource, while the publisher of this resource can get immediately notified and `rdfs:seeAlso` links can be automatically added to the Data Web resource.

This chapter is structured as follows:

- We describe the requirements which guided the development of Semantic Pingback in Section 5.1.
- We present an architectural overview including client and server communication behavior as well as autodiscovery algorithms of our solution in Section 5.2, Section 5.3 and Section 5.4.

5.1 REQUIREMENTS

In this section we discuss the requirements, which guided the development of our Semantic Pingback approach.

Semantic links.

The conventional Pingback mechanism propagates untyped HTML links between websites. In addition the Semantic Pingback mechanism

should be able to propagate typed links (e.g. OWL object properties) between RDF resources.

Use RDFa-enhanced content where available.

Since most traditional weblog and wiki systems are able to create semantically enriched content based on RDFa annotations[77], these systems should be able to propagate typed links derived from RDFa annotations to a Semantic Pingback server without any additional modification or manual effort.

Downward compatibility with conventional Pingback servers.

Conventional Pingback servers should be able to retrieve and accept requests from Semantic Pingback clients. Thus, widely used Social Web software such as WordPress or Serendipity can be pinged by a Linked Data resource to announce the referencing of one of their posts. A common use case for this is a Linked Data SIOC [Breslin et al., 2005] comment which replies and refers to a blog post or wiki page on the Social Web. Such a SIOC comment typically uses the `sioc:reply_of` object property to establish a link between the comment and the original post[78].

Downward compatibility for conventional Pingback clients.

Conventional Pingback clients should be able to send Pingbacks to Semantic Pingback servers. Thus, a blogger can refer to any pingback-enabled Linked Data resource in any post of her weblog. Hence, the conventional Pingback client should be able to just send conventional Pingbacks to the Linked Data server. Unlike a conventional Pingback server, the Semantic Pingback server should not create a comment with an abstract of the blog post within the Linked Data resource description. Instead an additional triple should be added to the Linked Data resource, which links to the referring blog post.

Support Pingback server autodiscovery from within RDF resources.

The conventional Pingback specification keeps the requirements on the client side at a minimum, thus supporting the announcement of a Pingback server through a `<link>`-Element in an HTML document.

77 This should be possible at least manually by using the systems HTML source editor, but can be supported by extensions as for example described in Corlosquet et al. [2009] for Drupal.

78 Since SIOC is a very generic vocabulary, people can also use more specific relations as, for instance, `swandr:disagreesWith` or `swandr:alternativeTo` from the Scientific Discourse Relationships Ontology of the SWAN project [Ciccarese et al., 2008].

Since the Semantic Pingback approach aims at applying the Pingback mechanism for the Web of Data, the autodiscovery process should be extended in order to support the announcement of a Pingback server from within RDF documents.

Provenance tracking.

In order to establish trust on the Data Web it is paramount to preserve the lineage of information. The Semantic Pingback mechanism should incorporate the provenance tracking of information, which was added to a knowledge base as result of a Pingback.

Spam prevention.

Another aspect of trust is the prevention of unsolicited proliferation of data. The Semantic Pingback mechanism should enable the integration of measures to prevent spamming of the Data Web. These measures should incorporate methods based on data content analysis and social relationship analysis.

5.2 OVERVIEW

The general architecture of the Semantic Pingback approach is depicted in Figure 13. A *linking resource* (depicted in the upper left) links to another (Data) Web resource, here called *linked resource* (arrow 1). The linking resource can be either an conventional Web resource (e.g. wiki page, blog post) or a Linked Data resource. Links originating from Linked Data resources are always typed (based on the used property), links from conventional Web resources can be either untyped (i.e. plain HTML links) or typed (e.g. by means of RDFa annotations). The *Pingback client* (lower left) is either integrated into the data/content management system or realized as a separate service, which observes changes of the Web resource (arrow 2). Once the establishing of a link was noted, the Pingback client tries to autodiscover a Pingback server from the linked resource (arrow 3). If the autodiscovery was successful, the respective Pingback RPC server is called (arrow 4), with the parameters linking resource (i.e. source) and linked resource (i.e. target). In order to verify the retrieved request (and to obtain information about the type of the link in the semantic case), the Pingback server fetches (or dereferences) the linking resource (arrow 5). Subsequently, the Pingback server can perform a number of actions (arrows 6,7), such as updating the linked resource (e.g. adding inverse links) or notifying the publisher of the linked resource (e.g. via email). This approach is compatible with the conventional Pingback specification [Langridge and Hickson, 2002], which illustrates the chain of communication steps with the help of a Alice and Bob scenario. This scenario as well

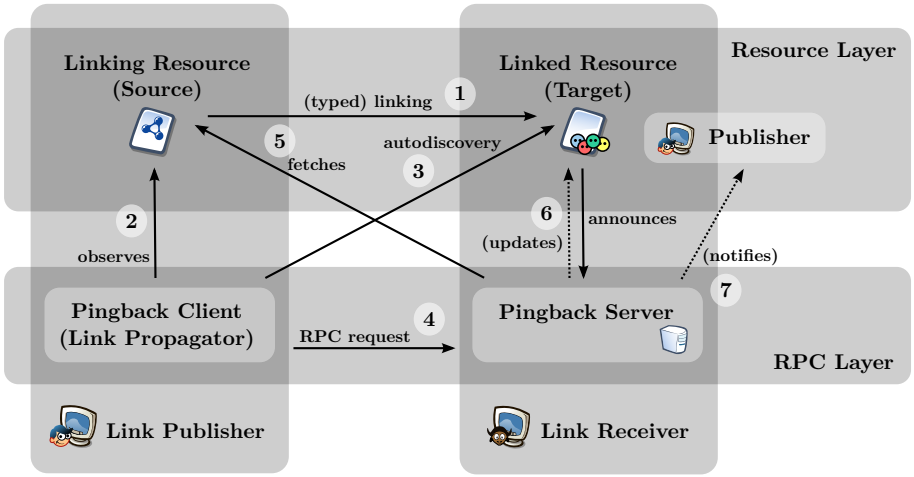

Figure 13: Architecture of the Semantic Pingback approach: (1) A *linking resource* links to another (Data) Web resource, here called *linked resource*. (2) The *Pingback client* is either integrated into the data/-content management system or realized as a separate service, which observes changes of the Web resource. (3) Once the establishing of a link has been noted, the Pingback client tries to auto-discover a Pingback server from the linked resource. (4) If the auto-discovery has been successful, the respective Pingback server is used for a ping. (5) In order to verify the retrieved request (and to obtain information about the type of the link in the semantic case), the Pingback server fetches (or de-references) the linking resource. (6 + 7) Subsequently, the Pingback server can perform a number of actions such as updating the linked resource (e.g. adding inverse links) or notifying the publisher of the linked resource (e.g. via email).

as the general architecture introduce four components, which we now describe in more detail:

Pingback client.

Alice's blogging system comprises the Pingback client. The Pingback client establishes a connection to the Pingback server on a certain event (e.g. on submitting a new blog post) and starts the Pingback request.

Pingback server.

Bob's blogging system acts as the Pingback server. The Pingback server accepts Pingback request via XML-RPC and reacts as configured by the owner. In most cases, the Pingback server saves information about the Pingback in conjunction with the target resource.

Target resource.

Bob's article is called the target resource and is identified by the *target URI*. The target resource can be either a web page or an RDF resource, which is accessible through the Linked Data mechanism. A target resource is called *pingback-enabled*, if a Pingback client is able to glean information about the target resource's Pingback server (see item 5.3 for autodiscovery of Pingback server information).

Source resource.

Alice's post is called the source resource and is identified by the *source URI*. Similar as the target resource, the source resource can be either a web page or an RDF resource. The source resource contains some relevant information chunks regarding the target resource.

These information chunks can belong to one or more of the following categories:

- An *untyped HTML link* in the body of the web page (this does not apply for Linked Data resources).
- A (possible RDFa-encoded) RDF triple linking the source URI with the target URI trough an arbitrary RDF property. That is, the extracted source resource model contains a *direct relation* between the source and the target resource. This relation can be directed either from the source to the target or in the opposite direction.
- A (possible RDFa-encoded) RDF triple where either the subject or the object of the triple is the target resource. This category represents *additional information* about the target resource including textual information (e.g. an additional description) as well as assertions about relations between the target resource and a third resource. This last category will most likely appear only in RDFa enhanced web pages since Linked Data endpoints are less likely to return triples describing foreign resources.

Depending on these categories, a Semantic Pingback server will handle the Pingback request in different ways. We describe this in more detail later in Section 5.4.

Figure 14 illustrates the complete life-cycle sequence of a (Semantic) Pingback.

- Firstly, the source publisher updates the source resource, which is observed by a Pingback client.
- The Pingback client then scans the source resource for links (typed or untyped) to other resources.

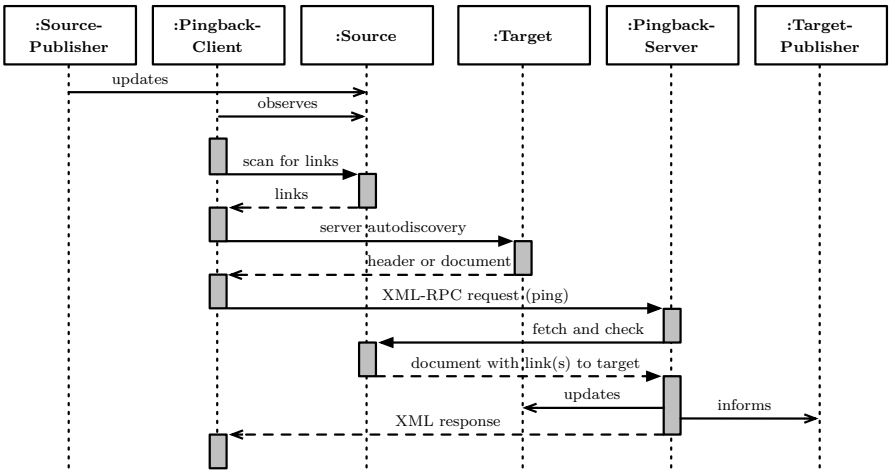

Figure 14: Sequence diagram illustrating the (Semantic) Pingback workflow.

- Each time the client detects a suitable link, it tries to determine a Pingback server by means of an autodiscovery process.
- Once a Pingback server was determined, the client pings that server via an XML-RPC request. Section 5.3 contains a more detailed description of these steps.
- Since the requested Pingback server only receives the source and target URIs as input, it tries to gather additional information. At least the source document is fetched and (possibly typed) links are extracted.
- Furthermore the target resource is updated and the publisher of the target resource is notified about the changes. In Section 5.4 the server behavior is described in more detail.
- Finally, the Pingback server responds with an XML result.

5.3 CLIENT BEHAVIOR

One basic design principle of the original Pingback specification is to keep the implementation requirements of a Pingback client as simple as possible. Consequently, Pingback clients do not even need an XML/HTML parser for basic functionality. There are three simple actions to be followed by a Pingback client:

1. Determine suitable links to external target resources,
2. detect the Pingback server for a certain target resource and
3. send an XML-RPC post request via HTTP to that server.

Conventional Pingback clients would naturally detect (untyped) links by scanning HTML documents for `<a>`-elements and use the `href`-attribute to determine the target. Semantic Pingback clients will furthermore derive suitable links by examining RDFa annotated HTML or RDF documents. Both conventional and Semantic Pingback clients are able to communicate with a Semantic Pingback server, since the Semantic Pingback uses exactly the same communication interface. In particular, we did not change the remote procedure call, but we introduce a third possible autodiscovery mechanism for Semantic Pingback clients in order to allow the propagation of server information from within RDF documents. On the one hand, this enables the publisher of a resource to name a Pingback server, even if the HTTP header cannot be modified. On the other hand, this allows caching and indexing of Pingback server information in a Semantic Web application. Since a large number of Semantic Web applications store the data retrieved from other parties, they can take advantage of the embedded Pingback server information without requesting the data again, thus accelerating the discovery process.

Server autodiscovery

The server autodiscovery is a protocol followed by a Pingback client to determine the Pingback server of a given target resource. The Pingback mechanism supports two different autodiscovery mechanisms which can be used by the Pingback client:

- an HTTP header attribute `X-Pingback` and
- a `link`-element in the HTML head with a relation attribute `rel="pingback"`.

Both mechanisms interpret the respective attribute value as URL of a Pingback XML-RPC service, thus enabling the Pingback client to start the request.

The `X-Pingback` HTTP header is the preferred autodiscovery mechanism and all Semantic Pingback server must implement it in order to achieve the required downward compatibility. We define an additional autodiscovery method for Linked Data resources which is based on RDF and integrates better with Semantic Web technologies.

Therefore, we define an OWL object property `ping:service`[79], which is part of the Pingback namespace and links a RDF resource with a Pingback XML-RPC server URL. The advantage compared to an HTTP header attribute is that this information can be stored along with a cached RDF resource description in an RDF knowledge base. Another benefit is, that different resources identified by hash URIs can be linked with different Pingback servers. However, a disadvantage (as for the

79 http://purl.org/net/pingback/service

HTML link element too) is that Pingback clients need to retrieve and parse the document instead of requesting the HTTP header only.

5.4 SERVER BEHAVIOR

While the communication behavior of the server is completely compatible with the conventional Pingback mechanism (as described in Langridge and Hickson [2002]), the manipulation of the target resource and other request handling functionality (e.g. sending email notifications) is implementation and configuration dependent. Consequently, in this section we focus on describing guidelines for the important server side manipulation and request handling issues spam prevention, backlinking and provenance tracking.

5.4.1 *Spam Prevention*

At some point every popular service on the Internet, be it email, weblogs, wikis, newsgroups or instant messaging, had to face increasing abuse of their communication service by sending unsolicited bulk messages indiscriminately. Each service dealt with the problem by implementing technical as well as organizational measures, such as black- and whitelists, spam filters, captchas etc. The Semantic Pingback mechanism prevents spamming by the following verification method.

When the Pingback Server receives the notification signal, it automatically fetches the linking resource, checking for the existence of a valid incoming link or an admissible assertion about the target resource. The Pingback server defines, which types of links and information are admissible. This can be based on two general strategies:

- *Information analysis.* Regarding an analysis of the links or assertions, the Pingback server can, for example, dismiss assertions which have logical implications (such as domain, range or cardinality restrictions), but allow label and comment translations into other languages.
- *Publisher relationship analysis.* This can be based e.g. on the trust level of the publisher of the linking resource. A possibility to determine the trust level is to resolve `foaf:knows` relationships from the linked resource publisher to the linking resource publisher.

If admissible links or assertions exist, the Pingback is recorded successfully, e.g. by adding the additional information to the target resource and notifying its publisher. This makes Pingbacks less prone to spam than e.g. trackbacks[80].

80 http://en.wikipedia.org/wiki/Trackback

In order to allow conventional Pingback servers (e.g. WordPress) to receive links from the Data Web, this link must be represented in a respective HTML representation of the linking resource (managed by the Pingback client) at least as an untyped HTML link. This enables the server to verify the given source resource even without being aware of Linked Data and RDF.

5.4.2 *Backlinking*

The initial idea behind propagating links from the publisher of the source resource to the publisher of the target resource is to automate the creation of backlinks to the source resource. In typical Pingback enabled blogging systems, a backlink is rendered in the feedback area of a target post together with the title and a short text excerpt of the source resource.

To retrieve all required information from the source resource for verifying the link and gather additional data, a Semantic Pingback server will follow these three steps:

1. Try to catch an RDF representation (e.g. RDF/XML) of the source resource by requesting Linked Data with an HTTP `Accept` header.
2. If this is not possible, the server should try to gather an RDF model from the source resource employing an RDFa parser.
3. If this fails, the server should at least verify the existence of an untyped HTML link in the body of the source resource.

Depending on the category of data which was retrieved from the source resource, the server can react in different ways:

- If there is only an *untyped HTML* link in the source resource, this link can be created as an RDF triple with a generic RDF property like `dc:references` or `sioc:links_to` in the servers knowledge base.
- If there is at least one *direct link* from the source resource to the target resource, this triple should be added to the servers knowledge base.
- If there is any other triple in the source resource where either the subject or the object of the triple corresponds to the target resource, the target resource can be linked using the `rdfs:seeAlso` property with the source resource.

In addition to the statements which link the source and the target resource, meta data about the source resource (e.g. a label and a description) can be stored as well.

5.4.3 *Provenance Tracking*

Provenance information can be recorded using a provenance vocabulary such as Hartig [2009][81]. This vocabulary describes provenance information based on data access and data creation attributes as well as three basic provenance related types: executions, actors and artifacts. Following the specification in Hartig [2009], we define a *creation guideline* for Pingback requests, identified by `ping:RequestGuideline`. A specific Pingback request *execution* is then performed by a Pingback *data creating service*, which uses the defined creation guideline.

Listing 6 shows an example provenance model represented in Turtle:

```
@prefix : <http://purl.org/net/provenance/ns#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sioc: <http://rdfs.org/sioc/ns#> .
@prefix ping: <http://purl.org/net/pingback/> .

[a rdf:Statement;
 rdf:subject <http://example1.org/Source>;
 rdf:predicate sioc:links_to;
 rdf:object <http://example2.org/Target>;
 :containedBy  [
  a :DataItem;
  :createdBy  [
   a :DataCreation;
   :performedAt "2010-02-12T12:00:00Z";
   :performedBy  [
    a :DataCreatingService;
     rdfs:label "Semantic Pingback Service" ];
     :usedData  [
      a :DataItem;
       :containedBy <http://example1.org/Source> ];
    :usedGuideline [a ping:RequestGuideline ]
]];].
```

Listing 6: Provenance model of an example Semantic Pingback request.

This provenance model describes a Pingback from `http://example1.org/Source` to `http://example2.org/Target`. The Pingback was performed Friday, 12 February at noon and resulted in a single statement, which links the source resource to the target resource using a `sioc:links_to` property.

81 The Provenance Vocabulary Core Ontology Specification is available at `http://trdf.sourceforge.net/provenance/ns.html`.

6

ACCESS DELEGATION FOR THE WEBID PROTOCOL

The results presented in this chapter were primarily published in Tramp et al. [2012] as well as partly in Tramp et al. [2014].

In this Chapter, we present our WebID protocol extension as part of the proposed DSSN architecture.

Agents on the web communicate with each other through a limited number of actions: by making requests for resources (`GET`), by creating resources (`POST` or `PUT`), or even by deleting resources. Creation or deletion of resources usually require authentication of the agent making the request, and so in many cases do requests for information. A *WebID* is a URI that refers to an agent — person, robot, group or other thing that can act intentionally. The WebID should be a URI which when dereferenced returns a representation whose description uniquely identifies the agent as the controller of a public key such as defined in Story et al. [2013].

The notion of an agent include persons directly controlling a browser or other software as well as software application which are not directly under the control of a user. This second category includes all types automation and productivity services such as Pocket[82] or IFTTT[83] as well as all applications from DSSN service and application layer (see Section 4.4).

The WebID protocol enables the global identification of agents using asymmetric cryptography in a way that fits cleanly with web architecture namely in such a way that agents can verify each others identity without having had any previous interactions and in such a way as to allow trust to build up in a decentralized manner through a linked web of trust.

It is worth noting that the host serving the WebID profiles controls the identity of every agent whose URI is within that server's namespaces. This service is known as the origin server [Barth, 2011].

The WebID protocol is designed for client authentication[84]. But the origin server does not just respond to requests, the server is also able to make requests. Indeed WebID authentication requires the server to make WebID profile requests to other servers in order to verify the identity of agents. Please note that fetching a WebID profile for WebID

82 Pocket (`http://getpocket.com/`) is a management tool for keeping articles for reading later.

83 IFTTT (`https://ifttt.com`) is an acronym for 'If This Then That', which is a service to connect different web applications through simple conditional statements such as 'If someone is posting a picture of me on Facebook, then backup it on my Google Drive.'

84 Server authentication using IETF DANE follows much the same logic, except that the lookup for the identity is not done using the HTTP protocol but DNSSEC [Hoffman and Schlyter, 2012].

authentication should be done anonymously, for fear of authentication deadlocks[85].

Things get more interesting in the authorization space. Consider a very natural application of WebID: Allowing friends of one's friends access to some resources. This authorization rule will require the web server to fetch each of the resource owner's friends profiles, in order to build up the list of authorized users. But there is a privacy issue involved here: not everyone wants to make all of their social network publicly visible, and some may not want to make any of it publicly visible. Those people may then protect their FOAF profile with access control rules such as one only allowing friends of their friends access to it. How can a server that needs access to these FOAF profiles in order to apply its own access control rules get access to the information?

This Chapter describes an extension of the WebID protocol in order to allow access problems as described above. It is structured in the following way:

- Section 6.1 describes preliminary requirements which we had in mind for our solution and clarifies some important terms,
- Section 6.2 goes into detail with the WebID specification and adds support for authorization delegation.
- Finally, in Section 6.3 we describe our reference scenarios based on two web applications.

6.1 REQUIREMENTS

In order to make discussion of the problems easier, we distinguish the following roles in the access delegation process:

- The *secretary* acts in the name of another agent, the *principal*.
- The *principal* is the agent who has a secretary that acts on its behalf.
- The *resource guard* is the application which has to decide if a request should be processed or not.

