visKWQL, a Visual Renderer for a Semantic Web Query Language

Andreas Hartl, Klara Weiand, François Bry

Institute for Informatics, University of Munich
Oettingenstr. 67, 80538 München, Germany
http://pms.ifi.lmu.de

Abstract. Querying a Wiki must be simple enough for beginning users, yet powerful enough to accommodate experienced users. To this end, the keyword-based KiWi query language (KWQL) supports queries ranging from simple lists of keywords to expressive rules for selecting and reshaping Wiki (meta-)data. In this demo, we showcase visKWQL, a visual interface for the KWQL language aimed at supporting users in the query construction process. visKWQL and its editor are described, and their functionality is illustrated using example queries. The editor provides guidance throughout the query construction process through hints, warnings and highlighting of syntactic errors. The editor enables round-tripping between the twin languages KWQL and visKWQL, meaning that users can switch freely between the textual and visual form when constructing or editing a query. It is implemented using HTML, JavaScript, and CSS, and can thus be used in (almost) any web browser without any additional software.

1 Introduction

Web query languages like XQuery and SPARQL allow for the precise and targeted selection and transformation of Web data. While these languages are powerful tools, they require their users to be knowledgeable about the language itself as well as the structure and schema of the queried data. This requirement excludes a large part of the web's user base (and thus the potential user base of the Semantic Web) from the benefits of these languages and the functionality they provide.

Visual languages have two advantages over textual languages that specifically benefit beginning users [2]: First, their visual structure can make them easier to learn and understand than textual languages. Secondly, editors for visual languages can support users in the creation of valid queries by providing guidance and preventing editing operations that would result in incorrect queries.

This demonstration presents visKWQL[1], a visual query interface for the Semantic Wiki KiWi [4]. visKWQL is not so much a separate query language but

[1] A detailed demonstration description as well as a demo of visKWQL are available at http://www.pms.ifi.lmu.de/visKWQL/

rather a visual rendering of KWQL, the keyword-based KiWi query language. KWQL [1] is a rule-based query language based on the label-keyword paradigm that combines a low entry barrier with powerful querying and ease of use of keyword search with advanced features and capabilities as used in traditional query languages in order to accommodate users with varying levels of expertise.

visKWQL aims at extending textual KWQL—which itself has been designed to be easy to use—to achieve two cohesive and tightly integrated querying modi in the KiWi Wiki and enable user-friendly and powerful querying.

The work described here has been presented before as a demonstration at the 2010 WWW conference [3].

2 visKWQL

visKWQL provides a visual alternative to textual KWQL. It fully supports KWQL in that every KWQL query can be expressed as an equivalent visKWQL query. Further, in order to avoid introducing additional constructs and thus additional complexity, visKWQL stays close to the textual language in its visual representation.

2.1 Visual Formalism

visKWQL uses a form-based approach, in which all KWQL elements are represented as boxes, and associations between them are represented as nestings (see Figure 1 for an example). Boxes consist of a *label*, in which the name of the represented KWQL element is included, and a *body*, which can hold child boxes.

This approach stays close to KWQL's textual structure, making it easier to learn visKWQL and to translate between the two representations; it also lends itself well to rendering in HTML.

2.2 Round-tripping

One of the key features of visKWQL is round-tripping, to achieve a tight coupling between KWQL and visKWQL.

Whenever the user makes a change to the visual query, the change is immediately represented in the textual version. The textual query can further be edited and parsed by the system to display it in its visual form.

This allows the user to make changes in the representation of his choice at any time during the query construction process, to import KWQL queries easily into visKWQL, and has the additional benefit of teaching the user KWQL while he experiments with the visKWQL editor.

2.3 User Guidance

User support in visKWQL is provided via tooltips, error prevention, and error and problem display and correction.

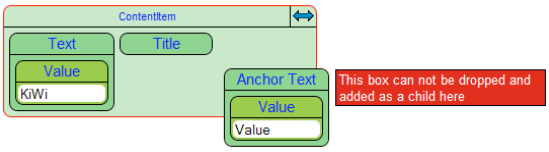


Fig. 1. Information hiding and error prevention

Tooltips: A text area below the workspace displays an explanation of the KWQL element represented by the box currently under the mouse cursor.
Error Prevention: A large number of syntactic errors result from invalid box nestings, and can be actively prevented by the editor during drag and drop actions. When a box is being dragged, the system continuously checks the validity of a child inclusion or a type switch with the box underneath it.

If dropping the box in its current location would result in a syntactic error, the border of the box underneath it is colored red, and a tooltip informs the user that he may not drop the box (see Figure 1).
Error Reporting and Correction: Some errors cannot be prevented during editing. These include variable names or values containing invalid characters, empty strings, misplaced operators and references to undefined variables.

After every user action, the query is checked for such errors. When an error is found, the label of the node is colored red and a tooltip indicating the error is displayed next to it.To make these errors easy to locate within the query, even if the erroneous node is currently hidden, the labels of all its parent boxes will also be colored red, and display a tooltip that a child box contains an error.

Errors that are less severe and can be corrected automatically, like empty boxes, cause the box label to be colored orange. A tooltip and a message below the workspace inform the user about the source of the problem.

3 Acknowledgements

The research leading to these results is part of the project "KiWi - Knowledge in a Wiki" and has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 211932.

References

1. F. Bry and K. Weiand. Flavours of KWQL, a keyword query language for a semantic wiki. In *Proceedings of SOFSEM 2010*, 2010.
2. T. Catarci, M. Costabile, S. Leviladi, and C. Batini. Visual Query Systems for Databases: A Survey. *Journal of Visual Languages and Computing*, 8(2), 1997.
3. A. Hartl, K. A. Weiand, and F. Bry. visKWQL, a visual renderer for a semantic web query language. In *WWW*, 2010.
4. S. Schaffert, J. Eder, S. Grünwald, T. Kurz, and M. Radulescu. Kiwi - a platform for semantic social software. In *ESWC*, 2009.

