

Weaving a Social Data Web with Semantic Pingback

Sebastian Tramp, Philipp Frischmuth, Timofey Ermilov, and Sören Auer

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany,
{lastname}@informatik.uni-leipzig.de
<http://aksw.org>

Abstract. In this paper we tackle some of the most pressing obstacles of the emerging Linked Data Web, namely the *quality, timeliness and coherence* as well as *direct end user benefits*. We present an approach for complementing the Linked Data Web with a social dimension by extending the well-known Pingback mechanism, which is a technological cornerstone of the blogosphere, towards a Semantic Pingback. It is based on the advertising of an RPC service for propagating typed RDF links between Data Web resources. Semantic Pingback is downwards compatible with conventional Pingback implementations, thus allowing to connect and interlink resources on the Social Web with resources on the Data Web. We demonstrate its usefulness by showcasing use cases of the Semantic Pingback implementations in the semantic wiki OntoWiki and the Linked Data interface for database-backed Web applications Triplify.

Introduction

Recently, the publishing of structured, semantic information as Linked Data has gained much momentum. A number of Linked Data providers meanwhile publish more than 200 interlinked datasets amounting to 13 billion facts¹. Despite this initial success, there are a number of substantial obstacles, which hinder the large-scale deployment and use of the Linked Data Web. These obstacles are primarily related to the *quality, timeliness and coherence* of Linked Data as well as to providing *direct benefits to end users*. 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 paper 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 [9], which is one of the technological cornerstones of the overwhelming success of the blogosphere in the Social Web. The Pingback

¹ <http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics>

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 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 `owl:seeAlso` links can be automatically added to the Data Web resource. In order to facilitate the adoption of the Semantic Pingback mechanism we developed three complementary implementations: a Semantic Pingback implementation was included into the semantic data wiki OntoWiki, we added support for Semantic Pingbacks to the Triplify database-to-RDF mapping tool and provide a standalone implementation for the use by other tools or services.

The paper is structured as follows: We describe the requirements which guided the development of Semantic Pingback in section 1. We present an architectural overview including communication behaviour and autodiscovery algorithms of our solution in section 2. A description of our implementations based on OntoWiki and Triplify as well as the standalone software is given in section 5. Finally, we survey related work in section 6 and conclude with an outlook on future work in section 7.

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 (X)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 annotations², these systems should be able to propagate typed links derived from the 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 [4] 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 post³.

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

² 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 [6] for Drupal.

³ Since SIOC is a very generic vocabulary, people can also use more specific relations as, for instance, `disagreesWith` or `alternativeTo` from the Scientific Discourse Relationships Ontology [5].

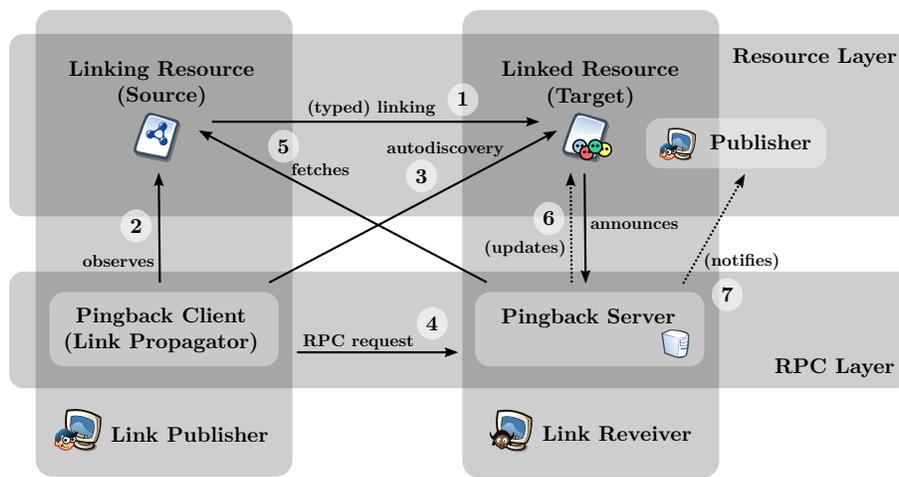


Fig. 1. Architecture of the Semantic Pingback approach.

through a `<link>`-Element in an HTML document. 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.

2 Architectural Overview

The general architecture of the Semantic Pingback approach is depicted in Figure 1. 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 [9].

The following scenario, which was introduced in the above mentioned specification, illustrates the chain of communication steps executed for a single Pingback request:

1. Alice posts to her blog. The post she has made (*source resource*) includes a link to a post on Bob's blog (*target resource*).
2. Alice's blogging system (*Pingback client*) contacts Bob's blogging system (*Pingback server*) and propagates that a link to a post inside Bob's environment was established.
3. Bob's blogging system then verifies, that Alice's post indeed includes a link to Bob and adds a link back to Alice's post on his original post.
4. Readers of Bob's article can follow this link to Alice's post to read her opinion.