The solution we propose will be based on the following general principles:

85 For example one can imagine an agent S with profile P_S requesting a resource on server R which requires authentication. S would send R its certificate, thereby requiring R to dereference S's profile P_S in order to verify the WebID. If P_S itself requires authentication of R and if R sends a certificate containing a WebID with its profile P_R, and if P_R itself requires authentication then a deadlock exists.

Distinguish secretary from principal

The identity of the different agents should as far as possible be transparent. In the context of access delegation this means that a secretary should have it's own WebID. The motivation for this is feature-driven as well as technical:

- It allows resource guards to permit or deny requests based on this information.
- Secretary that have many principals do not need to switch their certificate between requests.
- It makes it possible to describe the relationship between a principal and its secretary using Linked Data.
- If we do not distinguish secretary and principle in this way, then either the principal need to create and publish many different key pairs for each secretary or the principle need to push its private keys to each of its secretaries in order to allow access for them.

Easy to use

The one and only place to describe which secretary are allowed to operate for a principal should be the principal's WebID profile. To grant delegated access to a secretary agent, no other actions than adding 1 triple to the WebID profile should be needed. Retracting this grant should involve simply removing it from the WebID profile.

Linked Data and Read Write Web integration

The solution we try to architect aims to enhance the communication for consumption and modification of Linked Data especially from the applications point of view. This means, that existing Linked Data as well as Read-Write-Web principles should not be violated by the architecture.

Minimal protocol footprint

By using HTTP and working declaratively by placing statements in documents, we make adoption of the delegation easier and avoid complex protocol developments. We believe that this is a crucial feature of Linked Data in general.

Efficiency

Finally, the proposed solution should scale with growing number of users and connections. In our context this means that an Social Web application should be able to act in the name of thousands of users.

6.2 EXTENDING WEBID FOR ACCESS DELEGATION

To fulfill the stated requirements, we extended the WebID protocol in the following way:

Firstly, we defined an OWL object property which connects a principal with its secretaries. This object property is currently identified as `cert:secretary` in the namespace of the "Ontology for Certificates and crypto stuff." An example is listed in Listing 7.

```
25 @prefix cert: <http://www.w3.org/ns/auth/cert#">.
26 <http://philipp.frischmuth24.de/id/me> cert:secretary <http://akws.org/this>.
```

Listing 7: Extension of the minimal WebID profile from Listing 2: Adding a secretary relationship to the WebID of an OntoWiki instance (see Section 6.3)

Secondly, we extended the WebID authentication sequence in a way that resource guards can distinguish between the requesting agent (the secretary) and the agent on which behalf the secretary wants to request this resource (the principal). This is done by using the defined `cert:secretary` as well as an experimental HTTP request header field. The new authentication sequence is depicted in Figure 15 (on Page 63).

The following enumeration describes each authentication step of Figure 15 in detail but concentrates on the context of access delegation:

(1) The secretary opens a TLS connection with the server of the protected resource.

(2) Once TLS is set up, the HTTP request is sent to the server (e.g. a `HTTP GET`), with an additional `X-On-Behalf-Of` header, which thereby defines the requesting agent as a *secretary* and the referred to agent as the *principal*.

(3) The guard intercepts this request, and in turn requests client authentication using TLS session renegotiation. The *secretary* authenticates as itself by sending a certificate containing a WebID referring to it. The TLS-Light service verifies that the *secretary* really is in possession of the private key corresponding to the public key sent in the certificate. This is defined in the TLS protocol [Dierks and Rescorla, 2008].

(4) The guard asks the verification agent to verify the WebID.

(5a) The verification agent verifies the secretary WebID which is named in the certificate. This process is exactly as described in the WebID protocol [Story et al., 2009]. The guard also asks the verifier agent to check the secretary claim implied by the `X-On-Behalf-Of` header.

(5b) The *principal* agent's relation to the *secretary* is verified by dereferencing the *principal*'s WebID profile, and verifying it responds with a true to the SPARQL ASK query `ASK {$principal :secretary $secretary.}` where the `principal` and `secretary` variables have been bound to the correct URIs.

(6) The authentication and verification process having succeeded, the authorization process checking if the *principal* would get access to the requested resource.

(7) The resource representation can then be returned or not, depending on the access control rules.

6.3 APPLICATION SCENARIOS

We describe two different application scenarios where WebID delegation is essential: MyProfile is a WebID identity service application by Andrei Sambra and OntoWiki a semantic data wiki.

MyProfile

MyProfile[86] is a web service demonstrating how easy it is to both create a WebID profile, and to build up distributed social web applications upon it. Its main purpose is to provide a unified user account through a simple *user profile*. Currently these are tied to the MyProfile project web site, but it has been designed from the ground up to work with distributed Linked Data, making it easy to dissociate the software stack from the MyProfile project domain name, allowing it to be deployed on a machine under the user's control such as a FBx.

It is very important for MyProfile to be able to use WebID access delegation, because a single MyProfile server instance can host multiple users, and must fetch resources for each user asynchronously in order to be able to provide a seamless and rapid user experience. To improve user experience and overall performance, a caching mechanism is used to refresh local copies or "views" of external data. With multiple users coexisting on the same server, the caching mechanism needs to be able to distinguish views of remote resources as seen by different users, as they are served by remote servers depending on their access control and resource filtering policies. As the number of users on MyProfile

86 http://myprofile-project.org/

grows this has to be done efficiently, and so re-using TLS connections where possible is important.

OntoWiki

OntoWiki (Auer et al. [2006], more in detail discussed in Chapter 9) is a web application, which allows publication, exploration as well as manipulation of arbitrary RDF knowledge bases in distributed scenarios. We refer to it as a data wiki, since it adopts the wiki philosophy (ease of editing, tracking of changes, integrated discussions) on the one hand, while focusing on structured information on the other hand. Furthermore OntoWiki is an adaptable application framework, which supports the creation of Linked Data based applications on the web [Heino et al., 2009]. In addition to the usual features of wikis, OntoWiki provides a sophisticated extension system, such that it can be adapted for a variety of use-cases. Although the wikis usually enable anyone to edit everything, numerous real-world applications require access-control mechanisms. OntoWiki has built-in support for authorization on graph and action level. Furthermore several authentication protocols can be employed, including amongst others the WebID protocol.

A first use-case for WebID access delegation within OntoWiki arises from the need to *import external data*. Since WebID profiles can contain personal information, access to such data should be restricted with the WebID protocol. Although a user may (or may not, in the case of a periodically executed automatic synchronization process) initiate the import procedure manually via the OntoWiki user interface, the actual fetching is done in the background. An OntoWiki instance does not know of any private keys of users of the system. Thus the system is not able to use that information when requesting data. With WebID access delegation though, profiles can be fetched on behalf of the user instead.

Another use-case where access delegation can be employed is within the Semantic Pinback procedure (see Chapter 5). With Semantic Pingback owners of resources can be notified when for example a link to such a resource is created elsewhere on the web. In order to protect the protocol against spam attacks, a Pingback server will fetch the desired resource and check, whether the stated link is indeed contained in the data. The source resource that links to the target resource and thus is fetched by a Pingback server might be access restricted, for example in a scenario where a friending process is initiated Story et al. [2011]. For privacy reasons the owner of the WebID profiles will very likely hide the triples in question (e.g. `foaf:knows`) on anonymous access attempts. With WebID access delegation again, the resources can be fetched by the Pingback server on behalf of the resource owner.

This will require some further changes to the delegation protocol discussed up to now. Specifically the `:secretary` relation currently does not distinguish what kind of responsibilities the *principal* wishes to give to the *secretary*. It is currently assumed that the secretary has full rights. For OntoWiki knowing the Social Network is all that is needed to make access control decisions and full delegation powers may not be needed. The ability to describe more limited secretary relations could be very helpful here.

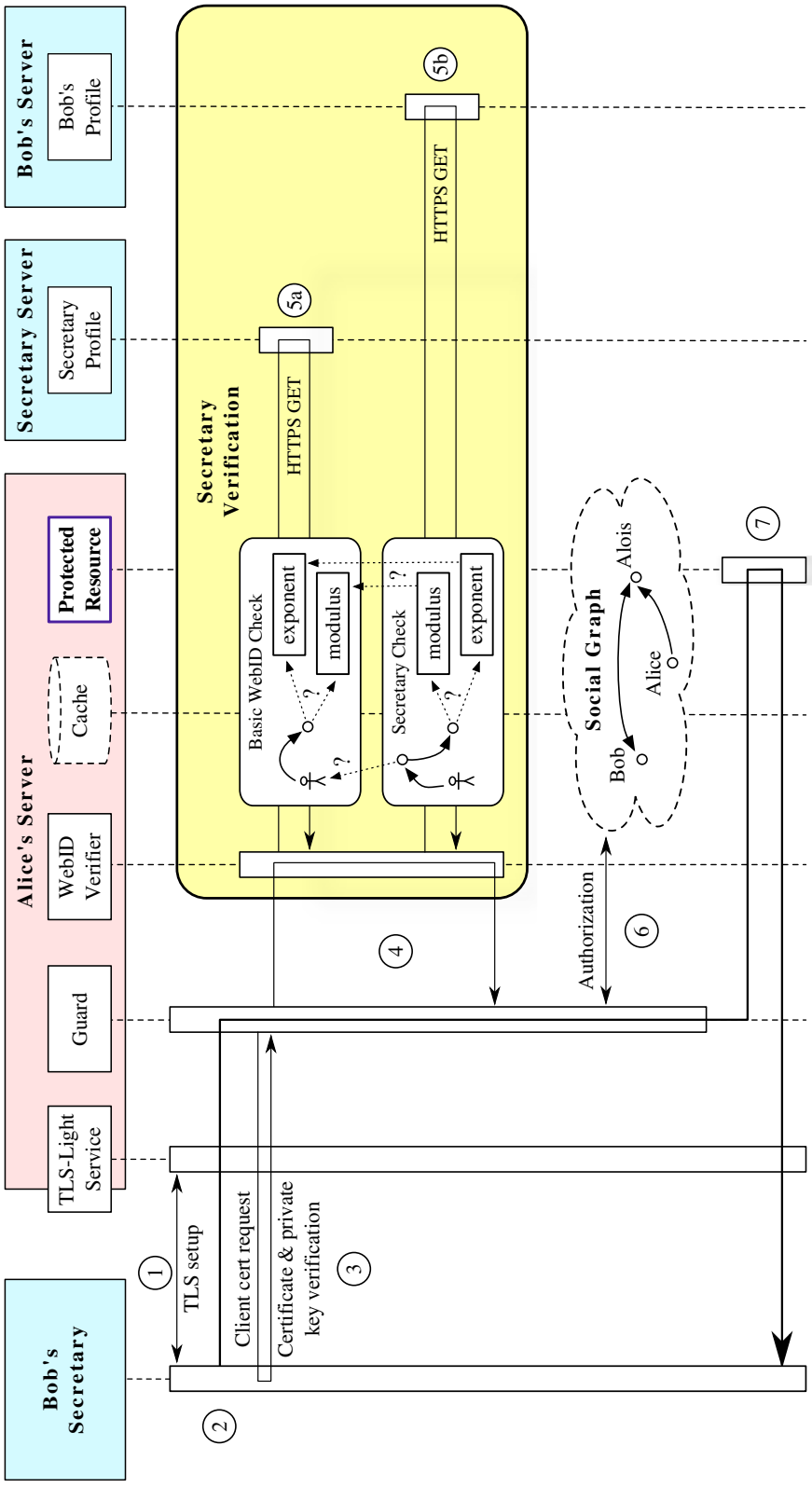

Figure 15: Extended WebID authentication sequence

Part III

APPLICATIONS

This part is structured in three chapters each describing an application which is to an extend integrated into the proposed architecture of Part ii. xOperator (Chapter 7) is an Instant Messaging agent which allows for querying the content of the DSSN data layer by using natural language templates. The MSSW client (Chapter 8) is deeply integrated Android DSSN client which allows for Social Network contact management by using the DSSN data layer and the Semantic Pingback service. OntoWiki (Chapter 9) is a semantic data wiki massively extended to be well connected in the DSSN architecture both as a consumer and a producer of Social Network data.

All three applications use a different subset of the proposed DSSN architecture as well as provide fundamental different user interfaces and usage concepts. xOperator provides query capabilities by using a natural language chat interface to "talk" to the Social Network. The MSSW client is a backend service provider of the Android plattform and is seamless integrated to bridge the DSSN data layer and the local phones contact data. OntoWiki is a browser based client which allows for sending and receiving activities as well as managing Social Network contacts.

7

XOPERATOR – AN INSTANT MESSAGING AGENT

The results presented in this chapter were primarily published in Dietzold et al. [2008b] and Dietzold et al. [2008c]. In addition to that, a vague idea of this application was coined by my colleagues Michael Martin and Sebastian Hellmann in 2007.

In this Chapter, we present the Intant Messaging client xOperator as specific user interface to access data from the proposed DSSN architecture via text based chat communication.

With estimated more than 500 million users Instant Messaging (IM) is in addition to Web and Email the most popular service on the Internet. Instant Messaging is used to maintain a list of close contacts (such as friends or co-workers), to synchronously communicate with those, exchange files or meet in groups for discussions. Examples of IM networks are ICQ, Skype, AIM or the Jabber protocol and network[87]. The latter is an open standard and the basis for many other Instant Messaging networks such as Google Talk, Meebo and Gizmo. With xOperator we present a strategy and implementation which deeply integrates Instant Messaging networks with the Semantic Web in general and the data layer of a DSSN more specifically (see Section 4.2).

The xOperator application is a collaborative information agent to query RDF documents or SPARQL endpoints via natural language templates. The agents and its users as well as the agents among themselves communicate over for the Jabber network with the Extensible Messaging and Presence Protocol (XMPP). User inputs are mapped with Artificial Intelligence Markup Language (AIML) templates to SPARQL queries. These queries are requested from the agents SPARQL endpoints or routed to other agents in the Jabber roster network. The resulting agent network exists on top of the existing social network from the jabber rosters. In this chapter we present usage scenarios, the technical architecture of our implementation and evaluation results.

While there were some proposals and first attempts to bring semantic technologies together with Instant Messaging (e.g. Osterfeld et al. [2005], Franz and Staab [2005] and Shum et al. [2002]) we present a strategy and implementation, which deeply integrates both realms in order to maximize benefits for prospective users. The xOperator concept is based on the idea of additionally equipping an users' Instant Messaging identity with a number of information sources from the DSSN data layer this user owns or trusts (e.g. his WebID profile, iCal calendar etc.). Thus the Instant Messaging network is overlaid with a network of trusted knowledge sources. An Instant Messaging user can query his local knowledge sources using a controlled (but easily extensible) language based on Artificial Intelligence Markup Language (AIML) templates [Wallace, 2005]. In order to pass the gen-

87 http://www.jabber.org/

erated machine interpretable queries to other xOperator agents of friends in the Instant Messaging network, xOperator makes use of the standard message exchange mechanisms provided by the Instant Messaging protocol. After evaluation of the query by the neighboring xOperator agents, results are transferred back, filtered, aggregated and presented to the querying user.

Such a deep integration of semantic technologies and Instant Messaging bears a number of advantages and benefits for users when compared to the separated use of Semantic Web technologies and Instant Messaging. From our point of view the two most crucial ones are:

- *Context awareness.* Users are not required to world wide uniquely identify entities, when it is clear what/who is meant from the context of their social network neighborhood. When asked for the current whereabout of Sebastian, for example, xOperator can easily identify which person in my social network has the name Sebastian and can answer my query without the need for further clarification.
- *Provenance and trust.* Instant Messaging networks represent carefully balanced networks of trust. People only admit friends and colleagues to their contact list, who they trust seeing their online presence, not being bothered by SPAM and sharing contact details with. Overlaying such a social network with a network for semantic knowledge sharing and querying naturally solves many issues of provenance and trust.

The Chapter is structured as follows:

- After presenting envisioned usage scenarios and requirements in Section 7.1,
- we exhibit the technical xOperator architecture in Section 7.2.
- We report about a first xOperator evaluation according to different use cases in Section 7.3,
- and present related work in Section 7.4.

7.1 COMMUNICATION SCENARIOS AND REQUIREMENTS

This section describes the three envisioned agent communication scenarios for xOperator. We will introduce some real-world application scenarios also later in Section 7.3. Figure 16 shows a schematic depiction of the communication scenarios. The figure is divided vertically into four layers.

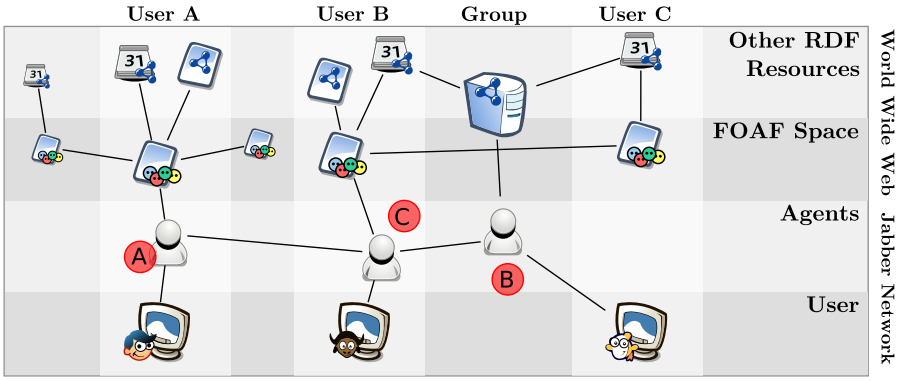

Figure 16: Agent communication scenarios: (A) personal agent, (B) group agent, (C) agent network.

- The first two layers represent the DSSN data layer in the World Wide Web. Mutually interlinked data layer resources (such as WebID profiles) reference each other using relations such as `foaf:knows`. These DSSN data layer resources could have been generated manually, exported from databases or could be generated from other information sources. These can be, for example, mailing list archives which are represented as SIOC ontologies or personal calendars provided by public calendaring servers such as Google calendar. In order to make such information available to the DSSN a variety of transformation and mapping techniques can be applied. For the conversion of iCal calendar information for example we used Masahide Kanzaki's ical2rdf service[88].

- The lower two layers in Figure 16 represent the Jabber Network. Here users are interacting synchronously with each other, as well as users with artificial agents (such as xOperator) and agents with each. A user can pose queries in natural language to an agent and the agent transforms the query into one or multiple SPARQL queries. Thus generated SPARQL queries can be forwarded either to a SPARQL endpoint or neighboring agents via the Instant Messaging networks transport protocol (XMPP in the case of Jabber). SPARQL endpoints evaluate the query using a local knowledge base, dynamically load RDF resources from the Linked Data Web or convert Web accessible information sources into RDF. The results of SPARQL endpoints or other agents are collected, aggregated, filtered and presented to the user depending on the query as a list, a table or a natural language response.

The different communication scenarios are described in the following subsections:

88 http://www.kanzaki.com/courier/ical2rdf

7.1.1 *Personal Agent (A)*

This scenario is the most important one and also builds the foundation for the other two communication scenarios. A user of an Instant Messaging network installs his own personal agent and configures information sources he owns or trusts. For easy deployment the software representing the agent could be distributed together with (or as a plugin of) the Instant Messaging client (such as Pidgin[89] or Adium[90]).

Information sources can be for example a WebID profile of the user containing personal information about herself and about relationships to other people she knows and where to find further information about these. This information is represented in FOAF using the properties `foaf:knows` and `rdfs:seeAlso`. Please have a look on Listing 2 for an example of a minimal WebID profile.

Additionally this WebID profile can link to other RDF documents in the DSSN data layer which contain more information about the user and his activities. The RDF version of his calendar (using `ical:Vcalendar` from the `ical` namespace), for example, could be linked as follows:

```
1 @prefix ical: <http://www.w3.org/2002/12/cal/ical#>.
2 <http://philipp.frischmuth24.de/id/me> rdfs:seeAlso <http://.../ical2rdf?u=http...
      > .
3 <http://.../ical2rdf?u=http...> a ical:Vcalendar;
4   rdfs:label "My Calendar" .
```

Listing 8: Extension of the minimal WebID profile from Listing 2: Integrating a machine readable calendar description.

Such links span a network of information sources as depicted in Figure 16. Each user maintains his own part of the DSSN data layer and links to resources of his acquaintances. Depending on the query, the agent will access the respective resources. The following example queries are possible, when WebID profiles are known to the agent:

- Tell me the phone / homepage / ... of Frank!
- What is the birthday of Michael?
- Where is Dave now?
- Who knows Alex?

7.1.2 *Group Agent (B)*

This communication scenario differs from the Personal Agent scenario in that multiple users get access to the same agent. The agent should

89 http://pidgin.im/
90 https://adium.im/

be able to communicate with multiple persons at the same time and to answer queries in parallel. As this is also depicted in Figure 16 the agent furthermore does not only access DSSN data layer resources but can also use a triple store for answering queries. When used within a corporate setting this triple store, can for example, contain a directory with information about employees or customers. The triple store can be also used to cache information obtained from other sources and thus facilitates faster query answering. For agents themselves, however, the distinction between resources on the Web and information contained in a local triple store is not relevant.