Not So Creepy Crawler: Crawling the Web with XQuery

Franziska von dem Bussche[1], Klara Weiand[1], Benedikt Linse[1], Tim Furche[1,2], François Bry[1]

[1] Institute for Informatics, University of Munich
Oettingenstr. 67, 80538 München, Germany
[2] Oxford University Computing Laboratories, Parks Rd, Oxford OX1 3QD, UK

Abstract. Web crawlers are increasingly used for focused tasks such as the extraction of data from Wikipedia or the analysis of social networks like last.fm. In these cases, pages are far more uniformly structured than in the general Web and thus crawlers can use the structure of Web pages for more precise data extraction and more expressive analysis.
In this demonstration, we present a focused, structure-based crawler generator, the *"Not so Creepy Crawler"* (NC^2). What sets NC^2 apart, is that all analysis and decision tasks of the crawling process are delegated to an (arbitrary) XML query engine such as XQuery or Xcerpt. Customizing crawlers just means writing (declarative) XML queries that can access the currently crawled document as well as the metadata of the crawl process. We identify four types of queries that together suffice to realize a wide variety of focused crawlers.

1 Introduction

In this demonstration, we present the *"Not so Creepy Crawler"* (NC^2), a novel approach to structure-based crawling that combines crawling with standard Web query technology for data extraction and aggregation. NC^2 differs from previous approaches to crawling in that it allows for high level of customization throughout every step in the crawling process. The crawling process is entirely controlled by a small number of XML queries written in any XML query language: some queries extract data (to be collected), some links (to be followed later), some determine when to stop the crawling, and some how to aggregate the collected data.

This allows easy, but flexible customization through writing XML queries. By virtue of the loose coupling between an XML query engine and the crawl loop, the XML queries can be authored with standard tools, including visual pattern generators [1]. In contrast to data extraction scenarios, these same tools can be used in NC^2 for authoring queries of any of the four types mentioned above.

A demonstration of NC^2, accessible online at `http://pms.ifi.lmu.de/ncc`, showcases two applications: The first extracts data about cities from Wikipedia with a customizable set of attributes for selecting and reporting these cities. It illustrates the power of NC^2 where data extraction from Wiki-style, fairly

homogeneous knowledge sites is required. The second use case demonstrates how easy NC^2 makes even complex analysis tasks on social networking sites, exemplified by last.fm.

NC^2 has already been presented at the 2010 WWW conference [3].

2 Crawling with XML Queries

"Not So Creepy Crawler": Architecture. The basic premise of NC^2 is easy to grasp: A crawler where all the analysis and decision tasks of the crawling process are delegated to an XML query engine. This allows us to leverage the expressiveness and increasing familiarity of XML query languages and provide a highly configurable crawler generator, which can be configured entirely through declarative XML queries.

To this end, we have identified those analysis and decision tasks that make up a focused, structure-based crawler, together with the data each of these tasks requires.

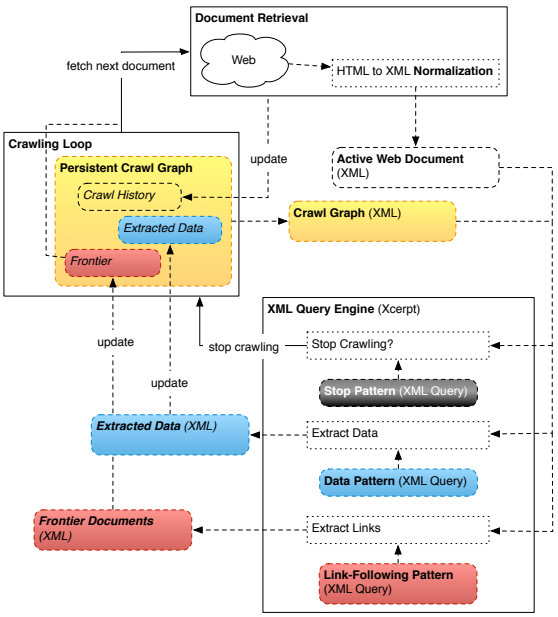


Fig. 1. Architecture NC^2

XML patterns. Central and unique to a NC^2 crawler is uniform access to both object data (such as Web documents or data already extracted from previously crawled Web pages) and metadata about the crawling process (such as the time and order in which pages have been visited, i.e., the crawl history). Our *crawl graph* not only manages the metadata, but also contains references to data extracted from pages visited previously. The tight coupling of the crawling and extraction process allows us to retain only the relevant data from already crawled Web documents.

This data is queried in a NC^2 crawler by three types of XML queries (see Figure 1):

(1) *Data patterns* specify how data is extracted from the current Web page. A typical extraction task is "extract all elements representing events if the current page or a page linking to it is about person X". To implement such an extraction task in a data pattern, one has to find an XML query that characterizes "elements representing events" and "about person X". As argued above, finding such queries is fairly easy if we crawl only Web pages from a specific Web site such as a social network.

(2) *Link-following patterns* extract all links from the current Web document that should be visited in future crawling steps (and thus be added to the crawling frontier). Often these patterns also access the crawl graph, e.g., to limit the

crawling depth or to follow only links in pages directly linked from a Web page that matches a data pattern.

(3) *Stop patterns* are boolean queries that determine when the crawling process should be halted. Typical stop patterns halt the crawling after a given amount of time (i.e., if the time stamp of the first crawled page is long enough in the past), number of visited Web pages, number of extracted data items, or if a specific Web page is encountered.