This scenario as well 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 section 3.1 for autodiscovery of Pingback server information).

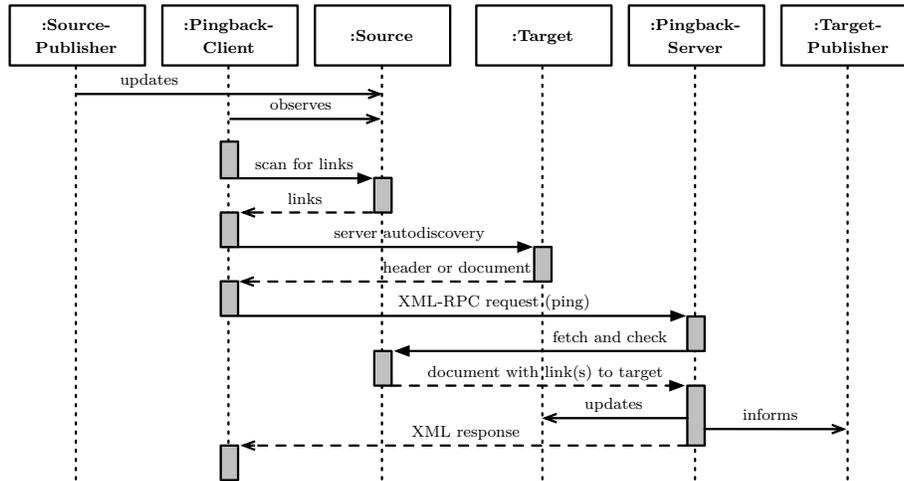


Fig. 2. Sequence diagram illustrating the (Semantic) Pingback workflow.

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 (X)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 through 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 4.

Figure 2 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. 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 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 4 the server behavior is described in more detail. Finally, the Pingback server responds with an XML result.

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.

3.1 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 **service**⁴, 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 resource 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 HTML link element too) is that Pingback clients need to retrieve and parse the document instead of requesting the HTTP header only.

4 Server Behavior

While the communication behavior of the server is completely compatible with the conventional Pingback mechanism (as described in [9]), 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.

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.

⁴ <http://purl.net/pingback/service>

- *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. trackbacks⁵.

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 X(HTML) link. This enables the server to verify the given source resource even without being aware of Linked Data and RDF.

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 (X)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 (X)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, metadata about the source resource (e.g. a label and a description) can be stored as well.

⁵ <http://en.wikipedia.org/wiki/Trackback>

4.3 Provenance Tracking

Provenance information can be recorded using the provenance vocabulary [8]⁶. 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 [8], we define a *creation guideline* for Pingback requests, which is described in this paper, and identified by the URI <http://purl.net/pingback/Request>. A specific Pingback request *execution* is then performed by a Pingback *data creating service*, which uses the defined creation guideline.

The following listing shows an example provenance model represented in N3:

```
1 @prefix : <http://purl.org/net/provenance/ns#> .
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
4 @prefix sioc: <http://rdfs.org/sioc/ns#> .
5
6 [a rdf:Statement;
7   rdf:subject <http://example1.org/Source>;
8   rdf:predicate sioc:links_to;
9   rdf:object <http://example2.org/Target>;
10  :containedBy [
11    a :DataItem;
12    :createdBy [
13      a :DataCreation;
14      :performedAt "2010-02-12T12:00:00Z";
15      :performedBy [
16        a :DataCreatingService;
17        rdfs:label "Semantic Pingback Service" ];
18        :usedData [
19          a :DataItem;
20          :containedBy <http://example1.org/Source> ];
21          :usedGuideline <http://purl.net/pingback/Request> ]];
22 ] .
```

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.

5 Implementation and Evaluation

In this section we describe the implementation and evaluation of Semantic Pingback in three different scenarios. We implemented Semantic Pingback server and client functionality for OntoWiki in order to showcase the semantic features of the approach. Semantic Pingback server functionality was integrated in Triplify,

⁶ The Provenance Vocabulary Core Ontology Specification is available at <http://trdf.sourceforge.net/provenance/ns.html>.

thus supporting the interlinking with relational data on the Data Web. Finally, we implemented a standalone Semantic Pingback server (also available as service), that can be utilized by arbitrary resources that do not provide a Pingback service themselves.

