Publish and Subscribe for RDF in Enterprise Value Networks

Marvin Frommhold
Agile Knowledge Engineering and Semantic Web
Institute of Computer Science
University of Leipzig, Germany
frommhold@informatik.uni-leipzig.de

Natanael Arndt
Agile Knowledge Engineering and Semantic Web
Institute of Computer Science
University of Leipzig, Germany
arndt@informatik.uni-leipzig.de

Sebastian Tramp
eccenca GmbH
Hainstr. 8
04109 Leipzig, Germany
sebastian.tramp@eccenca.com

Niklas Petersen
Enterprise Information Systems
Institute for Applied Computer Science
University of Bonn, Germany
petersen@cs.uni-bonn.de

ABSTRACT

Sharing information securely between business partners and managing large supply chains efficiently will be a crucial competitive advantage for enterprises in the near future. In this paper, we present a concept that allows for building value networks between business partners in a distributed manner. Companies are able to publish Linked Data which participants of the network can clone and subscribe to. Subscribers get notified as soon as new information becomes available. This provides a technical infrastructure for business communication acts such as supply chain communication or master data management. In addition to the conceptual analysis, we provide an implementation enabling companies to create such dynamic semantic value networks.

CCS Concepts

•Software and its engineering → Publish-subscribe / event-based architectures; •Computer systems organization → Client-server architectures; •Information systems → Resource Description Framework (RDF);

Keywords

Access control, Change propagation, Linked data, Publish and subscribe, RDF, Replication

1. INTRODUCTION

Empowering enterprises to share their data securely using semantic vocabularies and allowing them to actively push new information (updates) to connected business partners is the main goal of the LUCID research project[1]. Being able to create dynamic semantic value networks improves the sharing of information between business partners as well as enhances the management of complex supply chains. Mastering such value-added supply chains is a critical success factor, especially in the context of the recent Industry 4.0 movement [7].

For example, the SCORVoc vocabulary presented in [14] provides an RDF representation of the cross-industry Supply Chain Operation Reference (SCOR). It aims at reducing the complexity in supply chain management through semantic clarity. However, the tools currently available on the market struggle meeting the demands of secure and proactive distribution of crucial KPI (key performance indicator) information, especially serialized in RDF, among business partners.

In this paper, we present the concept and implementation of such a tool based on a publish-subscribe approach. It allows enterprises to publish RDF data as well as to subscribe to published data of other business partners to get notified as soon as new information becomes available. Authentication and authorization mechanisms are included to guarantee secure access to the published data. Furthermore, our proposed solution includes a versioning component for RDF change detection and an efficient propagation of changing data. Additionally, provenance information is added to the exchanged data assuring a minimum level of trust.

This paper is structured as follows: We give an overview of the challenges and the resulting requirements in section 2. The concept of our approach is presented in section 3, followed by the implementation in section 4. We report on related work in section 5 before we conclude with an outlook on future work in section 6.

2. CHALLENGES

We now present the challenges we have to tackle in the context of the LUCID research project. For each challenge, we provide a set of specific requirements demanded by busi-

WWW2016 Workshop: Linked Data on the Web (LDOW2016)

[1]http://www.lucid-project.org/

ness partners to enable them to share their data in dynamic value networks.

Open Standards.

As value networks should enhance the data distribution in terms of experience and reliance between the participants, it is crucial to rely on open, lightweight, and in case of the data, semantic standards. This ensures interoperability and allows every business partner to adapt its custom applications to be able to connect to other participants of a value network and share its data. The Resource Description Framework (RDF) [8], whose main goal is to provide a "vendor-neutral and operating system-independent system of metadata"[2], fulfills our requirements on interoperability and participation and thus was chosen as our data model.

On Demand Communication.

Each participant of a value network should be able to act as a source of data (publisher) as well as a consumer of data (subscriber). As participants of a value network connect or disconnect to each other on demand, there must be a specified mechanism to subscribe to or unsubscribe from the data of a publisher. A publisher must be able to inform all subscribers that its data has changed as soon as possible. This results in the need for a tool to publish RDF data and a service to manifest one's interest in updates of an RDF data publisher.