There is one more type of pattern, the *result pattern*, of which there is usually only a single one: It specifies how the final result document is to be aggregated from the extracted data. Once a stop pattern matches and the crawling is halted, the result pattern is evaluated against the crawl graph and the extracted data, e.g., to further aggregate, order, or group the crawled data into an XML document, the result of the crawling.

All four patterns can be implemented with any XML query language. In this demonstration we use Xcerpt [2] and XQuery.

System components. How are these patterns used to steer the crawling process? Crawling in NC^2 is an iterative process. In each iteration the three main components work together to crawl one more Web document (see Figure 1):

(1) The *crawling loop* initiates and controls the crawling process: It tells the document retrieval component to fetch the next document from the crawling frontier (the list of yet to be crawled documents).

(2) The *document retrieval* component retrieves and normalizes the HTML document and tells the crawling loop to update the crawl history in the crawl graph (e.g., to set the document as crawled and to add a crawling timestamp).

(3) The *XML query engine* (in the demonstrator, Xcerpt) evaluates the stop, data, and link-following patterns on both the active document and the crawl graph (containing the information which data patterns matched on previously crawled pages and the crawl history). Extracted links and data are sent to the crawling loop which updates the crawl graph.

(4a) If none of the stop patterns matches the iteration is finished and crawling starts again with the next document in step (1), if there is any.

(4b) If one of the stop patterns matches in step (3), the crawling loop is signalled to stop the crawling. The XML query engine evaluates the result pattern on the final crawl graph and the created XML result document is returned to the user.

References

1. R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with Lixto. In *VLDB*, 2001.
2. F. Bry, T. Furche, B. Linse, A. Pohl, A. Weinzierl, and O. Yestekhina. Four lessons in versatility or how query languages adapt to the web. In *Semantic Techniques for the Web, The REWERSE Perspective*, *LNCS* 5500. Springer, 2009.
3. F. von dem Bussche, K. A. Weiand, B. Linse, T. Furche, and F. Bry. Not so creepy crawler: easy crawler generation with standard XML queries. In *WWW*, 2010.

Linked Open Services: Update on Implementations and Approaches to Service Composition

Barry Norton[1] and Reto Krummenacher[2] and Adrian Marte[2]

[1] AIFB, Karlsruhe Institute of Technology, Germany
[2] Semantic Technology Institute, University of Innsbruck, Austria
firstname.lastname@{kit.edu | sti-innsbruck.at}

Abstract. Linked Open Services are a principled attempt to guide the creation and exposure of online services, in order to enable productive use of, and contribution to, Linked Data. As a technical basis, they communicate RDF directly via HTTP-based interaction, and their contribution to the consumer's knowledge, in terms of Linked Data, is described using SPARQL-based constructs. In this demo we will show some of our latest results in the production of Linked Open Services and the applications in which they can be taken up.

1 Overview

In [2], we present the groundwork of an initiative called Linked Open Services (LOS!). Subsequent work on developing such RDF/SPARQL-driven services at the community site http://www.linkedopenservices.org has led to the refinement of a number of defining principles:

1. Describe services as *LOD prosumers* with input and output descriptions as *SPARQL graph patterns*.
2. *Communicate RDF* by *RESTful content negotiation*.
3. Communicate and describe the *knowledge contribution* resulting from service interaction, including *implicit knowledge* relating input, output and service provider.

Associated with the last principle, there is an optional fourth one:

4. When wrapping non-LOS services, extend the (lifted, if non-RDF) message to make explicit the implicit knowledge it contains, and use Linked Data vocabularies and *SPARQL CONSTRUCT* queries to derive the constructed knowledge.

In our demo we present LOS implementations, applications and first results towards the use of Linked Open Services in processes and service compositions. In Linked Open Services, we see an approach to linked data mash-ups with real world side-effects. It is in the nature of the work, and the call to which we respond, that this will be of 'late breaking nature'. We have already taken forward particularly the challenge of geospatial services; both in the development of novel services according to these principles, and in wrapping existing services from http://www.geonames.org.[1]

[1] http://www.linkedopenservices.org/services/geo/

J	J	J	J	J	J	J	J	–	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	–	J	J	J	J	X
X	X	X	X	X	X	–	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	–	X	X	X	X	X
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	R	–	–	–	–
astergdem	children	cities	countryCode	countryInfo	countrySubdivision	earthquakes	extendedFindNearby	findNearby	findNearbyPlaceName	findNearbyPostalCodes	findNearbyStreets	findNearbyStreetsOSM	findNearByWeather	findNearbyWikipedia	findNearestAddress	findNearestIntersection	findNearestIntersectionOSM	get	gtopo30	hierarchy	neighbourhood	neighbours	ocean	postalCodeCountryInfo	postalCodeLookup	postalCodeSearch	rssToGeo	search	siblings	srtm3	timezone	weather	weatherIcao	wikipediaBoundingBox	wikipediaSearch

Fig. 1. The current services offered over the geonames Linked Data set (J=JSON,X=XML,R=RDF)

2 Geospatial Services

While `geonames.org` is rightly regarded as one of the key components of the Linked Data cloud (cf. `www.linkeddata.org`), for the services that offer functionality over this data the use of semantics is much less encouraging, as shown in Figure 1. In fact, while all but one service are capable of using JSON (marked with a 'J'), rapidly replacing XML ('X') as the de facto language for service communication, only one is currently enabled to directly communicate RDF ('R').

A number of Linked Open Services are therefore dedicated to re-exposing these services in a form where they can communicate RDF. In order to do so we employ a library we have developed, and will demonstrate, named JSON2RDF. The point of this library is not to directly produce usable RDF, but rather to automatically produce some kind of RDF that can then be transformed into a useful form simply using SPARQL CONSTRUCT queries, as per Principle 4.