7.1.3 *Agent Network (C)*

This scenario extends the two previous ones by allowing communication and interaction between agents. The rationale is to exploit the trust and provenance characteristics of the Instant Messaging network: Questions about or related to acquaintances in my network of trust can best be answered by their respective agents. Hence, agents should be able to talk to other agents on the Instant Messaging network.

First of all, it is crucial that agents on the Instant Messaging network recognize each other. A personal agent can use the account of its respective owner and can access the contact list (also called roster) and thus a part of its owner's social network. The agent should be able to recognize other personal agents of acquaintances in this contact list (auto discovery) and it should be possible for agents to communicate without interfering with the communication of their owners. After other agents are identified it should be possible to forward SPARQL queries (originating from a user question) to these agents, collect their answers and present them to the user.

7.2 TECHNICAL ARCHITECTURE

First of all, the xOperator agent is a mediator between the Jabber Instant Messaging network on one side and the World Wide Web on the other side. The xOperator is a client in both networks. He communicates anonymously (or using configured authentication credentials) on the WWW by talking HTTP with Web servers. On the Jabber network xOperator utilizes the Extensible Messaging and Presence Protocol (XMPP, Saint-Andre [2004]) using the Jabber account information provided by its owner. Jabber clients only communicate with the XMPP server associated with the user account. Jabber user accounts are have the same syntax as email addresses (e.g. `soerenauer@jabber.ccc.de`). The respective Jabber server cares about routing messages to the server associated with the target account or temporarily stores the message in case the target account is not online or its server is not reachable. Since 2004 XMPP is a standard of the Internet Engineering Task Force

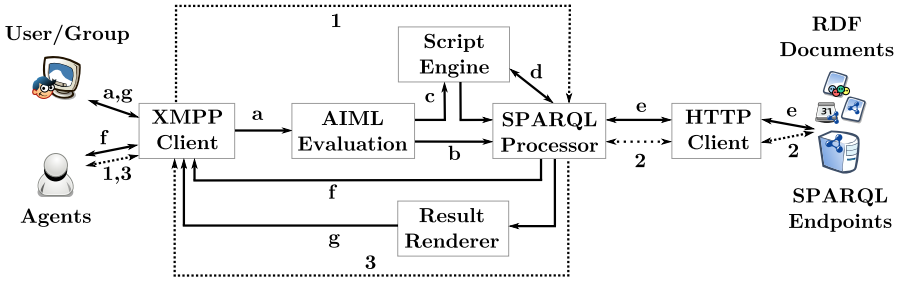

Figure 17: Technical architecture of xOperator.

and is widely used by various services (e.g. Google Talk). Figure 17 depicts the general technical architecture of xOperator.

The agent works essentially in two operational modi:

1. (Uninterrupted line) Answer natural language questions posed by a user using SPARQL queries and respond to the user in natural language according to a predefined template. Questions posed by a user (a) are either directly mapped to a SPARQL query template (b) or SPARQL queries are generated by a query script (c), which might obtain additional information by means of sub queries (d). The resulting SPARQL query will be evaluated on resources of the user (e), as well as passed on the Jabber network to neighboring agents for evaluation (f). All returned results are collected and prepared by a result renderer for presentation to the user (g). Algorithm 1 demonstrates the workings of xOperator.

2. (Dotted line) Receive SPARQL queries from neighbouring agents (1) on the IM network, evaluate these queries (2) on the basis of locally known RDF documents and SPARQL endpoints and send answers as XML SPARQL Result Set [Beckett and Broekstra, 2013] back via XMPP (3).

In both cases the agent evaluates SPARQL queries by querying a remote SPARQL endpoint via HTTP `GET` Request according to the SPARQL HTTP Bindings [Clark, 2013] or by retrieving an RDF document as well via HTTP and evaluating the query by means of a local SPARQL query processor.

In the following we describe first the natural language component on the basis AIML templates and address thereafter the communication in the Jabber network.

7.2.1 *Evaluation of AIML Templates*

The Artificial Intelligence Markup Language (AIML, Wallace [2005]) is an XML dialect for creating natural language software agents. In Freese [2007] the authors describe AIML to enable pattern-based, stimulus-response knowledge content to be served, received and processed

Algorithm 1: Evaluation of XMPP user input.

Input: User input I from XMPP
Output: Sendable Agent response
Data: set $S = A \cup D \cup E$ of agents, documents and endpoints
Data: set C of AIML categories
Data: set $R = \emptyset$ of results

```
1  if I is an admin or extension command then return executeCommand
   (I)
2  else if I has no match in C then return defaultmsg
3  else if I has standard match in C then return aimlResult (I, C)
4  else
5      if I has SPARQL template match in C then
6          Query = fillPatterns (aimlResult (I, C) )
7      else if I has query script match in C then
8          Query = runScript (aimlResult (I, C) )
9  if Query then
10     foreach s ∈ S do
11         R = R ∪ executeQuery(Query, s)
12     return renderResults (R);
13 else
14     return error
```

on the Web and offline in the manner that is presently possible with HTML and XML. AIML was designed for ease of implementation, ease of use by newcomers, and for interoperability with XML and XML derivatives such as XHTML. Software reads the AIML objects and provides application-level functionality based on their structure. The AIML interpreter is part of a larger application generically known as a bot, which carries the larger functional set of interaction based on AIML. A software module called a responder handles the human-to-bot or bot-to-bot interface work between an AIML interpreter and its object(s). In xOperator AIML is used for handling the user input received through the Instant Messaging network and to translate it into either a query or a call to a script for more sophisticated evaluations.

The most important unit of knowledge in AIML is the category. A category consists of at least two elements, a pattern and a template element. The pattern is evaluated against the user input. If there is a match, the template is used to produce the response of the agent. It is possible to use the star ($*$) as a placeholder for any word in a pattern. We have extended this basic structure in two ways:

Simple Query Templates

In order to enable users to create AIML categories on the fly we have created an extension of AIML. It allows to map natural language patterns to SPARQL query templates and to fill variables within those templates with parameters obtained from ∗-placeholders in the natural language patterns.

```
<category>
  <pattern>TELL ME THE PHONE OF *</pattern>
  <template>
    <external name="query"
      param="SELECT DISTINCT ?phone WHERE {...}" />
  </template>
</category>
```

Within the SPARQL template variables in the form of %%n%% refer to ∗-placeholder (n refers to the n^{th} ∗-placeholder in the category pattern). The question for the phone number of a person, for example, can be represented with the following AIML template:

```
TELL ME THE PHONE OF *
```

A possible (very simple) SPARQL template using the FOAF vocabulary could be stored within the AIML category as follows:

```
SELECT DISTINCT ?phone WHERE
    { ?s foaf:name "%%1%%". ?s foaf:phone ?phone. }
```

On activation of a natural language pattern by the AIML interpreter the corresponding SPARQL templates variables are bound to the values of the placeholders and the resulting query is send independently to all known SPARQL endpoints and neighboring agents. These answer independently and deliver result sets, which can complement each other, contain the same or contradictory results. The agent renders results as they arrive to the user, but filters duplicates and marks contradictory information. The agent furthermore annotates results with regard to their provenance.

This adoption of AIML is easy to use and directly extensible via the Instant Messaging client (cf. Section 7.2.2). However, more complex queries, which for example join information from multiple sources are not possible. In order to enable such queries we developed another AIML extension, which allows the execution of query scripts.

Query Scripts

Query scripts basically are small pieces of software, which run in a special environment where they have access to all relevant subsystems. They are given access to the list of known data sources and neighboring agents. xOperator, for example, allows the execution of query scripts in the Groovy scripting language for Java.The execution of a query script results in the generation of a SPARQL query, which is

evaluated against local information sources and passed to other agents as described in the previous section. We motivate and illustrate the workings of query scripts using an application scenario based on the WebID profile based Linked Data network (cf. Figure 16), which has the following characteristics:

- The `foaf:knows` relation points to other people known by this person.
- Other resources are linked through `rdfs:seeAlso`, allowing bots and agents to crawl through the FOAF space and to gather additional RDF documents like calendars or blog feeds.

To enable the agent to retrieve and evaluate additional information from sources, which are referenced from the user's WebID profile, a query script can contain subqueries, whose results are used within another query[91]. Query scripts also enable the usage of special placeholders such as *now* or *tomorrow*, which can be populated for the querying of iCal calendars with the concrete values.

In order to extend the agent for other application domains or usage scenarios, xOperator allows to dynamically assign new query scripts to AIML categories. A query script is assigned to an AIML template by means of an *external* tag (as are also simple SPARQL templates). An example script implementing a subquery to retrieve relevant resources about a `foaf:person` is presented in Section 7.3.

7.2.2 *Administration and Extension Commands*

Users can easily change the configuration of their agents by using a set of administration and extension commands. These commands have a fix syntax and are executed without the AIML engine and consists of the following categories:

- commands to manage trusted data sources
  - `list ds`
  - `add ds {name} {uri}`
  - `del ds {name}`

  Each source is locally identified by a name which is associated to an URI.
- template commands, to manage simple query templates which are associated by its AIML pattern
  - `list templates`
  - `add template {pattern} {query}`

91 Note that this is possible with SPARQL 1.1 subqueries too. TODO: reference!!

  - `del template {pattern}`

- A command to send on-the-fly SPARQL queries to the xOperator
  - `query {SPARQL query}`:

  The query will be evaluated on every data store and routed to every agent in the neighborhood. The query results will be rendered by a default renderer.

- namespace management commands
  - `list ns`
  - `add ns {prefix} {uri}`
  - `del ns {prefix}`

  The namespaces will be added to the namespace section in the on-the-fly query.

- an entry point for the help system
  - `help`

7.2.3 *XMPP Communication and Behavior*

While the HTTP client of the agent uses standard HTTP for querying SPARQL endpoints and the retrieval of RDF documents, we extended XMPP for the mutual communication between the agents. This extension complies with standard extension routines of XMPP will be ignored by other agents. With regard to the IM network the following functionality is required:

- The owner of the agent should be able to communicate easily with the agent. He should be able to manage the agent using the contact list (roster) and the agent should be easily recognizable.
- The agent has to have access to the roster of its owner in order to identify neighboring agents.
- It should be possible for other agents to obtain information about the ownership of an agent. Its requests will not be handled by other agents for security reasons if it can not be clearly assigned to an owner.
- The agent should be only visible for his owner and neighboring agents (i.e. agents of contacts of his owner) and only accept queries from these XMPP accounts.

As a consequence from those requirements it is reasonable that the agent acts using the account of its owner (main account) for the communication with other agents, as well as an additional account

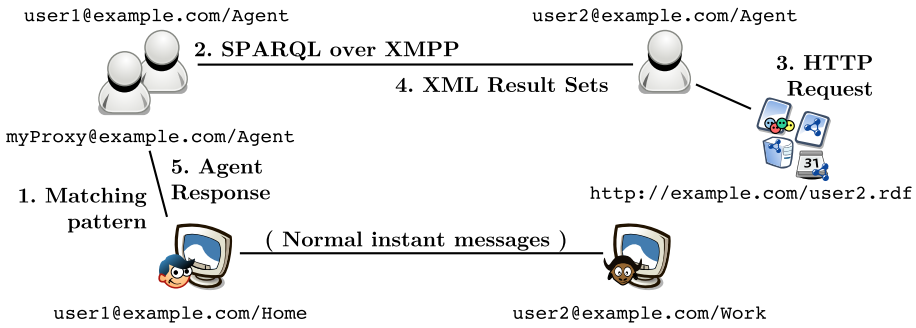

Figure 18: XMPP Communication example

(proxy account) for the communication with its owner[92]. Due to the usage of the main account other agents can trust the agents answers and easily track the provenance of query results. Figure 18 depicts the concept of using two different accounts for the communication with the owner and other agents. For unique identification of senders and recipients so called resource names (in the figure `Home`, `Work` and `Agent`) are used and simply appended to the account name.

We demonstrate the agent communication with two XMPP messages:

Agent Autodiscovery

Goal of the autodiscovery is the identification of agents among each other. For that purpose each agent sends a special message of type info/query (`iq`) to all known and currently active peers. Info/query messages are intended for internal communication and queries among Instant Messaging clients without being displayed to the human users. An autodiscovery message between the two agents from Figure 18, for example, would look as follows:

```
<iq from="user1@example.com/Agent" type='get'
    to="user2@example.com/Agent" id='...'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

A positive response to this feature discovery message from an xOperator agent would contain a feature with resource ID `http://www.w3.org/2005/09/xmpp-sparql-binding`. This experimental identifier/-namespace was created by Dan Brickley for SPARQL / XMPP experiments (cf. Section 7.4). The response message to the previous request would look as follows:

```
<iq from='user2@example.com/Agent' type='result'
    to='user1@example.com/Agent' id='...' />
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity
```

92 Technically, it is sufficient for the agent to use the owner's account which, however, could create confusing situations for the user when communicating with 'herself'.

```
      category='client' name='xOperator' type='bot'/>
    <feature
      var='http://www.w3.org/2005/09/xmpp-sparql-binding'/>
    <!-- ... more here -->
  </query>
</iq>
```

Similar XMPP messages are used for sending SPARQL queries and retrieving results. The latter are embedded into a respective XMPP message according to the SPARQL Query Results XML Format.

Routing and Recall

Queries are propagated to all neighboring xOperator agents. As currently there is no way of anticipating which agent could answer a question, asking all directly connected agents offers the best compromise between load and recall. Flooding the network beyond adjacent nodes would cause excessive load. Especially in the domain of personal information, persons or their respective agents directly related to the querying person or agent should be most likely to answer the query.

7.3 EVALUATION

The xOperator concept was implemented in Java and is available as open-source software from: `http://aksw.org/Projects/xOperator`. The agent is able to log into existing accounts and can receive querying and configuration commands.

Figure 19: Communication with the xOperator agent by means of an ordinary Jabber client.

We evaluated our approach in a number of scenarios, which included various heterogeneous information sources and a different

	Template	Scenario:	1	2	3
1	What is / Tell me (the) * of *		2.3	3.9	1.5
2	Who is member of *		3.5	4.3	1.6
3	Tell me more about *		3.2	5.6	1.1
4	Where is * now		5.1	6.7	4.2
5	Free dates * between * and *		5.1	6.8	4.7
6	Which airports are near *		–	–	3.4

Table 3: Average response time in seconds (client to client) of some AIML patterns used in three scenarios: (1) 20 documents linked from one WebID profile, 1 personal agent with no neighborhood (2) 20 documents linked from different FOAF profiles and spread over a neighborhood of 5 agents (3) one SPARQL endpoint as an interface to a Semantic Wiki or DBpedia store with one group agent

number of agents. As information sources we used WebID profiles (20 documents, describing 50 people), the SPARQL endpoint of our semantic Wiki OntoWiki [Auer et al., 2006] (containing information about publications and projects), information stored in the LDAP directory service of our department, iCal calendars of group members from Google calendar (which are accessed using iCal2RDF) and publicly available SPARQL endpoints such as DBpedia [Auer et al., 2007]. Hence the resulting information space contains information about people, groups, organizations, relationships, events, locations and all information contained in the multidomain ontology DBpedia. We created a number of AIML categories, interacting with this information space. Some example patterns and corresponding timings for obtaining answers from the agent network in the three different network scenarios (personal agent, agent network and group agent) are summarized in Table 3.

The first three templates represent queries which are answered using simple SPARQL templates. Template 4 makes use of a reserved word (`now`), which is replaced for querying with an actual value. Template 5 is implemented by means of a query script which retrieves all available time slots from the calendars of a group of people and calculates the intersection thus offering suitable times to arrange meetings or events, where the attendance of all group members is required. Template 6 uses the DBpedia SPARQL endpoint in a group agent setting to answer questions about the geographic location of places (such as airports). These query templates are meant to give some insights in the wealth of opportunities for employing xOperator. Further, AIML templates can be created easily, even directly from within the IM client (using the administration and extension commands as presented in Section 7.2.2).

A typical user session showing the communication with the agent is depicted in Figure 19. The response timings indicate that the major factor are latency times for retrieving RDF documents or querying SPARQL endpoints. The impact of the number of agents in the agent network as well as the overhead required by the xOperator algorithm is rather small. The timings are furthermore upper bounds, since answers are presented to the user as they arrive. This results in intuitive perception that xOperator is a very responsive and efficient way for query answering.

Experiences during the evaluation have led to the following rules for creating patterns and queries in xOperator.

Query as fuzzy as possible

Instant Messaging is a very quick means of communication. Users usually do not capitalize words and use many abbreviations. This should be considered, when designing suitable AIML patterns. If information about the person 'Sören Auer' should be retrieved, this can be achieved using the following graph pattern: `?subject foaf:name "Auer".` However, information can be represented in multiple ways and often we have to deal with minor misrepresentations (such as trailing whitespace or wrong capitalizations), which would result for the above query to fail. Hence, less strict query clauses should be used instead. For the mentioned example the following relaxed SPARQL clause, which matches also substrings and is case insensitive, could be used:

```
1 ?subject foaf:name ?name.
2 FILTER regex(?name,'.*Auer.*','i')
```

Please note that the usage of regular expressions is resource consuming and should be avoided if better `FILTER` clauses are available (such as `bif:contains` in Openlinks Virtuoso triple store).

Use patterns instead of qualified identifiers for properties

Similar, as for the identification of objects, properties should be matched flexible. When searching for the *homepage* of 'Sören Auer' we can add an additional property matching clause to the SPARQL query instead of directly using, for example, the property identifier `foaf:homepage`:

```
1 ?subject ?slabel ?spattern.
2 ?subject ?property ?value.
3 ?property ?plabel ?ppattern.
4 FILTER regex(?spattern,'.*Auer.*','i')
5 FILTER regex(?ppattern,'.*homepage.*','i')
```

This also enables multilingual querying if the vocabulary contains the respective multilingual descriptions. Creating fuzzy queries, of

course, significantly increases the complexity of queries and will result in slower query answering by the respective SPARQL endpoint. However, since we deal with a distributed network of endpoints, where each one only stores relatively small documents this effect is often negligible. One important assumption for this rule is the availability of the vocabulary namespaces (including the `rdfs:label` values).

Use sub queries for additional documents

In order to avoid situations where multiple agents retrieve the same documents (which is very probable in a small worlds scenario with a high degree of interconnectedness) it is reasonable to create query scripts, which only distribute certain tasks to the agent network (such as the retrieval of prospective information sources or document locations), but perform the actual querying just once locally.

7.4 RELATED WORK

Proposals and first prototypes which are closely related to xOperator and inspired its development are Dan Brickley's JQbus[93] and Chris Schmidt's SPARQL over XMPP[94]. However, both works are limited to the pure transportation of SPARQL queries over XMPP.

Quite different but the xOperator approach nicely complementing are works regarding the semantic annotation of IM messages. In Osterfeld et al. [2005] for example the authors present a semantic archive for XMPP instant messaging which facilitates search in IM message archives. Franz and Staab [2005] suggests ways to make IM more semantics aware by facilitating the classification of IM messages, the exploitation of semantically represented context information and adding of semantic meta-data to messages. Comprehensive collaboration frameworks which include semantic annotations of messages and people, topics are, for example, CoAKTinG [Shum et al., 2002] and Haystack [Karger et al., 2005]. The latter is a general purpose information management tool for end users and includes an instant messaging component, which allows to semantically annotate messages according to a unified abstraction for messaging on the Semantic Web [Quan et al., 2003].

In Kaufmann and Bernstein [2007] natural language interfaces (NLIs) are used for querying semantic data. The NLI used in xOpertor employs only a few natural language processing techniques, like stop word removal for better template matching. Generic templates would be possible to define, but as Kaufmann and Bernstein [2007] shows user interaction is necessary for clarifying ambiguities. For keeping IM conversation as simple as possible, domain specific templates using

93 http://svn.foaf-project.org/foaftown/jqbus/intro.html
94 http://crschmidt.net/semweb/sparqlxmpp/

AIML were chosen. Finally, in Freese [2007] the author enhanced AIML bots by generating AIML categories from RDF models. Different to xOperator, these categories are static and represent only a fixed set of statements.

8

MSSW – A MOBILE CLIENT FOR THE DISTRIBUTED SEMANTIC SOCIAL NETWORK

The results presented in this chapter were primarily published in Tramp et al. [2011b].

In this Chapter, we present a smartphone hosted user interface to access and manipulate resources from the proposed DSSN architecture.

Smartphones, which contain a large number of sensors and integrated devices, are becoming increasingly powerful and fully featured computing platforms in our pockets. For many people they already replace the computer as their window to the Internet, to the Web as well as to social networks. Hence, the management and presentation of information about contacts, social relationships and associated information is one of the main requirements and features of today's smartphones.

The problem is currently solved solely for *centralized* proprietary platforms (such as Google mail, contacts & calendar) as well as data-silo-like social networks (e.g. Facebook). As a result of this data centralization, users' data is taken out of their hands, they have to accept the predetermined privacy and data security regulations; users are dependent of the infrastructure of a single provider, they experience a lock-in effect, since long-term collected profile and relationship information cannot be easily transferred. Increasingly, many people argue that social networks should be evolving. That is, they should allow users to control what to enter and to keep a control over their own data. Also, the users should be able to host the data on an infrastructure, which is under their direct control, the same way as they host their own website [Berners-Lee, 2010].

A possibility to overcome these problems and to give the control over their data back to the users is the mobile realization of an interface for a truly *distributed* and semantics based social network such as the DSSN architecture depicted on Figure 11.

This Chapter is structured in the following way:

- Firstly, we briefly reviewing some use cases and requirements for a mobile, semantic social network application in Section 8.1.
- Then, we give an detailed insight on the implementation of an integrated user interface for the Android plattform in Section 8.2.
- After that, we give an overview on related work in Section 8.3

8.1 MOBILE USE CASES AND REQUIREMENTS

Before describing the overall strategy, the technical architecture and our implementation we want to briefly outline in this section the

key requirements, which guided our work. These requirements are common sense in the context of social networks and are not newly coined by us. Unfortunately most of them are not achieved in the context of semantics enabled and distributed social networks, so we describe them especially from this point of view.

8.1.1 *Make new friends*

Adding new contacts to our social network is the precondition in order to gather useful information from this network. Maintaining our social network directly from your mobile phones means that we are able to instantly connect with new contacts (e.g. on conferences or parties). In the context of a distributed social network, this use-case also includes the employment of semantic search engines to acquire the WebID of a new contact based on parts of its information (typically the contacts name). In order to shorten the overall effort for adding new contacts, functionality for scanning and decoding a contacts business cards QR code[95] are also included in this use-case.

8.1.2 *Be in sync with your social network*

Once our social network is woven and social connections are established, we want to be able to gather information from this network. For a distributed social network this means, that a combination of push and pull communications is needed to be as timely updated as needed and as fast synced as possible. Especially this use-case is bound to a bunch of access control requirements[96], where people want to permit and deny access to specific information in fine grained shades and based on groups, live contexts and individuals.

8.1.3 *Annotate contacts profiles*

It should be possible to annotate profiles of contacts freely, e.g. with updated information or contact group categorizations (e.g. friends, family, co-workers). These annotations should be handled in the same way as the original data from the friend's WebID profile, except that this data is not updated with the WebID but persists as an annotation. One additional feature request in this use-case is to share these annotations across ones personal devices on the web, e.g. by pushing them to a triple store which is attached to ones WebID profile.

95 QR codes are two-dimensional barcodes which can encode URIs as well as other information. They are especially famous in Japan, but their popularity grows more and more worldwide since mobile applications for decoding them with a standard camera can be used on a wide range of devices.

96 A typical requirement: Disallow access to my mobile number except for friends and family members.

8.1.4 *General requirements*

The development of the Mobile DSSN Client was driven by a few general requirements which derived from our own experience with mobile phones and WebID profiles:

- *Be as decent as possible:* Today's WebID profile based social networks are mostly driven by uploaded RDF files. In order to support such low end profiles, there should be no other required feature on a WebID than the availability as Linked Data. All other features (WebID protocol, Semantic Pingback, subscription service) should be handled as optional features and our client should require as little infrastructure as possible.
- *Be as transparent as possible:* Mobile user interfaces are built for efficiency and daily use. People become accustomed with them and any changes in the daily work flow of using information from the social network will annoy them. The client we had in mind should work mostly invisible from the user, which means it should be well integrated into the hosting mobile operating system.
- *Be as flexible as possible:* This is especially needed in an environment where vocabularies are not yet standardized and are subject to changes and extensions. Our solution should be flexible in the sense that we do not want built-in rules on how to deal with specific attributes or relations.

Based on these preliminaries we located as well as integrated our mobile client in the DSSN architecture from Chapter 4. Figure 20 depicts this mobile usecase centered DSSN architecture.

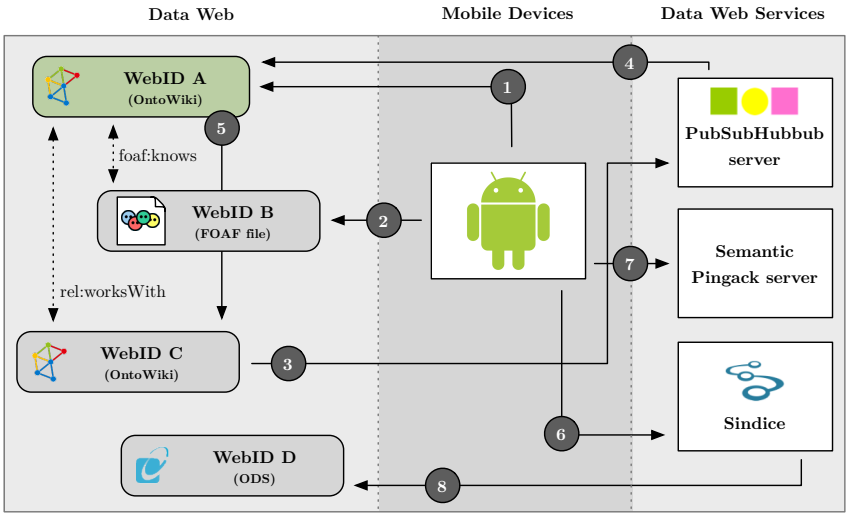

Figure 20: Mobile usecase centered DSSN Architecture

As clearly shown, the mobile device is well connected to the DSSN data layer and service layer (cf. Section 4.2 and Section 4.4). The relations between these layers and the mobile device can be described in the following way:

(1) A mobile user may retrieve updates from his social network via his WebID provider, e.g. from an OntoWiki or MyProfile (cf. Section 6.3).

(2) He may also fetch updates directly from the sources of the connected WebIDs.

(3) A WebID provider can notify a subscription service about changes.

(4) The subscription service notifies all subscribers.

(5) As a result of a subscription notification, another node can update its data.

(6) A mobile user can search for a new WebID by using a semantic search engine, e.g. Sindice.

(7) To connect to a new WebID he sends a Semantic Pingback request.

(8) The ping service notifies of the resource owner.

8.2 IMPLEMENTATION OF A MOBILE INTERFACE

After describing the architecture of a mobile usecase centered distributed, semantic social network we now present our implementation of a mobile interface for this network.

8.2.1 *Android System Integration*

Figure 21 depicts the mobile social Semantic Web client consisting of two application frameworks, which are built on top of the Android runtime and a number of libraries. In particular, *androjena*[97] is one of those libraries, which itself is a partial port of the popular Jena framework [McBride, 2002] to the Android platform. Both frameworks provided by the client share the feature that they are accessible through content providers. The Mobile Semantic Web middleware (MSW) is responsible for importing Linked Data resources (in particular via WebID auth) and persisting that data. It operates on triple level and provides access to the various triple stores through a content provider called *TripleProvider*. Each resource is stored separately, since named graphs are currently not supported. The Mobile Social Semantic Web middleware (MSSW) queries the triple data provided by MSW and

97 http://code.google.com/p/androjena/

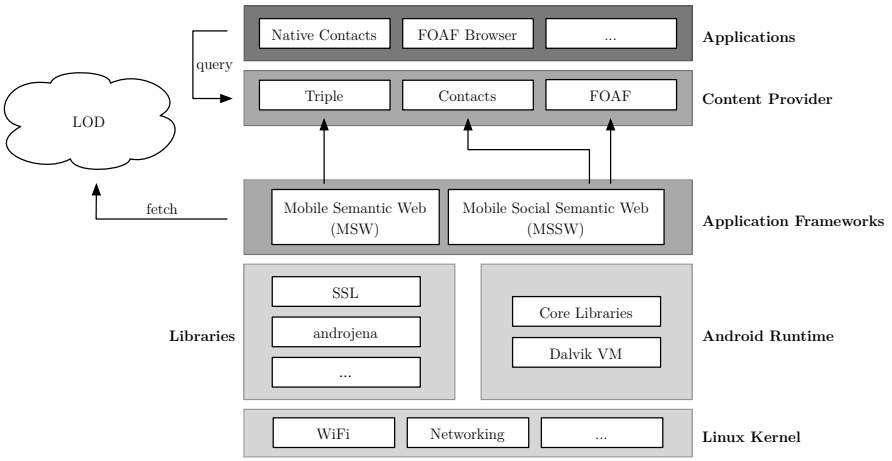

Figure 21: Android Integration Layer Cake

transforms that data into a format that is more appropriate for social applications. It propagates two content providers, one that integrates well with the layout of contact information on Android phones (*ContactProvider*) and one that is suitable for WebID based applications (`FoafProvider`).

8.2.2 *Model Management*

Since WebIDs are Linked Data enabled, they usually return data describing that resource. This circumstance makes it feasible to store a graph (referred to as a model here) for each WebID, since the redundancy between models is expected to be marginal. In reality MSW keeps more than one model per WebID for different purposes. On the mobile phones' user data space (SD card or emulated) we keep these models in the following subdirectories:

- `web` – This folder contains exact copies of the documents retrieved from the Web.
- `inf` – Models stored in this folder contain all entailed triples (more on this in Section 8.2.3).
- `local` – The user can annotate all WebIDs with personal information, which will be stored in this folder.

We decided to store all data as RDF files in the user space, since we expect the following user benefits:

- The data is more portable and can be reused on another phone or device. This makes the whole system more fail-proof.
- Most modern computers can handle SD-cards or USB cables and hence data can be easily backed up.

- Other applications on the Android phone running the mobile Semantic Web client can access and modify the data stored on the card. Thus they can further annotate the information and the client can again take advantage of such annotations.

8.2.3 *Rules and Data Processing*

One of our initial requirements from Section 8.1 is flexibility in the sense that specific vocabulary resources should not be encoded in the source code of the WebID provider. In order to achieve this requirement, we decided to encode as much data processing as possible in terms of user extensible rules. Since we employ the *androjena framework*, we were able to use the included Jena rules engine as well. All rules processed by this rule-based reasoner are defined as lists of body terms (premises), lists of head terms (conclusions) and optional names[98].