Provenance.

A value network must guarantee the traceability of the published data to fulfill the notion of provenance[3]. All participants of a value network must be able to trace back the origin of the exchanged data. The authenticity and integrity of the data must be ensured, emphasizing the need for protection against unperceived manipulation of the data. Participants must be able to sign the exchanged data to increase the level of trust. These considerations lead to the requirements of a shared RDF vocabulary for provenance information and a tool to sign and check the validity of RDF data.

Access Control.

The security of the data and secure access to the data are essential in the enterprise context. A publisher must have the ability to exchange its data under restricted conditions. For this reason, authentication and authorization mechanisms with the possibility to integrate custom solutions must be provided. When two participants connect to each other, the subscriber is authenticated and authorized. Therefore, publishers of the value network need a tool to authorize access to their RDF data.

Change Detection.

Publishers should be able to detect changes made to their data and share the change information with their subscribers. To enable sharing of any kind of RDF data, the change detection must support all features of RDF. Due to the semantics of blank nodes, special emphasis needs to be taken to guarantee their addressability outside the original graph [9].

[2]https://www.w3.org/Press/RDF

[3]https://www.w3.org/2005/Incubator/prov/XGR-prov-20101214/

In detail, participants need a shared RDF vocabulary for tracking changes to RDF data and a tool to detect changes made to RDF datasets.

The RDF data model enables a diversified and flexible representation of data. The concept of RDF named graphs allows to group parts of the data into logical units enabling different views on the data. Based on our experience, RDF named graphs represent the easiest manageable unit. Therefore, we only consider RDF named graphs as central view on the data regarding access, change tracking and publication.

3. PUBLISH AND SUBSCRIBE FOR RDF

As presented before, our main motivation is the on demand proactive notification of changes made to specific sets of RDF data in a distributed communication network. That means, instead of asking the data source for changes in a regular interval (pull), the interested entity is actively notified about any change (push). To realize this approach, we rely on PubSubHubbub [6], an open, decentralized publish-subscribe protocol. Adapted to our requirements of propagating changes made to RDF graphs, the workflow is defined as follows:

Subscribe: An interested entity (subscriber) sends a subscription containing the URI of the RDF graph of interest and a callback URL to the advertised server (hub) of the owner of the RDF dataset (publisher).

Publish: In case of a change to the dataset, the publisher informs its hub about the affected RDF graphs.

Notify: The hub looks up all subscriptions and notifies the affected subscribers by sending the changes to a graph as a diff (added and deleted triples).

We have designed this workflow to use the features of the eccenca DataPlatform[4] as it provides the base functionality helping us to address the mentioned challenges including context-based access control for RDF graphs and versioning for arbitrary RDF data. However, as noted in the challenges, the eccenca DataPlatform can be replaced with other tools providing the needed features. In the following, we present a more detailed overview of our concept with particular references to the challenges of section 2. We thereby only consider the subscription to and propagation of changes to RDF graphs. A priori reading or importing of published data on subscriber's side is not in the scope of our concept.

3.1 Subscription

To get notified about changes made to a specific graph of a dataset, the subscriber sends a subscription request to the hub of the publisher as described in section 5 of [6] with some specific adjustments to support our approach. The value of the `hub.topic` request parameter must be the URI of the graph the subscriber wants to get notified about changes. To tackle the challenge of access control, the protocol's optional subscription validation is a mandatory step in our communication. A subscription is only valid if the subscriber is successfully authenticated and authorized to read the requested graph. In case of a valid subscription, the hub saves the subscription in a persistent storage. Other features such as the

[4]https://www.eccenca.com/en/products/linked-data-suite.html

subscription process (subscription response and verification) as well as unsubscribing are implemented as described in the PubSubHubbub specification.