Table 1. Automatic JSON2RDF Lifting of GeoNames Weather Service

GeoNames output JSON:

```
{"weatherObservations": [
  { "clouds": "few clouds",
    "windDirection": 20,
    "ICAO": "CYOW",
    "lng": 41.7,
    "temperature": "35",
    ...
```

Generic RDF after lifting:

```
 [ json2rdf:weatherObservations [
     json2rdf:clouds "few clouds" ;
     json2rdf:windDirection "20"^^xsd:short ;
     json2rdf:ICAO airports:CYOW ;
     json2rdf:lng "41.7"^^xsd:double ;
     json2rdf:temperature "35" ;
     ...
```

Table 1 shows the result of applying JSON2RDF to a GeoNames weather service, while Table 2 sketches the CONSTRUCT query that is subsequently applied. Note that this is not only the opportunity to introduce the use of standard Linked Data vocabularies, but also of provenance information, which is increasingly becoming a 'hot topic' in the Linked Data community; cf. [1] for example.

Table 2. CONSTRUCT Query over GeoNames Weather Service

```
CONSTRUCT { [ met:weatherObservation [
  weather:hasStationID ?icao ;
  ...
  wgs84:long ?longitude ; wgs84:lat ?altitude ; ... ] ] .
}
WHERE { [ json2rdf:weatherObservations _:a ] .
  _:b json2rdf:ICAO ?icao ;
      json2rdf:lng ?longitude ; json2rdf:lat ?latitude ; ...
}
```

In our work with geospatial services we have also capitalised on the cross-fertilisation of Linked Data and RESTful service principles followed in Linked Open Services. Not only is the vocabulary for the airports that are used in weather reporting in geonames services missing, but the chance to view service calls as operations against the concerned resources, rather than as disembodied 'RPC calls', is missing. In response, Linked Open Services have been provided that openly manage spatial resources, such as those in the geonames dataset, and airports have been included so that the 'weatherICAO' service (where ICAO is a code identifying airports) is offered as an HTTP POST operation against a URI version of the ICAO code. The demo will also show how REST frameworks like JAX-RS can be used in developing such services in combination with semantic technologies.

3 Summary

In the demo we demonstrate our latest work on service development in the context of Linked Open Services [2]. The applications and services to be showcased focus foremost on our work with GeoNames data and services, and include demonstrations of our inclusion of provenance data and provision of resource-oriented services. We would also hope to present services related to the telecoms domain and demos supporting also the composition of Linked Open Services in the form of process.

References

1. Hartig, O., Zhao, J.: Publishing and Consuming Provenance Metadata on the Web of Linked Data. In: 3rd Int'l Provenance and Annotation Workshop (June 2010)
2. Krummenacher, R., Norton, B., Marte, A.: Towards Linked Open Services and Processes. In: 3rd Future Internet Symposium (September 2010)

ERP B3: Service Level Driven Management of On-Demand Business Support Systems. *

Ulrich Winkler [1], Daniel Playfair[1], and Wolfgang Theilmann[2]

SAP Research, SAP AG,
[1]The Concourse, Queen's Road, Belfast BT3 9DT, United Kingdom
[2]Vincent-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
{ulrich.winkler,daniel.playfair,wolfgang.theilmann}@sap.com
http://www.sap.com/research

Abstract. ERP B3 is a SLA@SOI framework based solution for Service Level Agreement driven, on demand and dynamic provisioned ERP systems. In this demonstration we want to show SLA management for hosted ERP systems from three perspectives; the customer, the sales, and IT administrator perspective. Aspects of the negotiation, planning and provisioning workflow, which involves all stakeholders, are outlined.

1 Introduction

Business support systems, such as Enterprise Resource Planning (ERP) systems are well established in large organisations and hosted on customer premise. However, the uptake for small and medium enterprises is still low due to high-complexity and high initial-costs of setup and maintenance. To offer ERP functionality as a Software-as-a-Service (SaaS) offering could be a solution. As ERP systems are business critical systems, customers should be enabled to specific Service Level Agreements (SLAs) which details penalties in case the EPR provider is not able to deliver ERP functionality as demanded.

The SLA@SOI project researches and develops a framework for SLA aware SaaS solutions [1, 2]. This framework provides (1) comprehensive support for holistic and transparent SLA management, SLA translation and SLA negotiation, (2) a means to predict service quality characteristics, (3) an automated service deployment apparatus and (4) mechanisms to monitor and to enforce service quality at runtime.

Lessons learned from utilising the SLA@SOI framework to provide on-demand business applications are discussed in [3]. The authors elaborate on details how the SLA@SOI framework is used to plan, translate and negotiate SLAs at different layers and explain technical and scientific particulars. However, the best way to illustrate this lengthy end-to-end planning, translation and negotiation workflow is with a demonstration.

* The research leading to these results is partially supported by the European Community's Seventh Framework Programme (FP7/2001-2013) under grant agreement no.216556.

2 ERP B3's SLA Negotiation and Translation Workflow

We anticipate three stakeholders in the negotiation and planning workflow; which are the customer, the sales officer and the IT administrator. For every stakeholder ERP B3 offers a tailored user front-end, called portal. The customer portal allows a customer to browse product offerings. The customer can initiate a quotation process and specify service level requirements. The sales portal provides functionality to manage customer requests, to plan business service level agreements and to perform price calculations. The IT administrator portal supports IT landscape planning and provide monitoring and adjustment functionality.