```
1  @prefix foaf: <http://xmlns.com/foaf/0.1/>.
2  @prefix android: <http://ns.aksw.org/Android/>.
3  @prefix acontacts: <http://ns.aksw.org/Android/ContactsContract.CommonDataKinds.>.
4  @prefix im: <http://ns.aksw.org/Android/ContactsContract.CommonDataKinds.Im.>.
5
6  [jabber:
7    (?s foaf:jabberID ?o), makeTemp(?d) ->
8      (?s android:hasData ?d),
9      (?d rdf:type acontacts:Im),
10     (?d im:DATA ?o),
11     (?d im:TYPE im:TYPE_HOME),
12     (?d im:PROTOCOL im:PROTOCOL_JABBER)
13 ]
```

Listing 9: Example transformation rule: If a `foaf:jabberID` is present with a WebID (line 7), then a new blank node of RDF type `acontacts:Im` is created (line 7), which is of Android IM type `HOME` (line 11) and which gets an IM protocol as well as the IM identifier (line 12 and 10).

Since we also did not want our implementation to depend on the FOAF vocabulary (alternative solutions include RDF vCards [Iannella et al., 2010]), we decided to create a native Android system vocabulary which represents the Android contacts database defined by the Android API. This vocabulary is deeply integrated into the Android system since it re-uses class and attribute names from the Android API and represents them as OWL class and datatype properties[99].

Based on this vocabulary, the given rules transform the downloaded WebID statements into Android-specific structures which are well suited for a straightforward import into the contacts provider. These

98 http://jena.sourceforge.net/inference/#RULEsyntax

99 An example class identifier is `acontacts:ContactsContract.CommonDataKinds.StructuredName`. Please have a look at the Android API reference as well (`http://developer.android.com/`).

structures are very flat and relate different Android data objects (e.g. email, photo, structured name etc.) via a `acontacts:hasData` property to a WebID. An example rule which creates an instant messaging account for the contact is presented in Listing 9.

After applying the given set of rules, the Android application post-processes the generated data in order to apply other constraints which we could not achieve with Jena rules alone. At the moment all `mailto:` and `tel:` resources are transformed to literal values, which is required for instantiating the corresponding Java class. In addition we download, resize and base64-encode all linked images. After that, the application goes through the generated data resources and imports them one by one.

8.2.4 *User perspective*

The Mobile Semantic Social Web client implementation consists of two software packages – the *Android Semantic Web Core library* containing the triple store and the *WebID content provider for Android*. Both are available on the *Android Market* since August 2010 (cf. upper left screenshot in Figure 22). According to the market statistics, they were installed more than 1000 times in 02/2014.

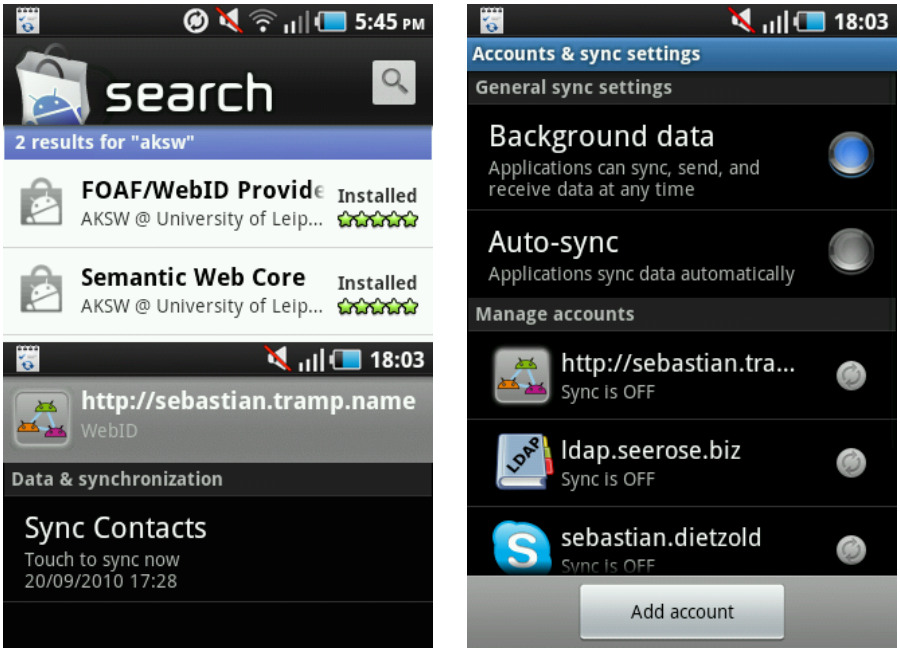

Figure 22: Screenshots of the Mobile Social Semantic Web Client: upper left side – The client as well as the triple store can be found in the official Google application market; right side – After installation, users can add a WebID account the same way they add an LDAP or Exchange account; lower left side – The account can be synchronized on request or automatically.

Once installed, a few initial configuration options have to be supplied. The right screenshot in Figure 22 shows the accounts and synchronization settings configuration menu, which allows a user to associate his WebID with his profile on the smartphone (the same way as adding an LDAP or Exchange account) and to configure synchronization intervals. The lower left screenshot in Figure 22 shows an actual WebID with the last synchronization date and the option to trigger the synchronization manually.

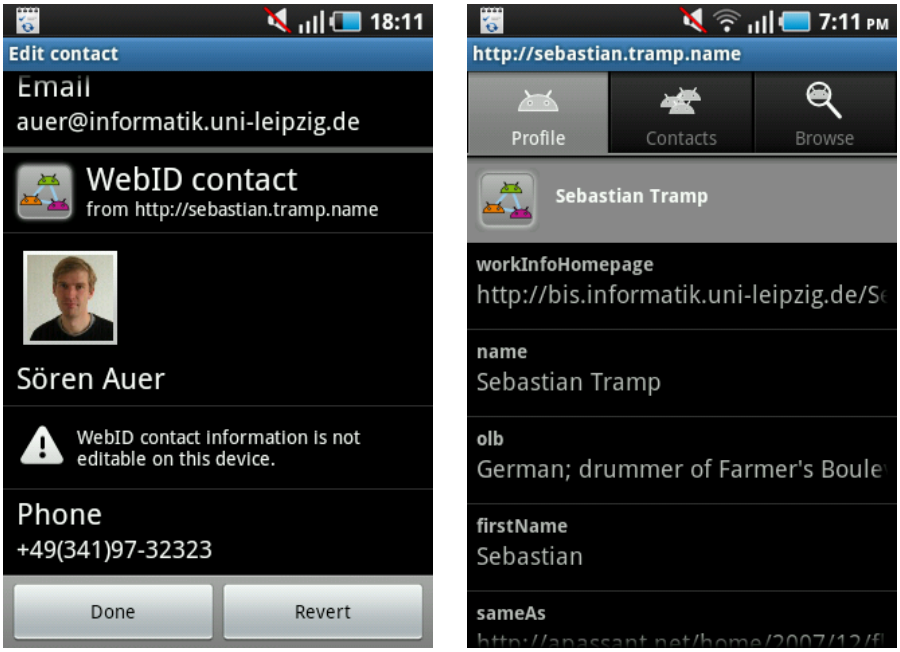

Figure 23: Screenshots of the Mobile Social Semantic Web Client: left – A contacts profile page merges the data from all given accounts; right – By using the FOAF browser, people can add contacts or browse the contacts of their friends.

After the user associated his profile with his WebID, information from linked WebIDs of the users contacts are synchronized regularly and the information are made available via the Android content provider to all applications on the device. During the import of the WebID contacts, they are merged based on the assumption of unique names. Independent of this automatic merge, the user can split and merge contacts manually in the edit view of these contacts. The left screenshot in Figure 23 shows the standard Android contact application, where our WebID content provider seamlessly integrates information obtained from WebIDs. Information obtained from WebIDs is not editable, since it is retrieved from the authoritative sources, i.e. the WebIDs of the respective contacts.

The right screenshot in Figure 23 shows the FOAF browser, allowing people to add contacts or to browse the contacts of their friends. In order to facilitate the process of connecting with new contacts the

Android implementation also allows to scan QR-codes of WebIDs (e.g. from business cards) and to search for WebIDs using Sindice.

8.3 RELATED WORK

Related work can be roughly divided into mobile Semantic Web projects and mobile social network clients. A comprehensive overview is contained in the final report of the W3C Social Web Incubator Group [Halpin and Tuffield, 2010].

Mobile Semantic Web applications

The application of Semantic Web technologies on mobile devices is not new – one of the earlier works dates back to 2003 [Lassila and Adler, 2003]. However, this research area was not pursued very actively due to the large number of mobile device limitations common in that era. In the light of increasing processing power and data connectivity of modern mobile devices, the use of mobile Semantic Web technologies is becoming more feasible. There are a few publications which review the state of the mobile Semantic Web and the possibilities thereof to improve mobile Web (such as Lassila [2005]). Also, there are publications on more complex systems, such as *SmartWeb* [Sonntag et al., 2007] which uses semantic technologies to enhance the backends of mobile web service. On the frontend side there are semantic mobile applications like *DBpedia Mobile* [Becker and Bizer, 2009] or *mSpace Mobile* [Wilson et al., 2005] that implement or utilize Semantic Web technologies directly on mobile devices. However, these frontend applications focus on very specific use cases – information about points of interest in the case of DBpedia Mobile and information for university students in the case of mSpace. Finally, there are publications which describe proof of concept applications as well as algorithms for consuming and replicating Linked Data and RDF in general (David and Euzenat [2010], Le-Phuoc et al. [2010] as well as Schandl and Zander [2009]).

Mobile social networking clients

All major social networking services (such as Facebook and LinkedIn[100]) have meanwhile clients for different mobile platforms, which are more or less integrated with the mobile phone platform itself. In addition to this, the Android market place lists more than 3500 applications in the category social networks with our implementation being one of them. However, up to our knowledge our MSSW client is the first to

100 An ordered list can obtained from Wikipedia: http://en.wikipedia.org/wiki/List_of_social_networking_websites

consequently employ W3C standards as well as Social Semantic Web best practices with regard to all aspects of data representation and integration.

9 ONTOWIKI – A DATA WIKI WITH INTEGRATED DSSN CAPABILITIES

The results presented in this chapter were primarily published in Tramp et al. [2014], Dietzold and Auer [2009] as well as in Tramp et al. [2010b].

In this chapter we give an overview to the web application OntoWiki and how we extended as well as integrated it into the proposed DSSN architecture.

OntoWiki is a semantic data wiki. We refer to it as a *Wiki*, since the focus of the application is on simplicity, adaptability and collaboration. However, other than annotating text-based Wiki pages with a special syntax (as suggested by text-based Semantic Wiki approaches), OntoWiki uses RDF in the first place to represent information. For human users, OntoWiki allows to create different views on data, such as tabular representations or maps. For machine consumption it supports various RDF serialisations as well as RDFa, Linked Data and SPARQL interfaces. Since its introduction in Auer et al. [2006], the application has evolved into a framework for building Semantic Web applications [Heino et al., 2009] and was recently updated to support the collaboration across multiple domains and application via Semantic Pingback [Tramp et al., 2010a] and RDFauthor [Tramp et al., 2010d]. After a general introduction, we go into detail with its latest extension in order to integrate this application into the DSSN architecture.

9.1 FEATURE INTRODUCTION

OntoWiki is a Semantic data wiki as well as Linked Data publishing engine. It can be used to author, manage and publish RDF-based knowledge bases. Instead of presenting architecture and design issues, we concentrate just on the core features here to provide an overview of the capabilities of the application.

9.1.1 *Navigation and Visualisation*

OntoWiki provides a number of ways for navigating through RDF knowledge bases. This includes *taxonomy and hierarchy browsing*, such as SKOS taxonomies [Miles and Bechhofer, 2009], the class hierarchy of RDF Schema or an organizational hierarchy [Reynolds, 2014], *facet-based browsing* (with *complex filter* conditions and *attribute based tag clouds*) and *full-text search*.

These different navigation facilities can be used as a *combined navigation*, for example a user can start with a full-text search and *refine the results* using faceted-search or restrict them to a certain part of

the selected hierarchy. OntoWiki translates these different navigation features into a single SPARQL query.

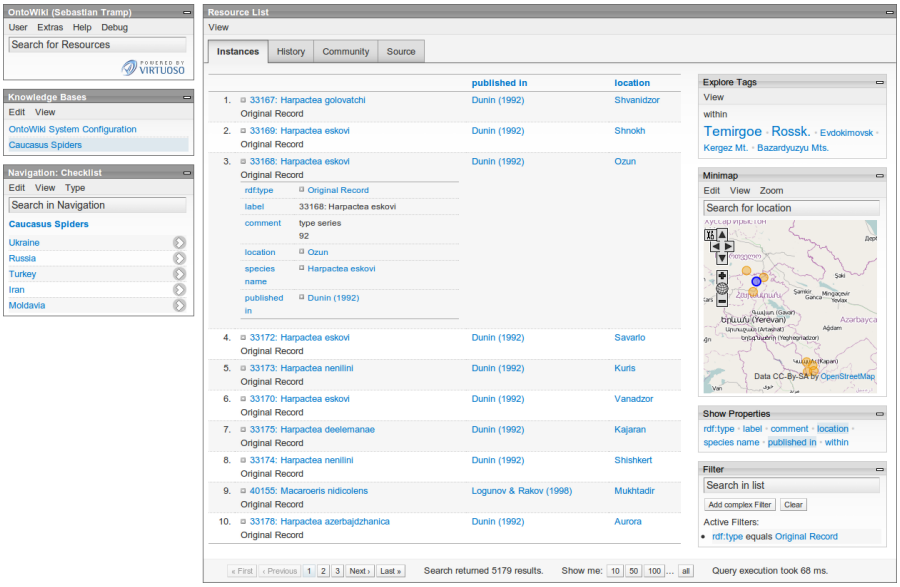

Figure 24: Generic OntoWiki view – resource list with two selected properties, an applied filter, map display and an attribute cloud.

In addition to rendering the results of such a query as a *resource list*, OntoWiki provides an *extension mechanism* for implementing dedicated views such as *maps and calendars* (Figure 24).

Once a specific entity has been selected, the *resource view* shows all available information for that entity (Figure 25). Besides rendering this information generically in a tabular way, OntoWiki can be extended with *domain specific resource views* (i.e. vocabulary based). Specific resource views for the SKOS and FOAF vocabularies are already included.

9.1.2 *Authoring*

All views in OntoWiki can be equipped with corresponding *RDFa annotations*. OntoWiki employs the *RDFauthor mechanism* [Tramp et al., 2010c] to automatically transform these views into *editable forms*. As a consequence, all information items displayed in OntoWiki can be directly edited in place. This allows OntoWiki users to edit the knowledge base even without being acquainted with the RDF, RDF-Schema or OWL data models. Following the Wiki design principles [Leuf and Cunningham, 2001] all changes are put under *version control* and can be easily *rolled back*.

In addition to this, the integration of RDFauthor as our main authoring mechanism led to two additional usage options:

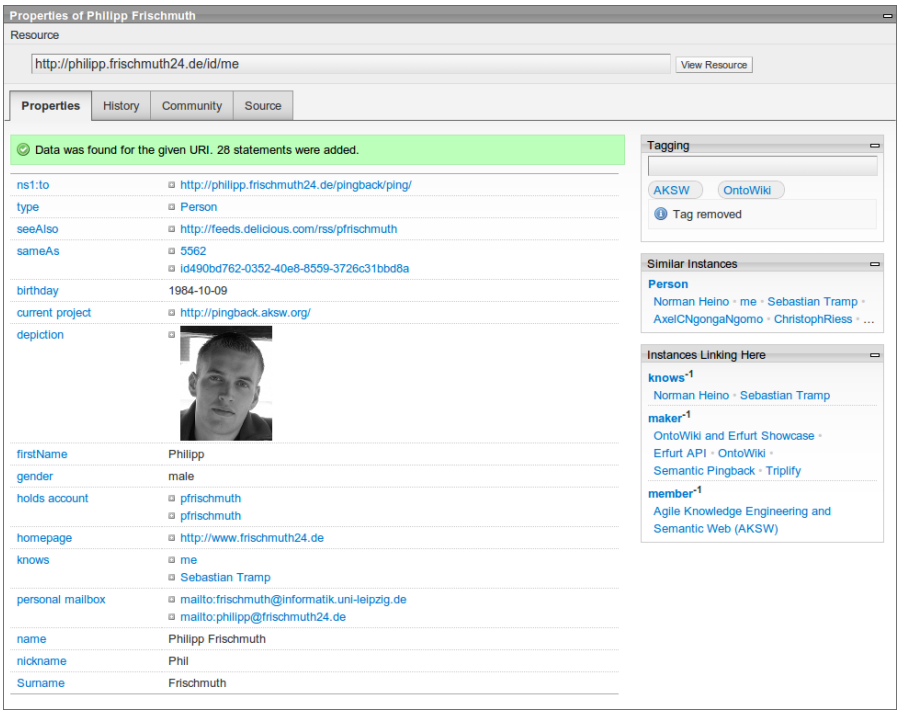

Figure 25: Generic OntoWiki view – resource view with similar instances, linking instances as well as a module for tagging.

- By using OntoWiki's RDFauthor *bookmarklet*, users can *collect data from different web pages* and import it directly into their personal knowledge bases (such as contacts or events).
- OntoWiki can be used as a service for hosting *editable mash-ups*. These mash-ups use data from OntoWiki's *SPARQL endpoint* and other data sources (which are interpreted as named graphs) and provide a merged view on this data. The bookmarklet is able to distinguish the statements from these named graphs and allow the user to edit these data directly inside the mash-up as well as to *propagate the changes back to the different sources* [Tramp et al., 2010c].

Knowledge Base Evolution

OntoWiki tries to support the creation of RDF *Knowledge Bases from the scratch* (rather than using predefined ontologies only) in a wiki way. This means that our users often follow the learning by doing path, where requirements, priorities and understanding of the domain change over time. Our *evolution framework* allow OntoWiki users to apply predefined *evolution pattern* from a *pattern repository* to their data, as well as to create their own pattern in order *maintain the data quality* of their Knowledge Base with little effort. Examples for such evolution patterns include the *splitting of classes by using their property*

values or the *transition from a data property to an object property*, such as the transition from literal based tagging to resource based tagging.

9.1.3 *Linked Data*

OntoWiki acts as a *Linked Data server* as well as an *Linked Data client*. Entities that use identifiers being governed by the OntoWiki installation are automatically served as Linked Data. OntoWiki supports *access control* on Linked Data employing the *WebID protocol* as well as traditional local account based authentication. References to entities using foreign identifiers can be de-referenced (i.e. consumed by OntoWiki) in order to *obtain additional information* from the original source. Even non-linked data resources can be consumed by utilising *custom data-wrapper*, that allow the *generation of RDF data* (e.g. from images, videos or web services).

9.2 DSSN IMPLEMENTATION FOR ONTOWIKI

As described in the last section, OntoWiki is already well connected in a Linked Data infrastructure. We took this application as a starting point for the development of a DSSN client. From the wiki point of view we specialized the application with specific view for important types of Social Network activities.

We provide a prototypical implementation of our approach. All features were realized by employing the extension mechanisms offered by OntoWiki. One main backend feature of OntoWiki is its storage layer independence. This means that an OntoWiki setup can use a high-performance RDF triple store (such as Openlink Virtuoso described Erling [2012]) for knowledge bases up to the size of the DBpedia project [Bizer et al., 2009] as well as a MySQL backend with a SPARQL2SQL query rewriter for small and mid-size knowledge bases. We tested and used the DSSN implementation with both backends.

The prototype described here, can be summarized as a WebID provider with an integrated communication hub. The following features are implemented so far:

- Users can create and manage their WebID profiles and any other Linked Data enabled resource (Section 9.2.1).
- Users can make friends, subscribe to their activities and profile updates and receive changes instantly on change (Section 9.2.2).
- Users can search and browse for friends and activities inside their social network as well as filter these resources by facets based on object and datatype properties (Section 9.2.3).

- Users can comment on and subscribe to any DSSN resource which is equipped with a Semantic Pingback service or a PubSubHubbub-enabled activity stream (Section 9.2.4).
- Users receive notifications if someone comments or links her WebID and send a pingback notification (Section 9.2.5).

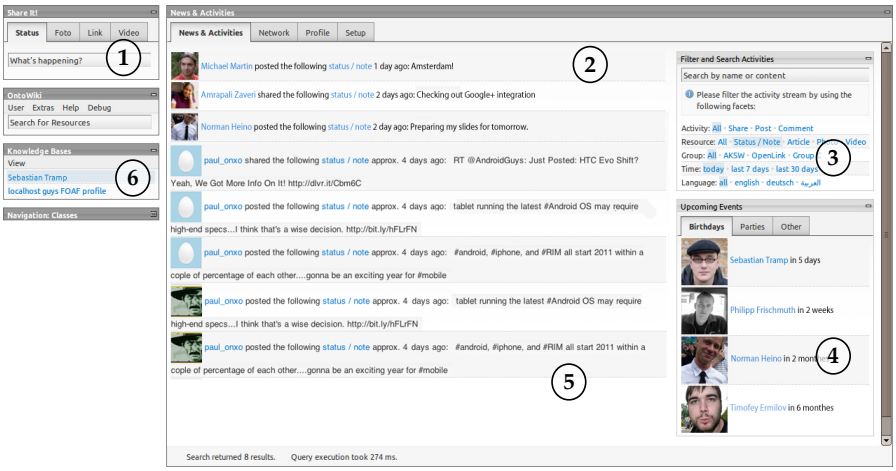

Figure 26: Screenshot of the OntoWiki DSSN activity stream view.

We now describe the implementation of these features and provide insights into our rationale for choosing certain technologies. A screenshot of the central activity stream interface can be seen on Figure 26.

The following interface elements are visible:

1. The *Share it!* activity creation module where users can post status notes and share links and media artefacts.
2. The main interface view tab to switch between configuration screen, profile manager, friending interface and the activity stream.
3. The activity filter and search module, to allow a facet-based browsing of activities.
4. The events module, which queries the cache of social network data for birthdays and other events.
5. The activity stream, which is the result of the SPARQL query modified by the filter module.
6. The generic wiki interfaces to create any type of Linked Data resource (e.g. comments).

9.2.1 *Creating and Updating Data Artefacts*

Creating and managing Linked Data enabled RDF resources can be achieved without modifying the OntoWiki basic functionality. We em-

ploy the RDFauthor [Tramp et al., 2010d] JavaScript widget library in order to automatically create forms out of RDFa annotated HTML documents[101]. Without any user effort, a newly created resource will be Linked Data enabled if it shares the namespace of the OntoWiki installation. In addition to that, we added support for WebID authentication as well as for certificate creation by implementing a dedicated WebID extension. Since there is quite some cryptographic processes involved, this extension needs some prior configuration steps, e.g. the OntoWiki instance needs to be configured as a SSL/TLS enabled Web application.

9.2.2 *Maintaining Social Network Connections*

In order to maintain friend connections and other social network connections we execute the following steps in our implementation:

1. When a user enters a WebID inside the friending module, OntoWiki adds a new statement (`foaf:knows`) into the RDF graph containing the users profile data. This automatically generates a Pingback request so the new friend will be notified.
2. The corresponding OntoWiki creates a new RDF graph, which will act as a cache for the WebID profile data about that particular friend. This new Knowledge Base is configured in such a way, that it is imported into the users graph automatically.
3. The wiki fetches the data from the friends WebID by employing the Linked Data principles. The newly generated graph caches that data in order to enhance the performance. Since those information is stored in a separate graph, a synchronization is trivial.
4. Finally, the wiki is subscribed to these feeds if they are published within the users profile:
   a) The users activity feed, which the wiki uses to create the network activity timeline for the user. The incoming atom activity entries are transformed to AAIR resources and imported to an additional activity RDF graph.
   b) The history feed for the friends' profile. This feed is employed in order to keep the cached WebID profile data up-to-date.

9.2.3 *Ignoring Activities and WebIDs*

Since a user may not be interested in all activities of her friends (e.g. gaming activities), we offer functionality to hide certain activities in

101 Each output page, which is created by OntoWiki, is RDFa enhanced.

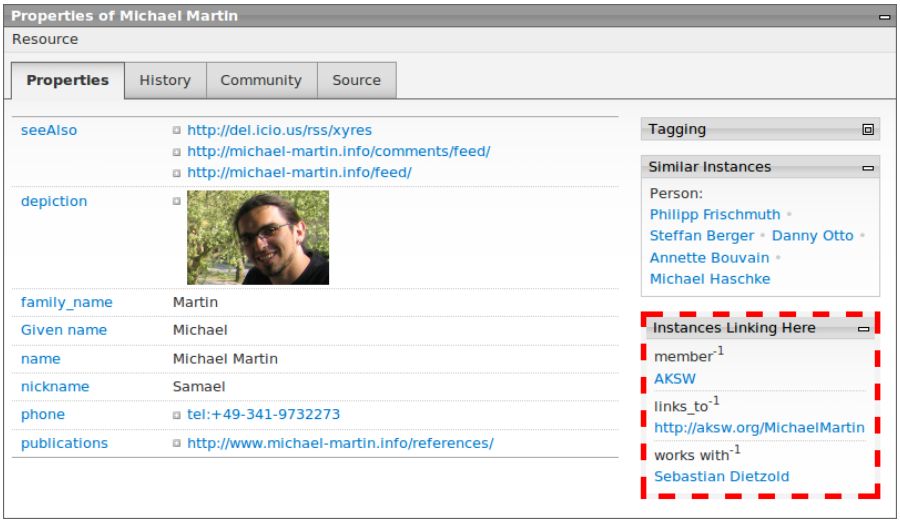

Figure 27: Semantic Pingback Integration – Backlinks that were established via the Pingback service are displayed in the standard OntoWiki user interface.

the timelines that visualize the activity streams to which a user is subscribed. We therefore employ EvoPat [Rieß et al., 2010], a pattern-based approach for the evolution and refactoring of RDF knowledge bases. EvoPat is integrated also as an OntoWiki extension and in conjunction with our DSSN implementation we apply this component to allow users to clean up their timelines.

Each time a user selects a *Hide all activities ...* button next to an activity, we create a new pattern, which subsequently matches such statements in the graph and removes them. Possible hide patterns are generated from the activity data itself, for instance *...from this user*, *...of this language* or *...about this object*. Once new data for a given user is added, we re-apply the pattern. In this way it also applies for future changes of the knowledge base.

9.2.4 *Generating and Distributing Activities*

A user is able to generate different kinds of activities in our implementation. In the current state we support three types of activities: status updates, photo sharing and recommendations. We implemented a small extension, which displays a *Share It!* module inside the OntoWiki user interface. As a result the user can quickly access this functionality so that sharing is facilitated for the user. Once the user generates an activity, her activity feed is updated and subscribers to that feed are delivered with the new content by the push service.

9.2.5 *Pingback Integration*

All activities as well as every resource created by the user in an accessible namespace of the application are represented as Linked Data resources that refer to a Pingback service and a corresponding activity feed. Thus, users can comment on any resource in their own social application and additionally subscribe to changes to that resource (e.g. comments by others).

Each time someone comments on a resource (or otherwise links to it on the Linked Data Web), a Pingback request is sent to the owning OntoWiki instance. Consequently the publisher gets notified and is able to react again on that new activity, which facilitates conversations in a distributed manner (Figure 27). If a user is subscribed to a resource activity feed (which is automatically done, once she comments on a particular resource), she gets notified about other comments, even if she is not the commenting person or the owner of the resource.

Part IV

EVALUATION, CONCLUSION AND FUTURE WORK

In this part of the thesis, we evaluate the proposed architecture in a different way than providing prototypes such as the application in Part iii. Instead, we simulate a DSSN based on real data from the well known Twitter service and measure triple distribution as well as query performance for different DSSN node types. Finally, we conclude this thesis and provide some directions for future work.

10 EVALUATION

The results presented in this chapter were primarily published in Tramp et al. [2014].

In this Chapter, we present our work in regards to the evaluation of the proposed DSSN architecture.

We divided our evaluation process into two independent parts: Firstly, the qualitative evaluation part aims to prove the functionality of the DSSN architecture by assessing use cases from the Social Web acid test (Section 10.1). Secondly, the quantitative evaluation part aims to prove the real-world usefulness of our prototypical implementation by testing the performance and distribution of data in the social network (Section 10.2). This evaluation is carried out by using a social network simulation approach.

10.1 QUALITATIVE EVALUATION: SOCIAL WEB ACID TEST

The Social Web Acid Test (SWAT) is an integration use case test conceived by the Federated Social Web Incubator Group of the W3C. Currently, only the first and very basic level of the test (SWAT0[102]) has been developed and described completely. Nevertheless, those parts of the next level (SWAT1) which are currently published are discussed here too.

10.1.1 *Social Web Acid Test – Level 0*

The objectives of the first SWAT level have been specified in the following use case[103]:

```
1 User A takes a photo of user B from her phone and posts it.
2 User A explicitly tags the photo with user B.
3 User B gets notified that she is in a photo.
4 User C who follows user A gets the photo.
5 User C leaves a comment on the photo.
6 User A and user B get notified about the comment.
```

Listing 10: Social Web Acid Test – Level 0

Utilizing all technologies described before, our DSSN architecture passes the SWAT0 without any problems. The following enumeration describes the corresponding steps. Each step is mapped in Figure 28.

102 http://www.w3.org/2005/Incubator/federatedsocialweb/wiki/SWAT0

103 For this use case the following assumptions are made: (1) Users employ at least two (ideally, three) different services each of which is built with a different code base. (2) Users only need to have one account on the specific service of their choice. (3) Ideally, participants A, B, and C use their own sites (personal URLs).

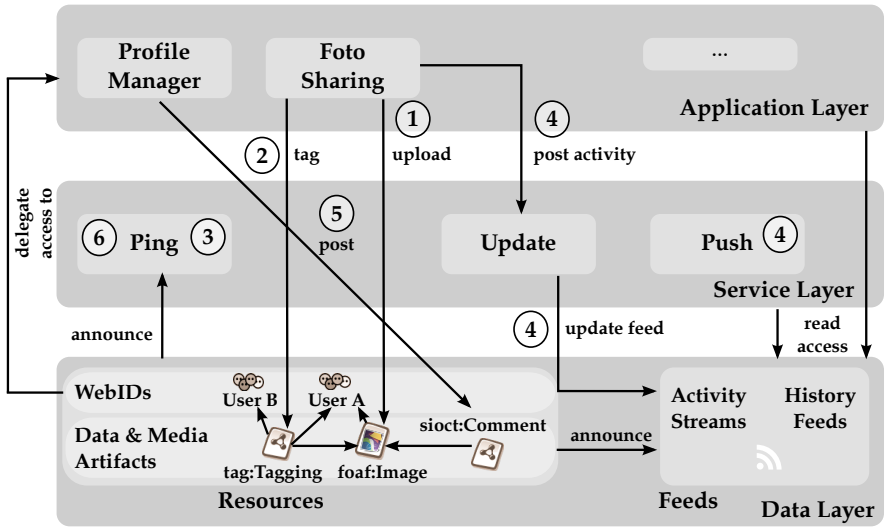

Figure 28: Social Web Acid Test Level 0 – Solution based on the proposed DSSN architecture

1. *User A takes a photo of user B and uploads it:* The photo upload application creates a new media artifact including an RDF description. This RDF description links to the creators WebID as well as to the Pingback Server. Optionally, the Web space returns a link to the user's pingback server in the HTTP header of the uploaded image. In addition to that, the RDF description has a link to the WebID of *User A* as the creator of the image resource. A possible minimal RDF description is listed in Listing 11.