Access Control.
To be able to validate if a subscriber has the permission to subscribe to a specific RDF graph, it needs to authenticate itself. We rely on the eccenca DataPlatform using OAuth 2.0[5] for authentication with support for various authentication providers such as an LDAP[6] directory server. To represent the access control rules, we use the eccenca Authorization Vocabulary[7]. It allows dataset publishers to define context-based access conditions (`eccauth:AccessCondition`) on the level of RDF graphs. A subscriber is authorized to read and therefore allowed to subscribe to a specific graph if the graph URI is contained in the set of readable graphs defined by the property `eccauth:readGraph` of all fulfilled access conditions. An access condition is fulfilled, if all required attributes attached to it (by sub-properties of `eccauth:requiresAttribute`) are met by the session of the authenticated subscriber.

3.2 Publication

When a dataset is updated, all designated hubs need to be informed about the changes. For this, the publisher requires a change detection system for RDF data that, according to the challenges, must support any RDF dataset even in the presence of blank nodes and a vocabulary to describe the changes in RDF. We use the change detection system of the eccenca DataPlatform that creates a patch for each updated graph containing the added and removed triples and additional meta-information. Additionally, patches created by the system provide full blank node support by adding the context for each added or removed triple containing blank nodes. Blank node support plays an important role considering that over 50% of published datasets contain blank nodes [9]. To tackle the challenge of trust, a publisher can optionally certify a patch. This is done by signing its hash value and then attaching the digital signature to the patch by, for instance, using the Cert Ontology[8]. The patches are send separately to the advertised hub(s) of the publisher.

Versioning.
To be able to notify the hub(s) about changes, the publisher needs to detect changes made to its dataset. To achieve this, we use the versioning functionality we have developed as part of the eccenca DataPlatform. For each update, the system detects the added or removed triples and creates one patch per graph. The patches are represented as instances of `eccrev:Commit` of the eccenca Revision Vocabulary[9]. The vocabulary is based on the structure of the Delta ontology[10] and reuses concepts of the PROV-O ontology[11]. By providing the author (`eccrev:commitAuthor`), change reason (`eccrev:commitMessage`) and a time stamp (`prov:atTime`) we argue that such a patch fulfills the notion of provenance.

[5]http://tools.ietf.org/html/rfc6749
[6]https://tools.ietf.org/html/rfc4511
[7]https://vocab.eccenca.com/auth/
[8]http://www.w3.org/ns/auth/cert#
[9]https://vocab.eccenca.com/revision/
[10]https://www.w3.org/DesignIssues/Diff
[11]https://www.w3.org/TR/prov-o/

Moreover, the system calculates a hash value for each patch (`eccrev:sha256`) using the algorithm described in [3]. To allow for attaching arbitrary meta-information to a patch, our system limits the properties used for hash calculation to a fixed set. This hash value allows for verifying the integrity of a patch. As each patch contains a reference and a hash value of its predecessor, the system offers protection against unperceived history manipulation.

3.3 Notification

For notification, we rely on the PubSubHubbub content distribution request as described in section 7 of [6] with specific modifications. For each patch received from a publisher, the hub must perform an authorization check ensuring the validity of a subscription, which includes a check if the subscriber is still allowed to read the graph referenced in the patch. If the subscription is valid, the hub sends a notification to the affected subscriber's callback URL, else the subscription is canceled. The patch is serialized into an official RDF serialization format and added as body of the notification. The `Content-Type` header must be set according to the used serialization format. Sending only a patch instead of the complete changed RDF graph is generally more space efficient and a client does not have to perform the delta calculation by itself. Before applying the patch, the subscriber must validate the integrity of the received patch by using the same hash calculation algorithm as used by the publisher. If the integrity of a patch is valid, the subscriber applies the patch to its copy of the appropriate graph as follows: (i) remove all triples marked as deleted; and (ii) insert all triples marked as added. This order ensures the correct application of the patch, especially if changes involving blank nodes were made . Optionally, the subscriber can save the patch to be able to retrace the change history at a later date.