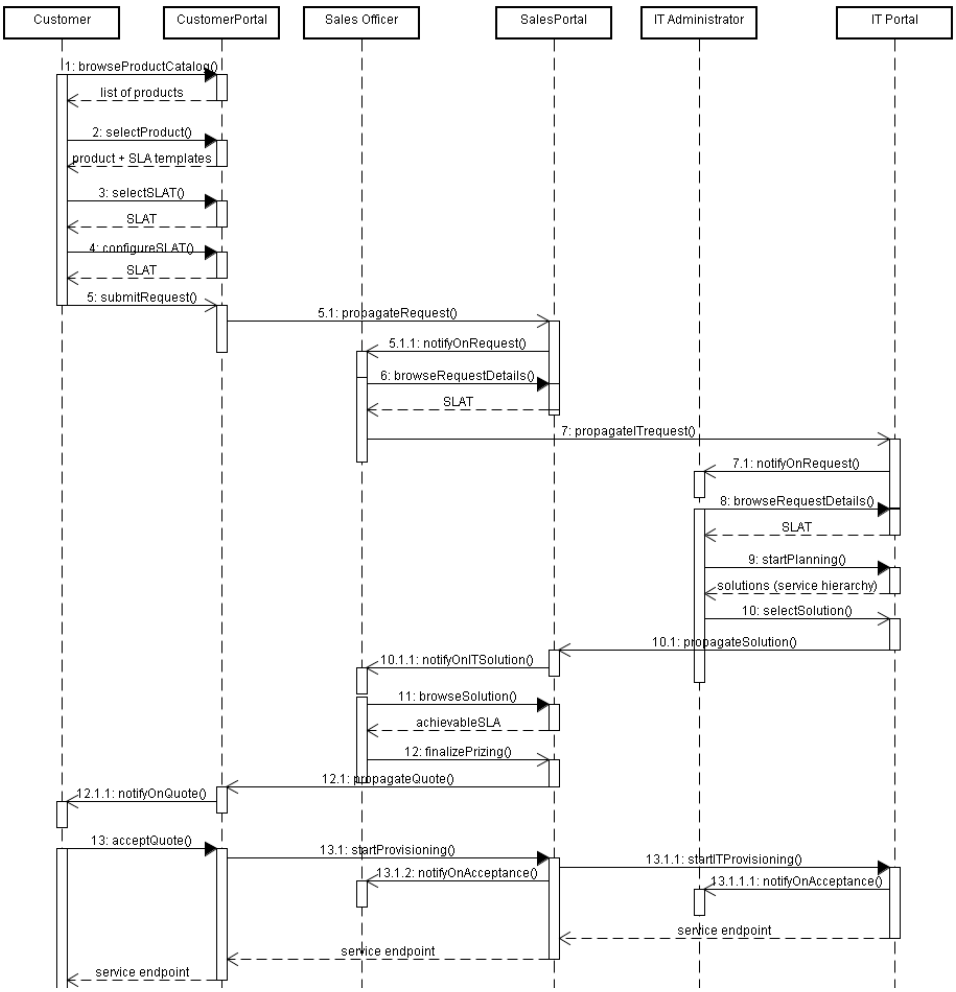


Fig. 1. The negotiation, planing and provisioning workflow.

The end-to-end negotiation, planning and provisioning workflow is depicted in Figure 1 and briefly discussed here:

The customer [steps 1-5.1] browses the product catalogue and select a product of interest. For every product the customer can choose from various pre-defined SLA templates, configures the selected template according to his business needs, and issues a request for quotation to the sales office.

The sales officer [steps 5.1.1-7] examines quotation request details and propagates the IT relevant parts to the IT administrator as an application service request.

The IT administrator [steps 7.1-9] receives the application service request and triggers the IT planning wizard [step 9 ff]. This wizard translates application service requests into a set of IT landscape plans, including middleware and infrastructure requirements. The IT administrator [steps 10-10.1] selects the most appropriate landscape plan. Once an IT landscape plan has been selected the sales officer can finalises the quote for the customer, i.e. she determines a final price.

The customer [step 13] accepts the quote. This triggers an automated provisioning process for the complete service hierarchy. The sales officer and IT administrator are notified accordingly.

3 Conclusions

Server virtualisation and cloud computing enable new kinds of service provisioning and management methodologies. The SLA@SOI framework supplements these methodologies with SLA management, which is required in a business context. ERP B3 makes use of this framework to provide service level aware on-demand business applications. In this paper we have shown the end-to-end negotiation, planning and provisioning workflow, which co-ordinates SLA management activities among various stakeholders.

References

1. SLA@SOI project: IST- 216556; Empowering the Service Economy with SLA-aware Infrastructures, `http://www.sla-at-soi.eu/`
2. Theilmann, W., Happe, J., Kotsokalis, C., Edmonds, A., Kearney. K.: A Reference Architecture for Multi-Level SLA Management. Journal of Internet Engineering, 2010 (to appear)
3. Wolfgang Theilmann, Ulrich Winkler, Jens Happe, and Ildefons Magrans de Abril: *Managing on-demand business applications with hierarchical service level agreements.* 3rd Future Internet Symposium (FIS 2010), Berlin, September 20-22, 2010

Nature Inspired Self-Organizing Semantic Storage

Robert Tolksdorf, Hannes Mühleisen, Kia Teymourian, Marco Harasic, Anne Augustin

Freie Universität Berlin
Department of Computer Science - AG Networked Information Systems
Königin-Luise-Str. 24/26, 14195 Berlin, Germany
tolk@ag-nbi.de, {muehleis,kia,harasic,aagusti}@inf.fu-berlin.de
http://www.ag-nbi.de

Abstract. Traditional approaches for semantic storage and analysis are facing their limits on the handling of enormous data amounts of today's applications. We believe that a more radical departure from contemporary architectures of stores is necessary to satisfy that central scalability requirement. One of the most promising new schools of thought in system design are swarm intelligent and swarm-based approaches for data distribution and organization. In this paper, we describe our current work on a swarm-based storage service for Semantic Web data.