5.1 OntoWiki

OntoWiki [2]⁷ is a tool for browsing and collaboratively editing RDF knowledge bases. Since OntoWiki enables users to add typed links on external resources, we integrated a Semantic Pingback client component. A recently added feature is the ability to expose the data stored in OntoWiki via the Linked Data mechanism. Based on that functionality, a Semantic Pingback server component was also integrated.

OntoWiki Pingback client. The Pingback client consists of a plugin that handles a number of events triggered when statements are added or removed from the knowledge base. Each time a statement is added or removed, the plugin first checks, whether:

- the subject resource is a URI inside the namespace of the OntoWiki environment,
- the subject resource is (anonymously) accessible via the Linked Data mechanism⁸ and
- the object of the statement is a resource with an de-referenceable URI outside the namespace of the OntoWiki environment.

If the above steps are successfully passed, the plugin tries to autodiscover a Pingback server. This process follows the algorithm described in the original Pingback specification but adds support for target resources represented in RDF as described in section 3.1. If a server was discovered, an XML-RPC post request is send.

OntoWiki Pingback server. The OntoWiki Pingback server is an extension consisting of a plugin handling some request cycle related events, as well as a component that provides a Pingback XML-RPC service. The plugin is responsible for exposing the **X-Pingback** HTTP-header in conjunction with the URL of the RPC service.

The provided Pingback service initially checks, whether the target resource is valid, i.e. is inside the namespace of the OntoWiki environment and accessible via the Linked Data mechanism. If a valid target resource was passed, the service takes the following steps:

⁷ <http://ontowiki.net>

⁸ This step is added to the process since OntoWiki is able to handle various access control mechanisms and we thus ensure that the Pingback server of the target resource is definitely able to access either the RDF or the (X)HTML representation of the source resource.

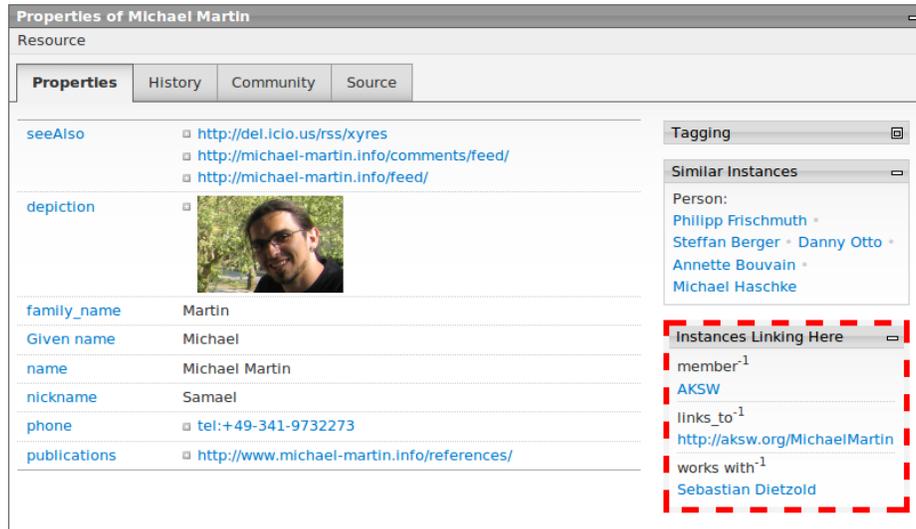


Fig. 3. OntoWiki backlinks are rendered in the "Instances Linking Here" side box. The example visualizes a personal WebID with three different backlinks using different relations.

1. The server tries to request the target resource as RDF/XML. If an RDF/XML document is retrieved, all relevant triples are extracted.
2. If the above step fails or no relevant triples are found, the OntoWiki Pingback server utilizes a configurable RDFa extraction service (e.g. the W3C RDFa Distiller⁹), which dynamically creates an RDF/XML representation from a target Web page.
3. If the second step fails, the target resource is requested without an additional **Accept**-header. If an HTML document is retrieved, all links in the document are checked. If a link to the target resource is found, a generic triple with the property `sioc:links_to` is formed together with the source as subject and the target resource as object.

Relevant triples are all triples that have either the source resource as subject and the target resource as object or vice versa. If no such statements were found, but the graph contains at least one statement that has the target resource as subject, a `rdfs:seeAlso` link is established from target resource to source resource.

All relevant statements are added to the knowledge base containing the target resource. By using the versioning functionality of OntoWiki, provenance information of statements added via Pingback requests can be determined, thus allowing the service to delete statements that are no longer contained by the source resource.

⁹ <http://www.w3.org/2007/08/pyRdfa/>

Backlinks that were established via the Pingback service are displayed in the standard OntoWiki user interface. The "Instances Linking Here" box shows all incoming links for a given resource in conjunction with the type of the link, as visualized in figure 3.