Due to space limitations, we do not discuss security issues for the communication between the subscribers, publishers and hubs. However, we want to refer to the usage of HTTPS as described in [6] which is fully compatible with our proposed concept.

4. IMPLEMENTATION

The proposed publish-subscribe concept is integrated into the eccenca DataPlatform which is a Java Spring[12] application acting as a proxy in front of one or more RDF repositories. Thus, it enables consistent publication of RDF data across various repositories according to the five-star deployment scheme[13].

In our implementation the publisher, subscriber and hub are part of the eccenca DataPlatform. The communication between these three components is realized through the use of Spring Integration[14]. This enables a lightweight messaging within the application and further supports the integration into external systems as well. In our case we use Apache Kafka[15] as messaging server to ensure scalability, stability and backup of the communication within the application.

An architectural overview of our implementation is depicted in Figure 1. An exemplary workflow starts with a new subscription message from the subscriber to its hub (1). The

[12]https://projects.spring.io/spring-framework/
[13]http://www.w3.org/DesignIssues/LinkedData.html
[14]http://projects.spring.io/spring-integration/
[15]http://kafka.apache.org/

hub then authenticates itself in the name of the subscriber at the advertised hub of the publisher and sends a PubSubHubbub subscription request (2). The hub of the publisher hands over the subscription request for validation to the publisher (3) that validates the authorization of the subscriber with the help of the access control component (4). In case of success, the publisher informs its hub (5) that verifies the subscription (6) before storing it in the local RDF repository (7). If an update is made to the subscribed data (8), the versioning component creates a patch (9) and informs the publisher about the change (10). The publisher sends a change notification to its hub (11) that validates for each affected subscription if it is still authorized to get updates (12). In the positive case, the hub sends a PubSubHubbub content distribution request to the affected subscriber hubs (13), else the subscription is canceled. The subscriber hub now informs its subscriber (14) that checks the integrity of the patch before applying it to its data (15). When applying patches, especially in the presence of blank nodes, it must be ensured, with respect to non-lean graphs and isomorphism, that the correct triples get removed.

5. RELATED WORK

Various approaches have been presented in the area of change propagation for RDF data. The proposed publish and subscribe systems can be divided into two categories: distributed peer-to-peer systems [2, 4, 16, 13] and client-server systems [18, 12, 15, 17, 1, 5].

In general, the peer-to-peer approaches are based on broadcasting subscriptions and change notifications between the participants of the network. Peers able to contribute to a subscription broadcast their changed data through the network, whereby changed data gradually arrives at the subscriber. Due to the requirement of a secure communication of the data, peer-to-peer systems are not rejected per se, however, all publications rely on approaches distributing the data freely readable between the network participants. Therefore this type of architecture is not advisable for enterprise value networks where sensitive data is exchanged.

In contrast to peer-to-peer systems, client-server systems provide direct real-time notification allowing a secure change propagation between the participants. RDFSync [18] provides an update synchronization for RDF data similar to the rsync utility with support for blank nodes by considering their context. We've used the RDFSync algorithms in our versioning component to support blank nodes. However, their approach uses pull requests with a lack of real-time notifications in case of a change event. DSNotify [15] offers a framework to notify about broken links in two datasets, but the subscriber needs to detect the changes by itself. The proposed sparqlPuSH system of [12] uses the PubSubHubbub protocol to inform about changed RDF data based on SPARQL query subscriptions. The main disadvantage of their approach is the execution of all subscription queries against the data when an update occurs, which can lead to performance problems in case of a large dataset and many subscriptions. The approaches of [17, 1] also use PubSubHubbub as communication protocol for the synchronization of RDF data. Unfortunately, these systems are limited to specific RDF structures, e.g. resources describing social network activities, not supporting arbitrary RDF data including blank nodes. The approach of [5] based on the iRap framework provides a way to subscribe to changes based on a subset of SPARQL queries. Their system is similar to ours, however, due to their restriction of possible queries, users are not able to subscribe to changes of specific graphs.