1 Introduction and Related Work

Semantic applications share the need for an efficient and scalable storage service which can handle huge amounts of semantic data. The performance and scalability level of the storage services will be defined by use case scenarios, e.g. the future Semantic Web applications need to scale to the size of the Web and the Internet network, respectively.

Conventional approaches for distributed storage services are facing complex problems in scaling and their adaptivity to changes in network infrastructure, both requirements for large-scale semantic applications. Thus new concepts and architectures for distributed storage have to be developed, and a more radical departure from contemporary architectures of storage might be the key to the realization of scalable storage systems. One of the most promising approaches for data distribution are swarm intelligent and swarm-based algorithms.

While some central storage systems support replication of their stored data, they rely on a central instance orchestrating the execution of storage and query requests. Distributed storage systems on the other hand should not rely on central nodes, as they pose single points. In contrast, distributed semantic storage systems have been proposed such as RDFPeers [1] or GridVine [3].

2 The Self-Organized Semantic Storage Service

In SwarmLinda [4], a distributed coordination system based on swarm algorithms was proposed. This coordination model makes use of a global tuple space. In SwarmLinda this tuple space is distributed to a network of nodes. The different operations are realized by using swarm algorithms to reach a high level of scalability and adaptivity to network changes which are both important properties for open distributed systems. SwarmLinda clusters tuples that match the same template and trails of virtual pheromones are left in the system to make these clusters traceable. Our SwarmLinda implementation has been extended in [5] adopting ant colony algorithms to realize a distributed storage for RDF triples. Both approaches aimed at clustering semantically related RDF triples.

Our concept is to build a Self-Organized Semantic Storage Service (S4) which uses swarm-based algorithms to store the user provided semantic data into diverse clusters potentially spanning multiple nodes. This structures can later be exploited for efficient data retrieval. We have adapted the SwarmLinda concept and its basic algorithms in order to enable it to store Semantic Web meta data in the data model RDF. Consistent with SwarmLinda, virtual ants move over a landscape consisting of a number of nodes (servers) which are interconnected.

One of the basic requirements of the ant algorithms is a similarity metric used to calculate the relative similarity between triples. Previous approaches have for example either employed string-based distance measures or more complex metrics based on ontologies. Any metric is required to yield a relative value for any pair of triples, thus making data organization and efficient clustering possible. However, our approach was deliberately designed to allow an easy replacement of the similarity metric, so this specific challenge is not detailed further.

The S4 concept based on the described algorithms offers a wide range of benefits compared to other approaches: Swarming individuals (ants) are controlled by simple algorithms, they do not require a complex rule set to perform well, additionally, all decisions can be made using only a local view. Still, individuals are able to dynamically adapt to their possibly changing environment. Network organization thus is decentralized and robust to changes in the network topology. In contrast to hash-based approaches such as DHT or B-Tree our approach does not require costly network reconstruction (achieved through communication) in the case of an error. Every node has sufficient information in the form of present scents to execute every possible query on its own, hence further eliminating single points of failure. Feasible solutions exist for issues like over-clustering, where a skewed data distribution leads unfair distribution

of data storage [2]. The same is true for hot point avoidance: if a node stores triples used in a large number of requests, it can simply move parts of them to another node or reject storage of similar triples in the future.

3 Conclusion and Outlook

We have described our motivation to extend the current state of storage systems for Semantic Web applications. The differences between centralized and distributed semantic storage services have been shown, as well as the current research in storage systems. We have outlined our design of a Self-Organized Semantic Storage Service (S4), and advantages over conventional data distribution algorithms.

While reasoning on central storage systems can be performed using logic engines, distributed reasoning is far more complex. As a general approach, achieving reasoning completeness is sacrificed in favor of scalability for distributed reasoning. Consequently, the next level for S4 is to design and implement a concept for a simple reasoning during retrieval along an is-a hierarchy and with best-effort guarantees. We then plan to add application-specific reasoning capabilities. We will continue working on the refinement and implementation of our S4 concept within our current joint research project "DigiPolis", where this system will form the basis for a indoor navigation system possibly covering entire cities. Example use cases include a semantic search for points of interest in a vicinity, in house routing with semantic restrictions, and semantic annotation of map entries.

Acknowledgments This work has been supported by the German Federal Ministry of Education and Research under grant number 03WKP07B.

References

1. Min Cai and Martin Frank. RDFPeers: A scalable distributed RDF repository, February 27 2004.
2. Matteo Casadei, Ronaldo Menezes, Robert Tolksdorf, and Mirko Viroli. On the problem of over-clustering in tuple-based coordination systems. In *SASO*, pages 303–306. IEEE Computer Society, 2007.
3. Philippe Cudré-Mauroux, Suchit Agarwal, and Karl Aberer. GridVine: An infrastructure for peer information management. *IEEE Internet Computing*, 11(5):36–44, 2007.
4. Ronaldo Menezes and Robert Tolksdorf. A new approach to scalable linda-systems based on swarms. In *Proceedings of ACM SAC 2003*, pages 375–379, 2003.
5. Robert Tolksdorf and Anne Augustin. Selforganisation in a storage for semantic information. *Journal of Software*, 4, 2009.