```
ex:Image a foaf:Image ;
  foaf:name "Me and UserB at the party ..." ;
  foaf:maker ex:UserA ;
  cc:license <http://creativecommons.org/licenses/by/3.0/> ;
  ping:to <http://pingback.aksw.org/> .
```

Listing 11: SWATo: Image description with autodiscovery links and license (the `foaf:Image` media artifact in Figure 28).

2. *User A explicitly tags the photo with user B:* This is done by creating a tagging resource (`tag:Tagging`) which links to the image (`tag:taggedResource`), to the creators WebID (`tag:taggedBy`) as well as to the WebID of user B (`tag:associatedTag`).

   A pingback client sends a ping request to all linked resources after publishing the tagging on the Web.

3. *User B is notified that she is on a photo:* The notification is created by the pingback service of the WebID of *User B*. The pingback service has received the request from the tagging application (in this case, the photo sharing tool) which was used by *User A*. Depending on *User B*s setup, he will receive an email or just

```
ex:Tagging a tag:Tagging;
  tag:taggedResource ex:Image;
  tag:taggedBy ex:UserA ;
  tag:associatedTag ex:UserB ;
  cc:license <http://creativecommons.org/licenses/by/3.0/> ;
  ping:to <http://pingback.aksw.org/> .
```

Listing 12: SWATo: Tagging description (the `tag:Tagging` data artifact in Figure 28).

get a new entry in his social network client such as depicted in Figure 27 (Page 97).

4. *User C, who follows user A, receives the photo:* User C is instantly provided with an update in her activity stream, informing her about the new image. The post / upload as well as the tagging activity result in new feed entries in the activity stream of the *User A*. In case of a full featured social network client such as described in Chapter 9, the activity can simply be written to the triple store which feeds the activity stream. In the SWAT example, the photo sharing application is different from the main profile management application. This means the photo sharing application needs to use the *update service* of the WebID of *User A*. This is done by using the *access delegation* extension to the WebID authentication protocol which is presented in Chapter 6. The photo sharing application acts as a *secretary* of *User A* and insert a new activity description to the profile triple store. This example activity is listed in Listing 13.

```
[] a aair:Activity;
  atom:published "2014-01-19T08:04:14"^^xsd:dateTime;
  rdfs:label "User A posted an image." ;
  aair:activityActor ex:UserA ;
  aair:activityVerb aair:Post;
  aair:activityObject ex:Image .
```

Listing 13: SWATo: Post activity description.

5. *User C leaves a comment on the photo:* This is done in the same way as publishing the tagging resource. The SWAT use case does not specify, which application should publish the comment about *User A*s photo. In a distributed network of application, any application could be used – our example has at least two candidates: the photo sharing application and *User C*s profile management application. We assume that in this case the profile management application is used since *User C* receives the photo post activity via push notification here. If she wants to comment on the photo now, her profile management application needs to publish a comment resource in its own namespace and send pings to all linked resources. Listing 14 depicts a valid comment

which is linked to all important stakeholder of the SWAT 0 use case.

```
ex:Comment a sioct:Comment ;
  sioc:reply_of ex:Image ;
  foaf:maker ex:UserC ;
  sioc:conent "nice! :-)" ;
  cc:license <http://creativecommons.org/licenses/by/3.0/> ;
  ping:to <http://pingback.aksw.org/> .
```

Listing 14: SWATo: Comment description posted by *User C* (the `sioct:Comment` data artifact in Figure 28).

6. *User A and user B are notified about the comment: User A* is the publisher of the image. She will be notified because her pingback service informs her about this comment. *User B* will be notified only if she has subscribed to the activity feed of the photo resource. Such an subscription could be created automatically based on the fact that *User B* was tagged on the photo. However, this is an optional step for *User B*.

10.1.2 *Social Web Acid Test – Level 1*

SWAT level is currently not finally defined[104], so an evaluation can be a rough sketch only. The next SWAT level will require a few different use cases which introduce some new Social Web concepts. However, most of the *user stories* are satisfied already as a consequence of the fully distributed nature of the DSSN architecture (e.g. data portability and social discovery). The more interesting user stories are:

1. The *Private content* and *Groups* use cases will require a distributed ACL management. Some ideas on using WebIDs for group ACL management are already published with dgFOAF in Schwagereit et al. [2010] and we think that this is a good starting point for further research.

2. The *Social News* use case introduces a new vote activity. Since our architecture applies schema agnostic social network protocols, this new type of activity can be communicated as any other activity.

Our next aim was to evaluate the architecture in quantitative terms.

10.2 QUANTITATIVE EVALUATION: DSSN PERFORMANCE

Based on the proposed architecture, a DSSN will be distributed over hundreds of servers. These social network nodes will have hardware

104 Available online at http://www.w3.org/2005/Incubator/federatedsocialweb/wiki/SWAT1_use_cases (received 29.07.2011).

specifications which can range from very light-weight (e.g. plug computers as proposed by the FreedomBox[105] project, smartphones and small virtual hosts) to medium and heavy class systems (e.g. cloud instances, hosted services and full root servers). Each of these nodes accesses only a small part of the complete social network graph since the information is shared only with the connected nodes.

Consequently, we need to pose the following questions in our quantitative evaluation:

1. How many incoming triples need to be cached from an averagely connected node in a week of average social network activity?
2. If a DSSN node queries these incoming triples with SPARQL, are the queries fast enough to provide the data for the user interface on such weak hardware?

To answer these research questions, we created an evaluation framework that allows for simulating the traffic within a DSSN. We apply this framework to measure the performance of our DSSN in a testbed using a large social network dataset.

10.2.1 *Evaluation Framework Architecture*

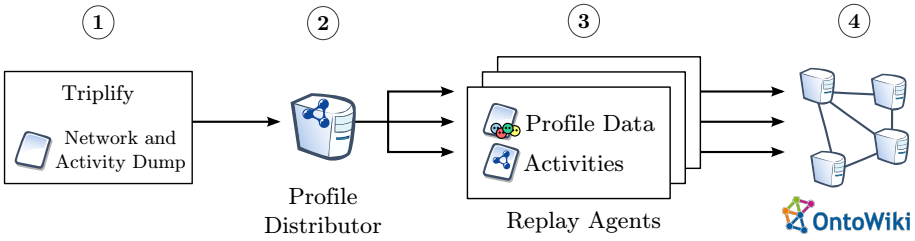

Figure 29: Workflow of the social network evaluation framework: (1) An activity and profile graph is preprocessed and dumped as RDF. (2) The knowledge base is split up into a part for every single profile and can be distributed to a list of server machines. (3) For each profile, a *replay agent* is initialized which will create and update a specific profile as well as activities on a single social network node. (4) The social network node (in our case an OntoWiki instance) connects to other nodes as well as creates activities for the social network in the same way as it would be achieved under control of a human user.

We decided to use a simulation approach in which activities are created artificially (but based on real data, see Section 10.2.2) on the social network rather than arranging a user evaluation, which strongly depends on the graphical user interface. In a first step, we added an additional service for remote procedure calls to the OntoWiki DSSN

105 http://www.freedomboxfoundation.org/

node as well as an execution client (replay agent) which can inject activities remotely controlled and based on input activity data in RDF. Then we used the public Twitter dataset described in Abel et al. [2011] and transformed it to an RDF graph as a base for a replay of activities of type status note.

The workflow of the evaluation framework is depicted in Figure 29. It has a pipeline architecture which is built by using three additional tools that help us to generate and manage the generated replay agent test data. The generated data was processed by the profile distributor which splits the data into separate user profiles, each with personal data and activities. After splitting the data into single parts, the profile distributor passes each part on to a corresponding replay agent. Upon call the replay agent can instantiate a new OntoWiki-based DSSN node with the given user profile and activities.

10.2.2 *Data Generation and Testbed Configuration*

In a first step, we used Triplify [Auer et al., 2009] to generate the activity data from the relational database. The database consists of 2.3 Mio public tweets fetched from 1701 Twitter accounts over a time frame of two months. For each posted tweet, we created a status note resource description and added two SIOC properties to link the creator account and publish the creation timestamp. In addition to that, we linked each status note to the Twitter terms of services to demonstrate the usage of licensing in our architecture (the data ownership issue from the Introduction)[106]. Listing 15 shows an example status note resource taken from the database[107]

```
:o5516682621620224 a aair:Note;
  rdfs:seeAlso <http://twitter.com/#!/youngglobal/status/5516682621620224>;
  sioc:created_at "2010-11-19T08:04:14"^^xsd:dateTime;
  sioc:has_creator :youngglobal;
  dct:license <http://twitter.com/tos>;
  aair:content "I'm at Norwood (241 W 14th St, btw 7th & 8th, New York) w/ 5
       others. http://4sq.com/66OndN".

:a5516682621620224 a aair:Activity;
  atom:published "2010-11-19T08:04:14"^^xsd:dateTime;
  aair:activityActor :youngglobalPerson;
  aair:activityVerb aair:Post;
  aair:activityObject :o5516682621620224.
```

Listing 15: Example status note resource and corresponding activity.

For the creation of the corresponding activities we intepreted retweets as sharing and (original) tweets as posting activities and assigned different activity verbs based on the data. Each activity is linked

106 Licensing statements can be easily added to any resource in the DSSN, e.g. by using the `dct:license` property.

107 The Listings in this section uses several prefixes which are explained at Page xviii.

to a FOAF person resource (without WebID specific enhancements) which we additionally enriched with a random `foaf:birthday`.

```
:youngglobal a sioc:UserAccount;
  sioc:name "youngglobal";
  sioc:account_of :youngglobalPerson;
  rdfs:seeAlso <http://twitter.com/youngglobal>.

:youngglobalPerson a foaf:Person;
  foaf:name "youngglobal";
  foaf:knows :RdubuchePerson;
  foaf:birthday "01-31";
  foaf:depiction <http://a2.twimg.com/profile_images/1152004614/
      pip_2825_0370_normal.jpg>;
  foaf:account :youngglobal.
```

Listing 16: Example user account and FOAF person resource.

Listing 16 shows the user account and FOAF person resource which is linked from the activity in Listing 15.

In order to evaluate the query performance on different types of DSSN nodes in our architecture, we extracted four exemplary queries, which are crucial for rendering the user interface depicted in Figure 26.

Query Q1 asks for an ordered list of the last ten status posts of a given time frame. The query is exemplary for fetching a resource list based on a given user defined configuration. The query is used for area 2 on Figure 26 and is typically followed by a query which fetches the needed data of exactly these ten activities (rather than doing both in one single query).

```
SELECT DISTINCT ?r
WHERE {
  ?r a aair:Activity.
  ?r atom:published ?pub.
  ?r aair:activityObject ?aairObject.
  ?aairObject a aair:Note.
  FILTER
    (?pub >= "2010-12-01T00:00:00"^^xsd:dateTime)
  FILTER
    (?pub <= "2010-12-01T23:59:59"^^xsd:dateTime)
}
ORDER BY ?pub
LIMIT 10
```

Listing 17: Ordered list of the last ten status posts (Q1).

Query Q2 is used to build the facet-based exploration module depicted in area 3 on Figure 26. It asks for all values of a specific exploration facet of the activities of a given time frame. The query that is depicted in Listing 9 asks for the used verbs in the selected set of activities (possible verbs are post, share, comment etc.).

```
SELECT DISTINCT ?verb
WHERE {
  ?r a aair:Activity.
  ?r atom:published ?pub.
  ?r aair:activityObject ?aairObject.
  ?aairObject a aair:Note.
  ?r aair:activityVerb ?verb.
  FILTER
    (?pub >= "2010-12-01T00:00:00"^^xsd:dateTime)
  FILTER
    (?pub <= "2010-12-01T23:59:59"^^xsd:dateTime)
}}
```

Listing 18: A list of verbs connected to a list of activities (Q2).

Query Q3 is used to fetch the list of the next five upcoming birthdays together with the associated person. Since `foaf:birthday` values are of datatype `xsd:string`, a string comparison has to be executed. Query 3 (see Listing 19) is used for area 4 on Figure 26.

```
SELECT DISTINCT ?person ?bday
WHERE {
  ?person a foaf:Person.
  ?person foaf:birthday ?bday.
  FILTER (xsd:string(?bday) >= xsd:string("01-29"))
}
ORDER BY ASC(?bday)
LIMIT 5
```

Listing 19: List the next five upcoming birthdays (Q3).

Query Q4 is applied to prepare a human readable label for all the resources which are currently visible in the interface. This includes schema resource as well as instance data. The query uses a list of resources (line 5) and a list of possible label attributes (line 6) and fetches them in a vertical result set (e.g. with a minimal amount of projection variables). The client receives the data and has to select a value based on an ordered internal list (e.g. a `foaf:name` value is preferred over an `rdfs:label` value, because the latter is more general). This query strategy is especially useful in combination with incomplete data (e.g. use the `foaf:nick` if you do not have a `foaf:name`).

Since one of the ideas of a distributed social network is the usage of low-end hardware, which everyone can afford or which already exists in most households (e.g. DSL router or WLAN access points), we defined three prototypical categories of DSSN nodes where for which we would like to test the query performance:

A *server* is a typical host in a computing center which can be used for a rental fee per month. Privacy is moderatly preserved on such a system since the computing center staff can access the system. This category of DSSN node is used mostly by people with a strong tech-