5.2 Triplify

Triplify [1] enables the publication of Linked Data from relational databases. It utilizes simple mappings to map HTTP-URLs to SQL queries and transforms the relational result into RDF statements. Since a large quantity of currently available web data is stored in relational databases, the number of available Linked Data resources increases. As people start to link to those resources, it becomes handy to notify the respective owner. Therefore, we integrated a Semantic Pingback server into Triplify, which exposes an **X-Pingback** HTTP header and handles incoming RPC requests.

The RPC service creates a new database table and stores all registered Pingbacks persistently. Pingbacks are unique for a given source, target and relation and hence can be registered only once. Each time the Pingback service is executed for a given source and target, invalid Pingbacks are removed automatically.

Triplify was extended to export statements for all registered Pingbacks regarding a given target resource along with the instance data. The following listing shows an excerpt of a Triplify export:

```
1 # ...
2
3 <post/1>
4   a sioc:Post ;
5   sioc:has_creator <user/1> ;
6   dcterms:created "2010-02-17T05:48:11" ;
7   dcterms:title "Hello world!" ;
8   sioc:content "Welcome to WordPress. This is your..." .
9
10 # ...
11
12 <http://blog.aksw.org/2008/pingback-test/>
13   sioc:links_to <post/1> .
```

5.3 Standalone implementation

Since a large amount of available RDF data on the Web is contained in plain RDF files (e.g. FOAF files), we implemented a standalone Semantic Pingback server¹⁰, that can be configured to allow Pingbacks also on external resources. Based on this implementation, we offer a Semantic Pingback service at: <http://pingback.aksw.org>. It is sufficient to add an RDF statement to an arbitrary web-accessible RDF document stating that the AKSW Pingback service should

¹⁰ Available at: <http://aksw.org/Projects/SemanticPingBack>

be used employing the `pingback:service` property. Once a Pingback was sent to that service, the owner of the document gets notified via email. This works well for FOAF profiles, since the service can detect a `foaf:mbox` statement in the profile, which relates the WebID to a `mailto:-URI`. If no such statement is found, the service looks for statements that relate the target resource via a `foaf:maker`, `dc:creator`, `sioc:has_creator` or `sioc:has_owner` relation to a resource for which an email address can be obtained.

6 Related Work

Pingback [9] is one of three approaches which allow the automated generation of backlinks on the Social Web. We have chosen the Pingback mechanism as the foundation for this work, since it is widely used and less prone to spam than for example Trackbacks¹¹. Pingback supports the propagation of untyped links only and is hence not directly applicable to the Data Web.

The PSI BackLinking Service for the Web of Data¹² 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.org¹³ [7]. Other than the Semantic Pingback it crawls the Web of Data in order to determine URIs describing the same resources. OKKAM [3] is a system that aims at unifying resource identifiers by employing metadata about resources in order to match them on entities.

The above approaches support interlinking of resources 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. Consequently the primary goal of these approaches is to reveal resource identifiers describing the same entities, rather than interlinking different resources - a key feature of the Semantic Pingback approach.

7 Conclusion and Future Work

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 quality, timeliness, direct end-user benefits are still obstacles for the Data Web to become an Web-wide reality. With this work we aimed at extending and transferring the technological cornerstone of the Social Web the Pingback mechanism towards the Data Web. The resulting Semantic Pingback mechanism has the potential to significantly improve

¹¹ http://www.sixapart.com/pronet/docs/trackback_spec

¹² <http://backlinks.psi.enacting.org>

¹³ <http://sameas.org>

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 Pingbacks represent direct end-user benefits and increase the timeliness. 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.

Future Work. Currently the Semantic Pingback mechanism is applicable to relatively static resources, i.e. RDF documents or RDFa annotated Web pages. We plan to extend the Semantic Pingback mechanism in such a way, that it is also usable in conjunction with dynamically generated views on the Data Web - i.e. SPARQL query results. This would allow end-users as well as applications using remote SPARQL endpoints to get notified once results of a query change.

References

1. S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify – Lightweight Linked Data Publication from Relational Databases. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2009*, 2009.
2. S. Auer, S. Dietzold, and T. Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006*, 2006.
3. 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.
4. J. Breslin, A. Harth, U. Bojars, and S. Decker. Towards Semantically-Interlinked Online Communities. In *The Semantic Web: Research and Applications Second European Semantic Web Conference, ESWC 2005*, 2005.
5. 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.
6. 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.
7. 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.
8. O. Hartig. Provenance Information in the Web of Data, 2009. LDOW2009, April 20, 2009, Madrid, Spain.
9. S. Langridge and I. Hickson. Pingback 1.0. Technical report, <http://hixie.ch/specs/pingback/pingback>, 2002.