Behaviour of Maximum-Throughput Dynamic Routing in Loopy Networks*

Venkataramana Badarla and Douglas J. Leith

Hamilton Institute, NUI Maynooth, Ireland

Abstract. Maximizing network throughput via back-pressure routing is the subject of a considerable body of literature. However, the associated optimal throughput guarantees come at the cost of excessively high delays. Though there exists some proposals in literature for improving the delay performance, the impact of these proposals on the delay performance is very little in loopy networks (i.e., networks with routing loops). In this demonstration, with series of experiments over different topologies, we show that the basic back-pressure algorithm and two of its variants for improving delay performance induce extensive routing loops with associated high packet delay even in simple network topologies.

1 Introduction

Throughput maximization is an important issue in current Internet and it would continue to be an issue in future Internet. Using dynamic multi-path routing, the state of art back-pressure routing[1] offers considerable gains in throughput over conventional single-path routing algorithms. Indeed, it guarantees to achieve the network capacity and so its throughput performance cannot be bettered by any algorithm. While maximizing network throughput, back-pressure routing comes with no guarantee on network delay. Indeed certain features of the back-pressure routing algorithm suggest that long delays will be common. Firstly, back-pressure routing uses queue buildup at nodes to create a "gradient" within the network that guides routing. However, this may come at the cost of increased queuing delay. Secondly, back-pressure routing tends to explore all paths in a network, including paths with loops and "dead-end" paths that cannot lead to the desired destination. Hence, packets generally may not take the shortest path to their destination, thereby leading to additional delay.

The basic back-pressure algorithm is detailed in Algorithm 1. Each forwarding node in the network uses per flow FIFO queuing and we let $q_n^f(t)$ denote the number of packets queued for flow f at node n at time t. Roughly speaking,

* This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/IN.1/I901.

whenever it has a transmission opportunity each node n forwards a packet from the flow f^* to the next hop $m^*(f^*)$ that jointly maximizes the utility

$$U^f_{n,m}(t) = \left(q^f_n(t) - q^f_m(t)\right) R_{n,m}$$

where $R_{n,m}$ is the mean transmit rate of the link from node n to node m. Observe

Algorithm 1 Basic back-pressure algorithm

1: For each flow f and neighbor node m, node n computes utility $U^f_{n,m}(t) = \left(q^f_n(t) - q^f_m(t)\right) R_{n,m}$, $m^*(f) = \arg\max_m U^f_{n,m}$, $f^* = \arg\max_f \left(U^f_{n,m^*(f)}\right)$ (ties broken arbitrarily).
2: If $U^{f^*}_{n,m^*(f^*)} > 0$, then node n schedules flow f^* and forwards $\min(q^{f^*}_n(t), R_{n,m^*(f^*)})$ packets to neighbor $m^*(f^*)$.
3: Otherwise node n takes no action at time t.

that this back-pressure algorithm tends to transmit packets to the neighbor(s) with the smallest queue and highest link rate and intuitively the queue backlogs provide a "gradient" down which packets are routed. However, it commonly occurs that $q^f_n(t)$ and $q^f_m(t)$ differ by only a small amount. The routing "gradient" is then both small and rapidly fluctuating, in which case routing loops can readily be induced. Furthermore, observe that even when a neighbor has no connectivity to the destination of a flow, packets will still be forwarded to this neighbor until such time as a sufficiently large queue backlog has developed to prevent further packets stop being forwarded in that direction. However, the packets already sent in that direction will never reach the flow destination.

Back-pressure With Distance Weighting: In [2] it is noted that the utility function $U^f_{n,m}$ can by modified by summing with a finite constant without affecting the stability properties of the basic back-pressure routing, and in particular, that selecting a constant based on the shortest-path distances from nodes n and m to the destination of flow f is a possible choice. We therefore consider a modified utility function of the form

$$U^f_{n,m}(t) = \left(q^f_n(t) - q^f_m(t)\right) R_{n,m} + \frac{M}{D^f_m}$$

where the intuition is that since the utility is inversely proportional to the distance D^f_m, shortest paths are favored when making forwarding decisions. The design parameter M allows the relative weight of the distance term to be adjusted. We refer to use of this utility function as *distance aided routing*.

2 Implementation, Demo Setup and Snapshot of Results

We implemented the back-pressure algorithms as a *DynamicRouter* element within the Click router framework in Linux. The Click router framework provides a modular architecture that lies within the Linux queuing layer (*i.e*, between the

network layer and the device driver layer) and interacts with the network layer and the interface device driver via the Click elements ToHost/FromHost and ToDevice/FromDevice, respectively.

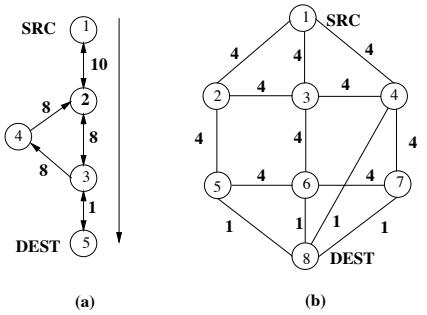


Fig. 1. Network Topologies.

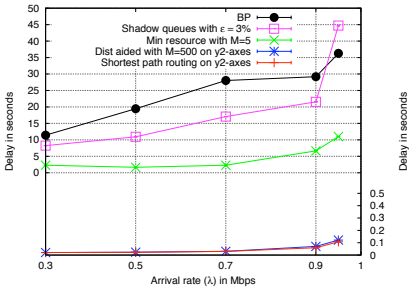


Fig. 2. Delay performance over the topology in Fig. 1(a).