```
SELECT DISTINCT ?s ?p ?o
FROM <http://aksw.org/>
WHERE {
OPTIONAL {?s ?p ?o.}
FILTER(sameTerm(?s, <...>) || sameTerm(?s, <...>) || ... )
FILTER(sameTerm(?p, skos:prefLabel) || sameTerm(?p, dc:title) || sameTerm(?p, dct:
    title) || sameTerm(?p, foaf:name) || sameTerm(?p, aair:name) || sameTerm(?p,
    sioc:name) || sameTerm(?p, rdfs:label) || sameTerm(?p, foaf:accountName) ||
    sameTerm(?p, foaf:nick) || sameTerm(?p, foaf:surname) || sameTerm(?p, skos:
    altLabel))}
```

Listing 20: Ask for all known title attributes for a given list of resources (Q4).

nical background. In this category, we tested a Virtuoso 6.1.4 backed OntoWiki DSSN node on a dual core 2.4GHz system, with 4GB of RAM and an SSD.

A *FreedomBox* is a personal server running on a low-end system in an area where the users privacy can be preserved (e.g. as a DSL router in his household). No-one else has access to the system which runs 24 hours a day in the same way a server does. In this category, we used a virtual machine with 1GB of memory, one core and a 25% CPU limitation from the server system above. In addition to that, we limited the triple store process to 300MB of RAM.

Smartphones are very important for social network activities today, but they are used mostly as a thin client without a backend. We argue that connection stability and battery issues will be solved in the near future and smartphones can be used as first class DSSN nodes. In this category, we used an in-browser JavaScript API store based on rdfQuery[108] and deployed the data as well as the triple store without a frontend on an iPhone 4S.

10.2.3 *Results and Discussion*

As described in Section 10.2.2, the testbed consists of 1701 DSSN node profiles with 2.3 Mio activities. We first looked, how this data was shared over these DSSN nodes to overview what amount of data these nodes have to store. Regarding this, two characteristic indices are important:

1. the number of activities of an account and
2. the number of related accounts which will receive these activities.

Figure 30 shows a scatter plot where each account corresponds to one point. The x axis represents the number of `foaf:knows` relations to other persons from the testbed network and the y axis depicts the amount of triples which are produced with the node's frontend (profile triples and activity triples).

108 https://github.com/alohaeditor/rdfQuery

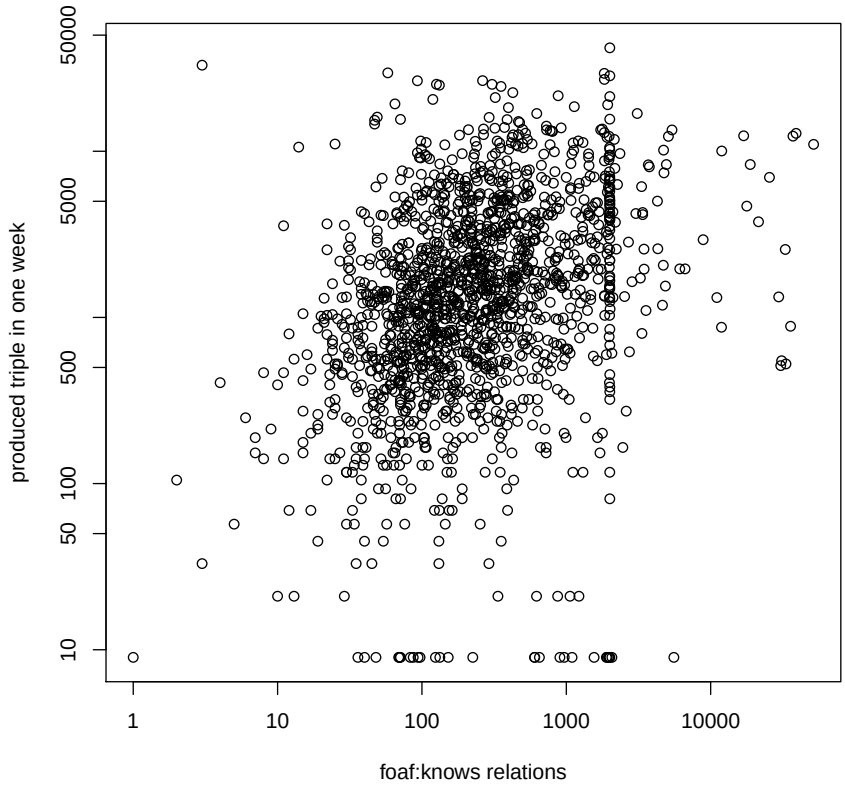

Figure 30: Scatter plot with logarithmic axes of related accounts vs. the number of created outgoing triples after one week of social network activity (taken from Abel et al. [2011], uncleaned).

Given this plot, we assume that the amount of friendship relations of an account and the amount of activities of an account do not correlate with each other. Since the given raw data included extreme values not suitable for our approach (see below), we cleaned the data by eliminating outliers in both dimensions. The average size of outgoing triples for all profiles after one week of activity is 1589 triples. The average amount of related contacts in the cleaned data is 225. We used these values as an artificial point in the graph and identified one single account which had the smallest distance to this point. This account was used for the query evaluation.

The motivation behind this approach is to evaluate the average performance of a DSSN node. Morsey et al. [2011] analyzed the correlation between knowledge base size and query performance. This setup simplifies these results and assumes a linear correlation which is acceptable in the context of this evaluation.

The evaluated account has 212 `foaf:knows` relations with other accounts, shared 113 foreign notes and posted 5 original notes in one

week. With this activities, the owner produced 1425 triples (including his profile). In addition to that, he received 396346 triples from his friends over Linked Data and PubSubHubbub.

We used this graph and executed the introduced queries on the three different systems, described in Section 10.2.2. Table 4 shows the average execution time in milliseconds after 5 runs for each query.

Table 4: Runtime (in ms) of different evaluation queries

Query	Server	FreedomBox	Smartphone
Q1 (posts)	35	71	41325
Q2 (facets)	150	312	37558
Q3 (birthdays)	29	36	11324
Q4 (titles)	522	2205	n/a

The results show, that querying this data is possible at least on real triple stores and with a moderate time frame. Using a DSSN node on a dedicated server should even work with much more data. The results from the smartphone demonstrate a gap of working triple store implementations for HTML5 applications.

Query 1–3 are similar in its structure since they combine at least two graph patterns with one or two filter. Query 2 is the slowest here since it uses graph pattern most which results in a query plan with many joins. Query 1 uses an `ORDER` clause on a big result set (all status notes timestamps). This may be the reason for being the slowest query on the smartphone.

Query 4 is hardest query in our experiment, since it does not use any graph pattern which can be used for filtering on index level. Instead it uses two complex filter which have to be evaluated on a lot of triple. Given these results, we will consider a different query strategy for querying the title attributes. One option is to send a query for each resource thus allow to restrict the index based on the subject.

The huge amount of data which is received by DSSN nodes in a realistic environment should be controlled not only by faster triple stores and more hardware, but also by implementing smart interfaces which cache at the right places and pre-calculate many interface elements fostering a faster user experience.

As a nice spin-off, this evaluation demonstrates how social network federation can be achieved if semantic interoperability is guaranteed. If a user wants to federate her DSSN node with other social networks, a semantic agent can easily fetch and write data from social network APIs, translate it to RDF and publish it as Linked Data including the discoverable services described in Chapter 4. The RDF translation can moreover be realized by the user's own DSSN node so that the user can be in full control of her data.

11 CONCLUSIONS AND FUTURE WORK

In this chapter, we conclude our work and discuss our results based on the initial research questions from Chapter 1[109]. Furthermore, we give some directions for future work.

11.1 ARCHITECTURE OF A DISTRIBUTED SEMANTIC SOCIAL NETWORK

In this thesis we described a reference architecture and proof-of-concept implementations of a distributed Social Network based on semantic technologies (Q1). Compared with the currently prevalent centralized social networks, this approach has a number of advantages regarding privacy, data security, data ownership, extensibility, reliability and freedom of communication. However, the work presented in thesis can only be a first step of a larger research and development agenda aiming at realizing a truly distributed social network based on semantic technologies.

We defined and re-used a robust framework of protocols and data definition standards for social interactions (Q2). We added services for a Linked Data infrastructure in order to shape a full featured DSSN architecture (Q3). Finally, we demonstrated that this architecture based on Semantic Web technologies is suitable in a distributed environment to build Social Networks (Q4).

Our prototype applications (each providing different user interfaces and usage concepts) demonstrate the usefulness of this architecture (Q1). Each application uses different parts of the architecture to provide its user interface (cf. Table 5).

- xOperator uses WebID profiles and attached data artifacts from the DSSN data layer and provides a read only chat interface based on an existing social network (Q5+6).
- MSSW fetches WebID profiles of the user and her linked friends. It uses the attached ping service to provide *friending* notification. A public search service is used to look for new friends. MSSW provides an integrated DSSN view for the mobile platform Android (Q6).
- OntoWiki/DSSN uses or provides the complete service layer. It can produce and consume history and activity streams as well as any type of data and media artifact incl. WebID profiles.

109 Research questions are referenced by its identifier *Q1 – Q7*.

Table 5: Comparison of all three applications according to their used architecture assets

Service	xOperator	MSSW	OntoWiki/DSSN
WebID profiles	query	fetch	fetch / provide
Data artifacts	query		fetch / provide
Media artifacts			link / provide
Activity streams			consume / provide
History feeds			consume / provide
Ping Service		use	use / provide
Update Service			provide
Search Service		use	use / provide
Push Service			use / provide

OntoWiki/DSSN integrates a well known data wiki application and transforms all data of the wiki into DSSN artifacts by enhancing them with DSSN services (ping and push, Q6+7). OntoWiki/DSSN can consume data artifacts as a users secretary using access delegation for the WebID protocol.

For a widespread use, the usability, scalability and the multi-client capabilities have to be improved. Likewise, the distributed realization of social networking applications (*apps*) as implemented in centralized social networks through APIs has to be investigated. We hope that the combination of standards and protocols as described in this thesis will be implemented in a number of additional platforms (e.g. Wordpress and Drupal), thus making them first-class nodes of the DSSN.

In addition to that, the proposed architecture is by no means limited to a specific set of agents, data artifacts and activities. In the same way as the architecture can be used by persons who share and comment pictures of their domestic animals, the architecture can be used to manage e.g. supply chains of a manufacturer. Some agents, artifacts and activities will change but the architecture is able to support such an usage scenario.

11.1.1 *The LUCID project*

As part of our future work we will work on the project LUCID – *Linked valUe ChaIn Data*. The project LUCID will tackle the data management problems in distributed supply chains networks. It is part of the small and medium enterprise (SME) innovation program in information

and communication technologies funded by the Federal Ministry of Education and Research in Germany.

One major issue of the LUCID project is the management of decentralized, secure and agile information streams. As part of this issue, the following research questions will be treated:

- How can we enable SMEs to publish relevant company data for supply chain networks up-to-date and secure on the Web in order to provide information for customers, suppliers and potential partners and to build supply networks on the fly and in an agile manner?
- Which attributes do we need in company profiles to allow for building of value added supply networks?
- How can we support the creation and management of company profiles and more detailed company information in order to allow SMEs to publish these data with minimal effort?
- How can we realize search features on this distributed catalog of data?

The LUCID project directly re-uses parts of the proposed architecture for distributed semantic social networks, esp. Semantic Pingback and WebID access delegation. In order to prepare the architecture for business use-cases we will extend them with more authorization and authentication control.

11.1.2 *The xodx project*

As a result of the presented research and esp. the presented architecture, the *xodx project* was created by former students of the University of Leipzig[110]. xodx is a more straightforward approach of creating an end-user interface for a distributed semantic social network node (cf. Figure 31). It implements the basic functionalities of a DSSN node which includes a WebID provider, Semantic Pingback functionality for friending / commenting and push notifications via PubSubHubbub. xodx is in use by a growing number of students as well as part of community projects such as the *Leipzig Data* project[111].

11.2 SEMANTIC PINGBACK

Although the Data Web is currently substantially growing, it still lacks a network effect as we could observe for example with the blogosphere in the Social Web. In particular coherence, information

110 https://github.com/white-gecko/xodx
111 http://www.leipzig-data.de/

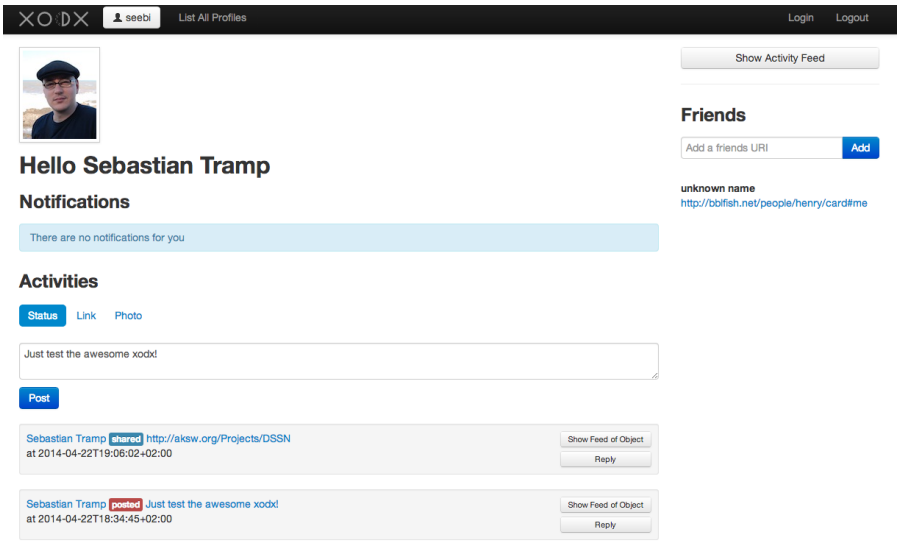

Figure 31: The xodx DSSN client

quality, and timeliness are still obstacles for the Data Web to become an Web-wide reality. With Semantic Pingback we aimed at extending and transferring the technological cornerstone of the Social Web the Pingback mechanism towards the Linked Data Web and thus a Distributed Semantic Social Network. The resulting mechanism has the potential to significantly improve the coherence on the Data Web, since linking becomes bi-directional. With its integrated provenance and spam prevention measures it helps to increase the information quality. Notification services based on Semantic Pingback such as used in the DSSN architecture increase the timeliness of distributed data. In addition these different benefits will mutually strengthen each other. Due to its complete downwards compatibility our Semantic Pingback also bridges the gap between the Social and the Data Web. We also expect the Semantic Pingback mechanism to support the transition process from data silos to flexible, decentralized structured information assets.

11.3 ACCESS DELEGATION

By presenting application use cases, we justified the need for an agent, such as a DSSN application or service, to act on behalf of a user in the role of a secretary. Our proposal for supporting such a communication schema involves adding a `:secretary` relation from the principal's profile to the agent that acts on its behalf. The secretary then connects to the resource using her own WebID. In addition to that, the secretary making the request need to add an `X-On-Behalf-Of` header field referring to the WebID of the principal in whose name she acts. This is especially useful if resources return different representations

depending on who makes the request, or on behalf of whom a request is made.

In our future work we will try to close the gap between the policy and access control vocabularies on the one hand and WebID authentication on the other hand. This will require mostly standardization work and seeing which type of solution is the most apt to gain traction in a wider community. As we deploy Social Web servers the need for a such a standard will be felt more and more strongly.

11.4 XOPERATOR

With the xOperator concept and its implementation, we have showed how a deeply and synergistic coupling of Semantic Web technology and Distributed Semantic Social Networks as well as Instant Messaging networks can be achieved. The approach naturally combines the well-balanced trust and provenance characteristics of IM networks with semantic representations and query answering of the Semantic Web. The xOperator approach goes significantly beyond existing work which mainly focused either on the semantic annotation of IM messages or on using IM networks solely as transport layers for SPARQL queries. It overlays the IM network with a network of personal (and group) agents, which have access to knowledge bases and Web resources of their respective owners. The neighbourhood of a user in the network can be easily queried by asking questions in a subset of natural language. By that xOperator resembles knowledge sharing and exchange in offline communities, such as a group of co-workers or friends. We have showcased how the xOperator approach naturally facilitates contacts and calendar management as well as access to large scale heterogeneous information sources. In addition to that, its extensible design allows a straightforward and effortless adoption to many other application scenarios such as, for example, sharing of experiment results in Biomedicine or sharing of account information in Customer Relationship Management.

In addition to adopting xOperator to new domain application we view the xOperator architecture as a solid basis for further technological integration of IM networks and the Semantic Web. This could include adding light-weight reasoning capabilities to xOperator or the automatic creation of AIML categories by applying NLP techniques. A more fine grained access control will be implemented in a future version. Instead of simply trusting all contacts on the roster, individual and group based policies can be created. Another issue for further research is the implementation of a more sophisticated routing protocol, that allows query traversal beyond directly connected nodes without flooding the whole network.

11.5 MOBILE DSSN CLIENT

The MSSW client is a further crucial piece in the medium-term agenda of realizing a truly distributed social network based on semantic technologies. Since mobile devices are playing an increasingly important role as clients and platforms for social networks, our realization focused on providing a extensible framework for social semantic networking on the Android platform. With this work we aimed at showcasing how different (social) Semantic Web standards, technologies and best practices can be integrated into a comprehensive architecture for social networking e.g. on mobile devices.

With regard to future work we plan to further decrease the entrance barrier for ordinary users. A current obstacle is that users are required to have a WebID and – if they want to use authentication and access control features – a FOAF+SSL enabled WebID. In particular creating a FOAF+SSL enabled WebID is, due to the certificate creation, still a cumbersome process. A possible simplification of this process would be to enable mobile phone users to create and upload the required profile and certificates directly from their mobile device. We also plan to implement a more efficient and user-friendly way for subscribing to updates of contacts. These will include profile changes, status updates, (micro-)blog posts as well as updates retrieved from social networking apps. This feature would be facilitated by a proxy infrastructure, which caches updates until the device re-connects to the network after a period of absence (e.g. due to limited network connection or switched-off devices). A further important aspect to be developed is the standardization and realization of social networking applications, which seamlessly integrate with and run on top of the DSSN. Such applications would comprise everything we know from centralized social networks (e.g. games, travel, quizzes etc.), but would make use of WebID Profiles and the other DSSN components for authentication, access control, subscription/notification etc.

11.6 ONTOWIKI/DSSN

With the extension of the data wiki application OntoWiki, we demonstrated how existing social web applications can be enabled to be part of the DSSN. With OntoWiki/DSSN users can interact in multiple ways with resources and services from their DSSN. OntoWiki/DSSN provides can handle incoming activity and history feed pushes, sends ping messages in each outgoing as well es receives and checks incoming ping requests. These enhancements are not only useful in a DSSN context but in a more general Linked Data infrastructure context as well. By using these protocols an enabling its managed resources, OntoWiki/DSSN is able consume and spin a multidomain Linked Data Web and not only a Web of social content. In addition to that, by

developing OntoWiki/DSSN, we evaluated which query requirements such DSSN applications impose on their triple backends.

With regard to future work we want to emphasize the *xodx* project again, which uses backend APIs and some libraries of OntoWiki/DSSN and places a more user-friendly and more social network specific user interface on top.

Part V

APPENDIX

A

CURRICULUM VITÆ

PERSONAL DATA

Sebastian Tramp, geb. Dietzold

Born on 29.09.1977

http://sebastian.tramp.name[112]

April 25, 2014

A.1 COMMUNITY SERVICES

A.1.1 *Organizing Committee*

- LSWT2013: 5. Leipziger Semantic Web Tag – Von Big Data zu Smart Data; Leipzig, Germany; September 23-24, 2013
- LSWT2011: 3. Leipziger Semantic Web Tag – Linked Data für die Massen; Leipzig Germany; May 4-5, 2011
- IMC-SSW2008: International Workshop on Interacting with Multimedia Content in the Social Semantic Web; in conjunction with the 3rd International Conference on Semantic and Digital Media Technologies (SAMT 2008); December 03, 2008; Koblenz, Germany

A.1.2 *Research Program Committee*

- WASABI2014: Workshop on Semantic Web Enterprise Adoption and Best Practice @ ESWC2014; Crete, Greece; May 26, 2014
- SWCS2013: Workshop on Semantic Web Collaborative Spaces; Montpellier, France; May 27, 2013
- WASABI2013: Workshop on Semantic Web Enterprise Adoption and Best Practice @ ISWC2013; Sydney, Australia; October 22, 2013
- Linked Data Cup @ I-Semantics; Graz, Austria; September 5-7, 2012

112 Usable as WebID

- LDOW2012: Workshop on Linked Data on the Web; Lyon, France; April 16, 2012
- SWCS2012: Workshop on Semantic Web Collaborative Spaces; Lyon, France; April 17, 2012
- ESWC2012: The 9th Extended Semantic Web Conference 2012 – Research Track; Heraklion, Crete, Greece; May 27-31, 2012
- STAKE2011: Semantic Technology and Knowledge Engineering; Putrajaya Campus, Uniten, Malaysia; July 2011
- SDoW2011: Social Data on the Web Workshop; Bonn, Germany; October 23, 2011
- SDoW2010: 3rd Social Data on the Web workshop; Shanghai, China; November 8, 2010
- STAKE2010: 2nd Semantic Technology and Knowledge Engineering Conference; Kuching, Sarawak, Malaysia; July 28, 2010
- SKIL2010: Studentenkonferenz Informatik Leipzig 2010; Leipzig, Germany; September 28, 2010
- SFSW2010: 6th Workshop on Scripting and Development for the Semantic Web; Hersonissos, Crete, Greece; May 31, 2010
- SemWiki2010: Fifth Workshop on Semantic Wikis – Linking Data and People; Hersonissos, Crete, Greece; May 31, 2010
- SDoW2009: 2nd Social Data on the Web workshop; co-located with the 8th International Semantic Web Conference (ISWC2009); Washington DC (USA); October 25, 2009
- ESIW 2009: Workshop on Exploiting Structured Information on the Web; in conjunction with 10th International Conference on Web Information Systems Engineering (WISE 2009); Poznań, Poland; October 5–7, 2009
- SFSW2009: 5th Workshop on Scripting and Development for the Semantic Web; Colocated with ESWC 2009; May/June, 2009; Crete, Greece
- SemWiki2009: Fourth Workshop on Semantic Wikis, co-located with the 6th Annual European Semantic Web Conference (ESWC2009) in Crete, Greece
- SAW2009: 3rd Workshop on Social Aspects of the Web; in conjunction with the 12th International Conference on Business Information Systems (BIS 2009); April 27 2009; Poznan, Poland

- SDoW2008: 1st Social Data on the Web Workshop; co-located with the 7th International Semantic Web Conference (ISWC2008); October 26/27, 2008; Karlsruhe, Germany
- SemWiki2008: 3rd Semantic Wiki Workshop; co-located with the 5th European Semantic Web Conference (ESWC2008); June 02, 2008; Tenerife, Spain
- SFSW2008: 4th Workshop on Scripting for the Semantic Web; co-located with the 5th European Semantic Web Conference (ESWC2008); June 01, 2008; Tenerife, Spain

A.1.3 *Reviewing*

- I-Challenge 2013 on 9th International Conference on Semantic Systems; Graz, Austria; September 4-6, 2013
- International Journal On Semantic Web and Information Systems; 2013
- JWS: Journal of Web Semantics – Special Issue on the Semantic and Social Web, 2012
- Open Data Challenge 2011; June 2011
- ICWE2011: 11th International Conference on Web Engineering; Paphos, Cyprus; 20-24 June, 2011
- KEOD2011: International Conference on Knowledge Engineering and Ontology Development; Paris, France; 26 Oktober 2011
- SWJ: Semantic Web Journal – Interoperability, Usability, Applicability
- Springer Science+Business Media, Publishing Group, Heidelberg; Book in Computer Science
- IJSWIS: International Journal On Semantic Web and Information Systems, Special Issue on Linked Data