The work of [10, 6, 11] provide specifications for publish and subscribe protocols addressing the notification issue. Notifications about events are pushed to interested clients via hubs. The PubSubHubbub specification [6] is simple, well documented and provides a detailed installation description and thus was chosen for our approach.

6. CONCLUSIONS AND FUTURE WORK

By empowering enterprises to create dynamic semantic value networks based on open standards for publishing data, they are able to improve the sharing of information between their business partners and to enhance the management of complex supply chains. In this paper, we presented a concept allowing the creation of such value networks. We used the existing components for access control and versioning of RDF data of the eccenca DataPlatform and integrated the publish-subscribe approach to create a tool tackling the mentioned challenges.

However, first system deployments have yield to issues and questions hampering a wide deployment in enterprises without any concerns. First, there is a need for a more flexible solution regarding authentication, for example based on WebID[16]. For now business partners have to manually agree on and maintain the credentials used to authenticate among each other. Secondly, and of particular interest, is the question of how to deal with changing permissions, especially if a previous granted read access is withdrawn. Is it enough to just cancel the affected subscriptions or do publishers want to perform additional steps?

7. ACKNOWLEDGMENTS

This work was partly supported by the following grants from the German Federal Ministry of Education and Research (BMBF) for the LUCID Project (GA no. 01IS14019A) as well as for the the LEDS Project (GA no. 03WKCG11A and GA no. 03WKCG11C).

8. REFERENCES

[1] N. Arndt and S. Tramp. Xodx: A node for the distributed semantic social network. In *International Semantic Web Conference*, Aachen, Germany, Germany, 2014.

[2] M. Cai, M. Frank, B. Yan, and R. MacGregor. A subscribable peer-to-peer RDF repository for distributed metadata management. *Journal of Web Semantics*, 2004.

[3] J. J. Carroll. Signing RDF graphs. In *Internatinal Semantic Web Conference*. 2003.

[4] P.-A. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Publish/Subscribe for RDF-based P2P networks. In *The Semantic Web*. 2004.

[5] K. M. Endris, S. Faisal, F. Orlandi, S. Auer, and S. Scerri. Interest-Based RDF update propagation. In *International Semantic Web Conference*. 2015.