- ESWC2009: The 6th Annual European Semantic Web Conference – Demonstrations Track; 31 May – 4 June 2009, Heraklion, Greece
- LDOW2008: Linked Data on the Web; Workshop at the 17th International World Wide Web Conference (WWW2008); April 22, 2008; Beijing, China
- CSSW2007: Conference on Social Semantic Web; September 26–28, 2007; Leipzig, Germany

A.2 SEMINARS AND TEACHING

- Web-Engineering, Web-Anwendungssysteme und Mobile Computing (WS2013/2014, Leipzig School of Media)
- Praktikum Distributed Semantic Social Networks (WS2012/2013, Institute of Computer Science at University of Leipzig)
- Web-Engineering, Web-Anwendungs- systeme und Mobile Computing (WS2012/2013, Leipzig School of Media)
- Semantic Web Software (SS2012, Information Systems Institute at University of Leipzig)
- Praktikum Semantic Web (SS2012, Institute of Computer Science at University of Leipzig)
- Distributed Semantic Social Networks (2012, LOD2 Indian Summer School)
- Web-Engineering und Web-Anwendungssysteme (WS2011/2012, Leipzig School of Media)
- Web-Engineering und Web-Anwendungssysteme (WS2010/2011, Leipzig School of Media)
- Semantic Web (SS2010, Gastvorlesung Information Systems Institute at University of Leipzig)
- Semantic Web Praktikum (SS2010, Institute of Computer Science at University of Leipzig)
- Softwaretechnik-Praktikum (SS2009, Institute of Computer Science at University of Leipzig)
- Semantik Web Praktikum (SS2009, Institute of Computer Science at University of Leipzig)
- Schwerpunktpraktikum Semantische Technologien / Social Semantic Web (WS2008/2009, Institute of Computer Science at University of Leipzig)
- Semantische Unterstützung von Software Entwicklungsprozessen (SS2008, Institute of Computer Science at University of Leipzig)
- Softwaretechnik-Praktikum (SS2008, Institute of Computer Science at University of Leipzig)
- Entwicklung von Semantischen Webanwendungen (SS2007, Institute of Computer Science at University of Leipzig)
- Semantic Web Services und Interfaces (WS2006/2007, Institute of Computer Science at University of Leipzig)

- Agiles Software and Knowledge Engineering (SS2006, Institute of Computer Science at University of Leipzig)

A.3 SUPERVISION

A.3.1 *Bachelor*

- Markus Freudenberg: Konzept und Implementierung eines Triple Store Event Frameworks
- Norman Radtke: Konzept und Implementierung aktivitätsbasierter Kommunikation für verteilte semantische Soziale Netzwerke
- Alrik Hausdorf: Konzept und Implementierung erweiterter Methoden für Web-basierte Präsentationen
- Matthias-Christian Ott: Spezifikation einer Ausführungsumgebung für Web-Anwendungen in Mozilla Firefox
- Konrad Abicht: Gewichtung von Aussagen in RDF Graphen
- Natanael Arndt: Entwicklung eines mobilen Social Semantic Web Clients
- Jonas Brekle: Ein SPARQL Query Komponente für die API-basierte Manipulation von SPARQL Abfragen
- Leszek Kotas: Evaluation und Erweiterung des OntoWiki Plugin-Systems
- Maria Moritz: Integration des DL-Learners in OntoWiki
- Thanh Nghia Lam: SPARQL Templates für die Darstellung von RDF Inhalten
- Thomas Sobik: Management von Standard Operation Procedures
- Atanas Alexandrov: Implementierung eines Oracle basiertem RDF Store Backends
- Martin Peklo: Semantische Supportdatenbank

A.3.2 *Master*

- Natanael Arndt: Xodx – Konzeption und Implementierung eines Distributed Semantic Social Network Knotens
- Rolland Brunec: Ein modularer und backend-unabhängiger SPARQL / SPARUL Parser für PHP
- Atanas Alexandrov: Tagging in Semantischen Daten-Wikis

- Martin Peklo: JavaScript API zur Manipulation von RDF Modellen
- Jörg Unbehauen: Konzept und Implementierung eines Semantischen Agenten
- Thomas Kappel: Integration von Sozialen Netzwerken mit dem Sozializr
- Christian Lehmann: Implementierung eines D2RQ Mapping Backends für RAP

A.3.3 *Diploma*

- Elena Kop: Eine Semantic Web Ontologie für den Gasmarkt
- Christoph Riess: Evaluations-Pattern in Semantic Web Wissensbasen
- Philipp Frischmuth: Interwiki-Kommunikation in semantischen Daten-Wikis
- Nomunbilegt Batsukh: Unterstützung von Online Shop Systemen mit Semantischen Technologien
- Feng Qiu: Semantisches Plugin Repository
- Norman Heino: Intelligente RDFa-Widgets
- Michael Haschke: Semantic Personal Knowledge Management mit OntoWiki
- Stefan Berger: Access Control on Triple Stores
- Enrico Popp: WebDAV Dateisystem für RDF Modelle
- Denis Gärtner: LDAP Query to SPARQL Transformation

BIBLIOGRAPHY

F. Abel, I. Celik, G.-J. Houben, and P. Siehndel. Leveraging the Semantics of Tweets for Adaptive Faceted Search on Twitter. In *10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings*, volume 7031 of *LNCS*, pages 1–17, 2011. ISBN 978-3-642-25072-9. URL http://link.springer.com/chapter/10.1007%2F978-3-642-25073-6_1.

B. Adida, M. Birbeck, S. McCarron, and S. Pemberton. RDFa in XHTML: Syntax and Processing. Recommendation, W3C, 2008. URL http://www.w3.org/TR/rdfa-syntax/.

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007, Proceedings*, number 4825 in LNCS, pages 722–735, 2007. ISBN 978-3-540-76297-3. URL http://link.springer.com/chapter/10.1007%2F978-3-540-76298-0_52.

A. Barth. The Web Origin Concept. RFC 6454, IETF, Dec. 2011. URL https://tools.ietf.org/html/rfc6454.

C. Becker and C. Bizer. Exploring the Geospatial Semantic Web with DBpedia Mobile. *J. Web Sem.*, 7(4):278–286, 2009. URL http://www.websemanticsjournal.org/index.php/ps/article/view/266.

D. Beckett. RDF/XML Syntax Specification (Revised). Recommendation, Word Wide Web Consortium, 10 Feb. 2004. URL http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

D. Beckett and T. Berners-Lee. Turtle - Terse RDF Triple Language. Member Submission, W3C, 2004. URL http://www.w3.org/TeamSubmission/turtle/.

D. Beckett and J. Broekstra. SPARQL Query Results XML Format. Recommendation, W3C, Mar. 2013. URL http://www.w3.org/TR/rdf-sparql-XMLres/.

T. Berners-Lee. Linked Data – Design Issues, July 2006. URL http://www.w3.org/DesignIssues/. last change: 2010/04/12; retrieved: 2014/03/05.

T. Berners-Lee. Long Live the Web: A Call for Continued Open Standards and Neutrality. *Scientific American*, Dec. 2010. URL http://www.scientificamerican.com/article.cfm?id=long-live-the-web.

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, pages 29–37, May 2001. URL http://www.scientificamerican.com/article/the-semantic-web/.

T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, IETF, Jan. 2005. URL https://tools.ietf.org/html/rfc3986.

A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, and H. Zhang. The growth of Diaspora - A decentralized online social network in the wild. In *2012 Proceedings IEEE INFOCOM Workshops, Orlando, FL, USA, March 25-30, 2012*, pages 13–18. IEEE, 2012. ISBN 978-1-4673-1016-1. URL http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=6193476.

C. Bizer and R. Cyganiak. The TriG Syntax. Syntax Reference, Freie Universität Berlin, July 2007. URL http://wifo5-03.informatik.uni-mannheim.de/bizer/trig/.

C. Bizer, R. Cyganiak, and T. Heath. How to Publish Linked Data on the Web. Technical Report, Freie Universität Berlin, 2007. URL http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/.

C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009. URL http://dx.doi.org/10.1016/j.websem.2009.07.002.

U. Bojars, A. Passant, R. Cyganiak, and J. Breslin. Weaving SIOC into the Web of Linked Data. In *Proceedings of the Linked Data on the Web Workshop*, 2008. URL http://events.linkeddata.org/ldow2008/papers/01-bojars-passant-weaving-sioc.pdf.

P. Bouquet, H. Stoermer, C. Niederée, and A. Mana. Entity Name System: The Back-Bone of an Open and Scalable Web of Data. In *Proceedings of the 2th IEEE International Conference on Semantic Computing (ICSC 2008)*, 2008. URL http://www.okkam.org/publications/stoermer-EntityNameSystem.pdf.

T. Bray, D. Hollander, A. Layman, R. Tobin, and H. S. Thompson. Namespaces in XML 1.0 (Third Edition). Recommendation, W3C, Dec. 2009. URL http://www.w3.org/TR/REC-xml-names/.

J. Breslin, A. Harth, U. Bojars, and S. Decker. Towards Semantically-Interlinked Online Communities. In *Second European Semantic*

Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29–June 1, 2005, Proceedings, 2005. URL http://link.springer.com/chapter/10.1007%2F11431053_34.

J. G. Breslin, S. Decker, A. Harth, and U. Bojars. SIOC: an approach to connect web-based communities. *International Journal of Web Based Communities*, 2(2):133–142, 2006. URL http://www.inderscience.com/info/inarticle.php?artid=10305.

D. Brickley and R. Guha. RDF Schema 1.1. Recommendation, W3C, Feb. 2014. URL http://www.w3.org/TR/rdf-schema/.

D. Brickley and L. Miller. FOAF Vocabulary Specification. Namespace Document 2 Sept 2004, FOAF Project, 2004. URL http://xmlns.com/foaf/0.1/. http://xmlns.com/foaf/0.1/.

G. Carothers. RDF 1.1 N-Quads: A line-based syntax for RDF datasets. Recommendation, W3C, Feb. 2014. URL http://www.w3.org/TR/n-quads/.

G. Carothers and A. Seaborne. RDF 1.1 N-Triples: A line-based syntax for an RDF graph. Recommendation, W3C, Feb. 2014. URL http://www.w3.org/TR/n-triples/.

P. Ciccarese, E. Wu, G. T. Wong, M. Ocana, J. Kinoshita, A. Ruttenberg, and T. Clark. The SWAN biomedical discourse ontology. *Journal of Biomedical Informatics*, 41(5):739–751, 2008. URL http://www.j-biomed-inform.com/article/S1532-0464(08)00058-0/.

K. G. Clark. SPARQL 1.1 Protocol. Recommendation, W3C, Mar. 2013. URL http://www.w3.org/TR/sparql11-protocol/.

S. Corlosquet, R. Cyganiak, A. Polleres, and S. Decker. RDFa in Drupal: Bringing cheese to the web of data. In *Proc. of 5th Workshop on Scripting and Development for the Semantic Web at ESWC 2009*, 2009. URL http://ceur-ws.org/Vol-449/ShortPaper3.pdf.

R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 Concepts and Abstract Syntax. Recommendation, W3C, Feb. 2014. URL http://www.w3.org/TR/rdf11-concepts/.

J. David and J. Euzenat. Linked data from your pocket: The Android RDFContentProvider. In *Posters and Demos of the ISWC2010*, 2010. URL http://iswc2010.semanticweb.org/pdf/492.pdf.

I. Davis, T. Steiner, and A. J. L. Hors. RDF 1.1 JSON Alternate Serialization (RDF/JSON). Working Group Note, W3C, Nov. 2013. URL http://www.w3.org/TR/rdf-json/.

F. Dawson and D. Stenerson. Internet Calendaring and Scheduling Core Object Specification (iCalendar). RFC 2445, IETF, Nov. 1998. URL https://tools.ietf.org/html/rfc3986.

T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol – Version 1.2. RFC 5346, IETF, Aug. 2008. URL https://tools.ietf.org/html/rfc5246.

L. Ding, T. W. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*, pages 652–659, 2004. ISBN 1-58113-874-1. URL http://ebiquity.umbc.edu/_file_directory_/papers/116.pdf.

L. Dodds and I. Davis. *Linked Data Patterns – A pattern catalogue for modelling, publishing, and consuming Linked Data*. 2014. URL http://patterns.dataincubator.org/.

M. Duerst and M. Suignard. Internationalized Resource Identifiers (IRIs). RFC 3987, IETF, Jan. 2005. URL https://tools.ietf.org/html/rfc3987.

D. Eastlake and A. Panitz. Reserved Top Level DNS Names. RFC 2606, IETF, June 1999. URL https://tools.ietf.org/html/rfc2606.

O. Erling. Virtuoso, a hybrid rdbms/graph column store. *IEEE Data Eng. Bull.*, 35(1):3–8, 2012. URL http://sites.computer.org/debull/A12mar/vicol.pdf.

O. Erling and I. Mikhailov. Rdf support in the virtuoso dbms. In S. Auer, C. Bizer, C. Müller, and A. V. Zhdanova, editors, *The Social Semantic Web 2007, Proceedings of the 1st Conference on Social Semantic Web (CSSW), September 26-28, 2007, Leipzig, Germany*, volume 113 of *LNI*, pages 59–68. GI, 2007. URL http://subs.emis.de/LNI/Proceedings/Proceedings113/gi-proc-113-006.pdf.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999. URL https://tools.ietf.org/html/rfc2616.

T. Franz and S. Staab. SAM: Semantics Aware Instant Messaging for the Networked Semantic Desktop. In *Semantic Desktop Workshop at the ISWC*, 2005. URL http://CEUR-WS.org/Vol-175/11_franzstaab_sam_final.pdf.

E. Freese. Enhancing AIML Bots using Semantic Web Technologies. In *Proc. of Extreme Markup Languages*, 2007. URL http://conferences.idealliance.org/extreme/html/2007/Freese01/EML2007Freese01.html.

H. Glaser, A. Jaffri, and I. Millard. Managing Co-reference on the Semantic Web. In *Proceedings of the Linked Data on the Web Workshop*

(LDOW2009), 2009. URL http://ceur-ws.org/Vol-538/ldow2009_paper11.pdf.

H. Halpin and M. Tuffield. A Standards-based, Open and Privacy-aware Social Web. Incubator Group Report, W3C, Dec. 2010. URL http://www.w3.org/2005/Incubator/socialweb/XGR-socialweb/.

E. Hammer. The OAuth 1.0 Protocol. RFC 5849, IETF, Apr. 2010. URL https://tools.ietf.org/html/rfc5849.

D. Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, IETF, Oct. 2012. URL https://tools.ietf.org/html/rfc6749.

S. Harris and A. Seaborne. SPARQL 1.1 Query Language. Recommendation, W3C, Mar. 2013. URL http://www.w3.org/TR/sparql11-query/.

O. Hartig. Provenance Information in the Web of Data. In *Linked Data on the Web Workshop, April 20, 2009, Madrid, Spain.*, 2009. URL http://events.linkeddata.org/ldow2009/papers/ldow2009_paper18.pdf.

T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool Publishers, Feb. 2011. URL http://linkeddatabook.com/.

P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure. *Semantic Web: Grundlagen*. eXamen.press, 2007. URL http://semantic-web-grundlagen.de/.

P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2 Web Ontology Language Primer (Second Edition). Recommendation, W3C, Dec. 2012. URL http://www.w3.org/TR/owl-primer/.

P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, IETF, Aug. 2012. URL https://tools.ietf.org/html/rfc6698.

R. Iannella, H. Halpin, B. Suda, and N. Walsh. Representing vCard Objects in RDF. Member Submission, W3C, Jan. 2010. URL http://www.w3.org/Submission/vcard-rdf/.

I. Jacobs and N. Walsh. Architecture of the World Wide Web, Volume One. Recommendation, W3C, Dec. 2004. URL http://www.w3.org/TR/webarch/.

K. Jones. The growth of social media v2.0. *Search Engine Journal*, 11-15, 2013. URL http://www.searchenginejournal.com/growth-social-media-2-0-infographic/77055/.

A. M. Kaplan and M. Haenlein. Users of the world, unite! The challenges and opportunities of Social Media. *Business Horizons*, 53(1): 59 – 68, 2010. ISSN 0007-6813. doi: http://dx.doi.org/10.1016/j.bushor.2009.09.003. URL http://www.sciencedirect.com/science/article/pii/S0007681309001232.

D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A General-Purpose Information Management Tool for End Users Based on Semistructured Data. In *Proc. of the Second Biennial Conference on Innovative Data Systems Research (CIDR2005), Asilomar, CA, USA, January 4-7, 2005*, pages 13–26, 2005. URL http://www.cidrdb.org/cidr2005/papers/P02.pdf.

E. Kaufmann and A. Bernstein. How Useful are Natural Language Interfaces to the Semantic Web for Casual End-users? In *6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007, Proceedings*, pages 281–294, 2007. URL http://link.springer.com/chapter/10.1007%2F978-3-540-76298-0_21.

M. Krohn, A. Yip, M. Brodsky, R. Morris, and M. Walfish. A World Wide Web Without Walls. In *6th ACM Workshop on Hot Topics in Networking (Hotnets)*, Atlanta, GA, USA, November 2007. URL http://pdos.csail.mit.edu/papers/w5-hotnets07.pdf.

S. Langridge and I. Hickson. Pingback 1.0. Technical Report, personal document, 2002. URL http://hixie.ch/specs/pingback/pingback.

O. Lassila. Using the Semantic Web in Mobile and Ubiquitous Computing. In *Proc. of the 1st IFIP WG12.5 Working Conference on Industrial Applications of Semantic Web, Jyväskylä, Finland*, volume 188 of *IFIP*, pages 19–25, 2005. URL http://www.springerlink.com/index/6t254j1736537p41.pdf.

O. Lassila and M. Adler. Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web. In *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 363–376. MIT Press, 2003. ISBN 0-262-06232-1. URL http://www.lassila.org/publications/2002/lassila-nist-pervasive-2002.pdf.

D. Le-Phuoc, J. X. Parreira, V. Reynolds, and M. Hauswirth. RDF On the Go: An RDF Storage and Query Processor for Mobile Devices. In *Posters and Demos of the ISWC 2010, Shanghai, China*, 2010. URL http://iswc2010.semanticweb.org/pdf/503.pdf.

B. Leuf and W. Cunningham. *The Wiki Way – Quick Collaboration on the Web*. Addison-Wesley Longman, 2001. URL http://www.wiki.org.

A. Lukas. The Mine! as VRM infrastructure. Technical Report, 2008. URL http://www.mediainfluencer.net/wp/wp-content/uploads/2008/05/mine-paper-v1.pdf.

B. McBride. Jena: A Semantic Web Toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1067737.

A. Miles and S. Bechhofer. SKOS Simple Knowledge Organization System Reference. Recommendation, W3C, Aug. 2009. URL http://www.w3.org/TR/skos-reference/.

M. Minno and D. Palmisano. *Atom Activity Streams RDF mapping*. NoTube Project, 2010. URL http://xmlns.notu.be/aair/.

M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo. DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In *10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings*, volume 7031 of *LNCS*, pages 454–469, 2011. ISBN 978-3-642-25072-9. URL http://link.springer.com/chapter/10.1007%2F978-3-642-25073-6_29.

OpenSocial Spec. Opensocial Specification 2.5.1. Technical Report, OpenSocial and Gadgets Specification Group, 2011. URL http://opensocial.github.io/spec/2.5.1/OpenSocial-Specification.xml.

F. Osterfeld, M. Kiesel, and S. Schwarz. Nabu – A Semantic Archive for XMPP Instant Messaging. In *Semantic Desktop Workshop at the International Semantic Web Conference*, 2005. URL http://CEUR-WS.org/Vol-175/40_kiesel_nabu_final.pdf.

A. Passant and P. N. Mendes. sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub. In *Proceedings of the Scripting for the Semantic Web Workshop 2010*, 2010. URL http://ceur-ws.org/Vol-699/Paper6.pdf.

A. Passant, J. G. Breslin, and S. Decker. Rethinking Microblogging: Open, Distributed, Semantic. In B. Benatallah, F. Casati, G. Kappel, and G. Rossi, editors, *10th International Conference, ICWE 2010, Vienna, Austria, July 5-9, 2010. Proceedings*, volume 6189 of *LNCS*, pages 263–277, 2010. URL http://link.springer.com/chapter/10.1007%2F978-3-642-13911-6_18.

D. Peterson, S. S. Gao, A. Malhotra, C. M. Sperberg-McQueen, and H. S. Thompson. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. Recommendation, W3C, Apr. 2012. URL http://www.w3.org/TR/xmlschema11-2/.

A. Phillips and M. Davis. Tags for Identifying Languages. BCP 47, IETF, Sept. 2009. URL http://tools.ietf.org/html/bcp47.

E. Prodromou, B. Vibber, J. Walker, and Z. Copley. OStatus 1.0 Draft 2. Technical Report, StatusNet Inc, 2010. URL http://www.w3.org/community/ostatus/wiki/images/9/93/OStatus_1.0_Draft_2.pdf.

D. Quan, K. Bakshi, and D. R. Karger. A Unified Abstraction for Messaging on the Semantic Web. In *The Twelfth International World Wide Web Conference (Posters)*, 2003. URL http://dblp.uni-trier.de/db/conf/www/www2003p.html#QuanBK03.

D. Reynolds. The Organization Ontology. Recommendation, W3C, Jan. 2014. URL http://www.w3.org/TR/vocab-org/.

M. Rowe. Interlinking Distributed Social Graphs. In *Proceedings of the Linked Data on the Web Workshop 2009*, 2009. URL http://ceur-ws.org/Vol-538/ldow2009_paper5.pdf.

M. Sabadello. A Federated Social Web for Peace. In *Federated Social Web Europe 2011, Berlin June 3rd-5th 2011*, 2011. URL http://d-cent.org/fsw2011/wp-content/uploads/fsw2011-A-Federated-Social-Web-for-Peace.pdf.

P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920, IETF, Oct. 2004. URL https://tools.ietf.org/html/rfc3920.

B. Schandl and S. Zander. Adaptive RDF Graph Replication for Mobile Semantic Web Applications. *Ubiquitous Computing and Communication Journal (Special Issue on Managing Data with Mobile Devices)*, 7 2009. URL http://www.cs.univie.ac.at/upload//223/papers/2009/ubicc2009-schandl.pdf.

G. Schreiber and Y. Raimond. RDF 1.1 Primer. Working Group Note, W3C, Feb. 2014. URL http://www.w3.org/TR/rdf11-primer/.

F. Schwagereit, A. Scherp, and S. Staab. Representing Distributed Groups with dgFOAF. In *7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 2, 2010, Proceedings*, pages 181–195, June 2010. URL http://data.semanticweb.org/conference/eswc/2010/paper/social_web/6.

S.-W. Seong, J. Seo, M. Nasielski, D. Sengupta, S. Hangal, S. K. Teh, R. Chu, B. Dodson, and M. S. Lam. PrPl: a decentralized social networking infrastructure. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, MCS '10, pages 8:1–8:8, 2010. ISBN 978-1-4503-0155-8. doi: http://doi.acm.org/10.1145/1810931.1810939. URL http://doi.acm.org/10.1145/1810931.1810939.

N. Shadbolt, T. Berners-Lee, and W. Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006. URL http://dl.acm.org/citation.cfm?id=1155373.

S. B. Shum, D. D. Roure, M. Eisenstadt, N. Shadbolt, and A. Tate. CoAKTinG: Collaborative Advanced Knowledge Technologies in the Grid. In *Proc. of 2. Workshop on Adv. Collab. Env. at the HPDC-11*, 2002. URL http://www.bib.ecs.soton.ac.uk/data/7480/pdf/CoAKTinG-WACE2002.pdf.

D. Sonntag, R. Engel, G. Herzog, A. Pfalzgraf, N. Pfleger, M. Romanelli, and N. Reithinger. SmartWeb Handheld – Multimodal Interaction with Ontological Knowledge Bases and Semantic Web Services. In *Artifical Intelligence for Human Computing – ICMI 2006 and IJCAI 2007 International Workshops, Banff, Canada, November 3, 2006 Hyderabad, India, January 6, 2007 Revised Selected Papers*, volume 4451 of *LNCS*, 2007. ISBN 978-3-540-72346-2. URL http://www.dfki.de/~romanell/ijcai07-sw.pdf.

M. Sporny, G. Kellogg, and M. Lanthaler. JSON-LD 1.0: A JSON-based Serialization for Linked Data. Recommendation, W3C, Feb. 2014. URL http://www.w3.org/TR/json-ld/.

H. Story, B. Harbulot, I. Jacobi, and M. Jones. FOAF+SSL: RESTful Authentication for the Social Web. In *Proceedings of the 1st Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, 2009. URL http://CEUR-WS.org/Vol-447/paper5.pdf.

H. Story, S. Corlosquet, and A. Sambra. WebID-TLS – WebID Authentication over TLS. Editor's Draft, W3C, July 2013. URL https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/tls-respec.html.

A. D. Thurston. DSNP: Distributed Social Networking Protocol. Technical Report, 2011. URL http://www.complang.org/dsnp/spec/dsnp-spec.pdf.

G. Tummarello, R. Delbru, and E. Oren. Sindice.com: Weaving the Open Linked Data. In *ISWC'07/ASWC'07 Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, volume 4825 of *LNCS*, pages 552–565, 2007. ISBN 978-3-540-76297-3. URL http://link.springer.com/content/pdf/10.1007%2F978-3-540-76298-0_40.

R. Wallace. Artificial Intelligence Markup Language (AIML). Working Draft, A.L.I.C.E. AI Foundation, 2005. URL http://docs.aitools.org/aiml/spec/WD-aiml-1.0.1-20050218-007.html. 18 February 2005.

M. G. White. What types of social networks exist? Blogpost, 2014. URL http://socialnetworking.lovetoknow.com/What_Types_of_Social_Networks_Exist.

M. Wilson, A. Russell, D. A. Smith, A. Owens, and M. C. Schraefel. mSpace Mobile: A Mobile Application for the Semantic Web. In *Proc. of the ISWC 2005 Workshop on End User Semantic Web Interaction, Galway, Ireland*, 2005. URL http://eprints.soton.ac.uk/261101/1/iswc-final.pdf.

COLOPHON

This thesis was typeset with LaTeX 2ε using Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL* were used). The listings are typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

Beside multiple standard LaTeX 2ε packages, this thesis uses a Semantic Web related package called *qname*[113].

The typographic style was inspired by R. Bringhurst as presented in *The Elements of Typographic Style*. It is available for LaTeX via CTAN as "`classicthesis`".

The custom size of the textblock was calculated using the directions given by Mr. Bringhurst (pages 26–29 and 175/176). 10 pt Palatino needs 133.21 pt for the string "abcdefghijklmnopqrstuvwxyz". This yields a good line length between 24–26 pc (288–312 pt). Using a "*double square textblock*" with a 1:2 ratio this results in a textblock of 312:624 pt (which includes the headline in this design).

Final Version as of April 25, 2014 at 1:28.

113 https://github.com/white-gecko/qname.sty

DECLARATION

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, April 2014

Sebastian Tramp