[16]https://www.w3.org/2005/Incubator/webid/spec/

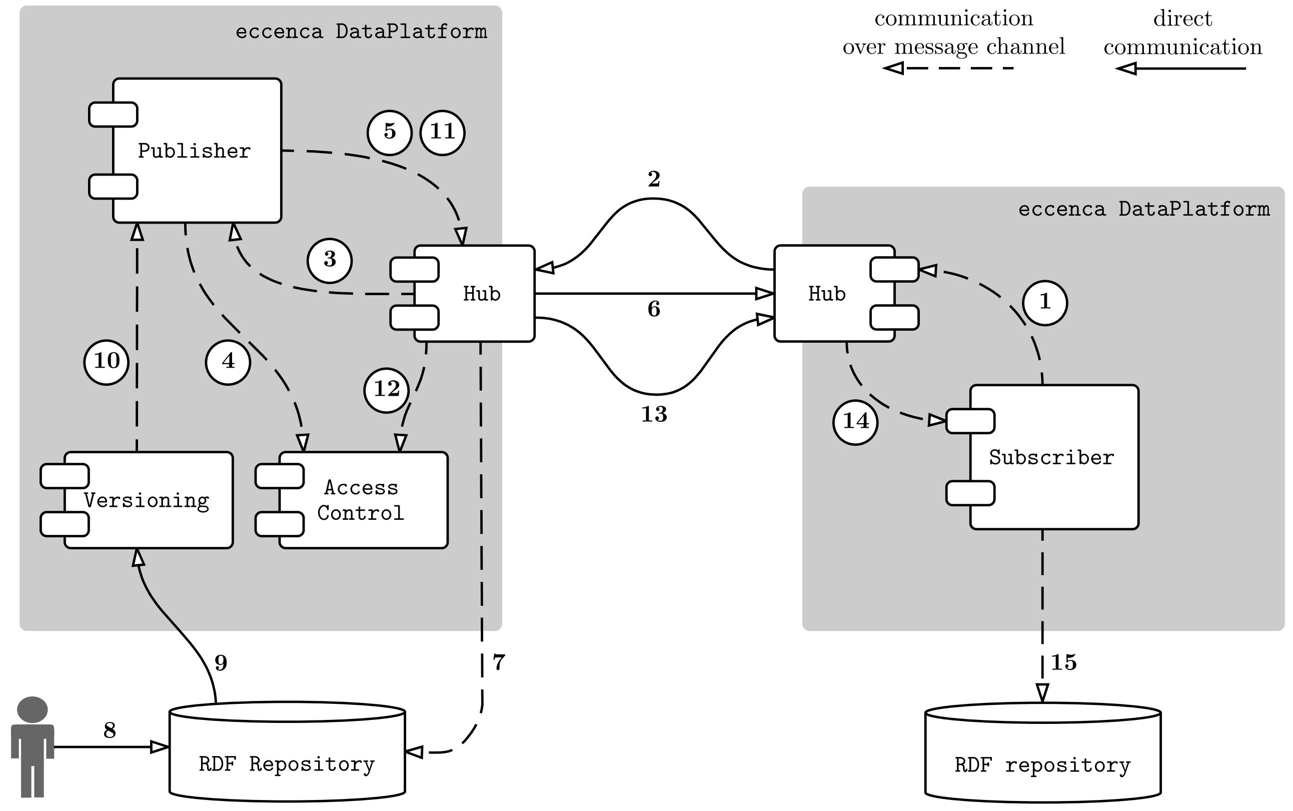

Figure 1: Architectural overview of our implementation.

[6] B. Fitzpatrick, B. Slatkin, M. Atkins, and J. Genestoux. PubSubHubbub core 0.4 – working draft. Technical report, 4 Feb. 2014.

[7] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster. Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of german manufacturing industry. Technical report, 2013.

[8] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. *W3C Recommendation*, 10 Feb. 2004.

[9] A. Mallea, M. Arenas, A. Hogan, and A. Polleres. On blank nodes. In *International Semantic Web Conference*. 2011.

[10] P. Millard, P. Saint-Andre, and R. Meijer. XEP-0060: Publish-Subscribe. Technical report, 12 July 2010.

[11] Object Management Group. Data distribution service (DDS) version 1.4. Technical report, 2015.

[12] A. Passant and P. N. Mendes. sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub. In *Extended Semantic Web Conference*, 2010.

[13] L. Pellegrino, F. Huet, F. Baude, and A. Alshabani. A distributed Publish/Subscribe system for RDF data. In *Data Management in Cloud, Grid and P2P Systems*. 2013.

[14] N. Petersen, I. Grangel-González, G. Coskun, S. Auer, M. Frommhold, S. Tramp, and M. LeFrancois. SCORVoc: Vocabulary-based information integration and exchange in supply networks. *IEEE International Conference on Semantic Computing*, 2016.

[15] N. P. Popitsch and B. Haslhofer. DSNotify: Handling broken links in the web of data. In *International World Wide Web Conference*, 2010.

[16] D. Ranger and J.-F. Cloutier. Scalable Peer-to-Peer RDF query algorithm. In *Web Information Systems Engineering*. 2005.

[17] S. Tramp, P. Frischmuth, T. Ermilov, S. Shekarpour, and others. An architecture of a distributed semantic social network. *The Semantic Web*, 2014.

[18] G. Tummarello, C. Morbidoni, R. Bachmann-Gmür, and O. Erling. RDFSync: Efficient remote synchronization of RDF models. In *The Semantic Web*. 2007.