Table 1. Demonstration setup for the Demo Session

Item	Description
Soekris computers (quantity 10)	Single-board computers with 433MHz CPU, 256MB RAM and four 100Mbps ethernet ports
N/w topologies	Shown in Figure 1
Performance metrics	a) packet delay b) throughput c) Buffer size at intermediate nodes d) reassembly buffer at receiver
Traffic type	a) TCP using iperf and scp b) UDP using mgen

Details of the demonstration setup, such as type of computers will be used, network topologies will be considered etc are given in Table 1. Fig 2 presents snapshot of the experimental results conducted over the topology in Fig 1(a). In Fig. 2(a), which shows measured delay performance, we can observe that the delay for back-pressure algorithm and its variants (shadow-queue and min-resource algorithms [3]) is excessively high (ranges from 3s to 45s) at all offered loads. Also can be observed that the proposed distance aided approach shows the lowest delays which are at par with the shortest path routing. More detailed results over the second topology will be shown during the live demonstration session at the conference venue.

References

1. A. L. Stolyar, "Maximizing Queueing Network Utility subject to Stability: Greedy-Primal Dual Algorithm," *Queueing Systems*, vol. 50, no.4, pp. 401-457, 2005.
2. L. Georgiadis et al, *Resource Allocation and Cross Layer Control in Wireless Networks* 2006.
3. L. Bui et al, "Novel Architectures and Algorithms for Delay Reduction in Back-pressure Scheduling and Routing", *in Proc. IEEE Infocom Mini-Conference* 2009.

Poster: Persistent Content-based Publish/Subscribe Service On Top Of DHT

Yan Shvartzshnaider[1], Maximilian Ott[2], and David Levy[1]

[1] School of Electrical and Information Engineering
The University of Sydney, Australia
yshv6985@uni.sydney.edu.au
david.levy@sydney.edu.au
[2] National ICT Australia (NICTA)
max.ott@nicta.com.au

Approach

We propose design for a distributed and persistent content-based publish/subscribe service based on the Rete algorithm. Rete [1] has been designed to support pattern matching in production rule systems. In particular, Rete is able to manage, interpret and evaluate a set of rules against a large persistent dataset. The matching process relies on a loosely coupled dataflow matching network which makes the Rete algorithm a suitable candidate for implementation in distributed settings, such as with DHTs. In our approach we make the following key contributions to the previous work: efficient information filtering, persistence of subscriptions and publications, scalability, query expressiveness.

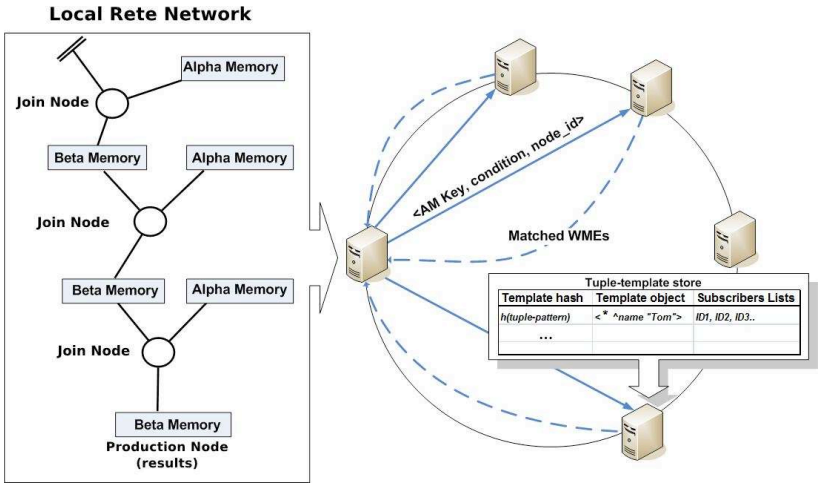


Fig. 1. An extremely simplified illustration of our implementation approach.

Rete generates a dataflow network from a given set of rules. This dataflow network consists of loosely coupled alpha memory (AM), beta memory (BM)

and join nodes. The alpha memories act as predicates on the dataset. With each new matched tuple *right activation* is triggered. The beta memories store the intermediate results from previous join node (note, these results can be shared by different subscriptions at the node). The join nodes perform variable binding tests between sets of tuples in the BM and data tuples in the AM, and outputs binds as variable tuples to the next BM. This process is called *left activation*. In other words, Rete's matching process is a chain of successively triggered right and left activations with the final result stored in the last node in the chained beta-memory, also called *production node*. The activations allow Rete dynamically to adjust its results to any incremental changes in the data.

Our approach. Similarly to Rete, our system treats tuples as primitives. Published events are tuples and subscription is essentially a set of bindings by variables rule tuples. Intuitively, we propose to follow the rendezvous model to distribute Rete's alpha-memories over the DHT nodes to arrange a meeting between each AM and matching to it published tuples. Each node in the network has the ability to subscribe to and publish content. All subscriptions at the subscribing node are managed locally by a single Rete network. As Figure 1 shows, our DHT's nodes store the subscription's rules and forward every matched tuple to appropriate subscriber nodes where on arrival the tuples are distributed among the alpha memories, triggering the matching process.

Contributions and Future work

We have presented a design for a distributed and persistent content-based publish/subscribe system on top of DHT. Our main contributions compared to other related work are: (1) *Persistence* – our system stores both subscriptions and published events. Hence a query can also return previously published, matching events. (2) *Subscription expressiveness* – subscriptions are defined as a set of conditions (rules) bound by variables. (3) Efficient information filtering due to the efficiency in pattern matching of the Rete algorithm. In our future work we plan to improve the current design to mitigate any potential load balancing problems due to non-uniform distribution of published events' attributes. We will analyse system's performance, and in particular, will compare the current push-based design against alternative pull- based approaches. Finally, our ultimate goal is to create a persistent and content-based pub/sub facility that can be used as a building block by distributed Internet application and services.

References

[1] Forgy, C.: Rete: A fast algorithm for the many patterns/many objects match problem. Artificial Intelligence **19**(1) (1982) 17–37

Access Control for RDF: Experimental Results

Giorgos Flouris[1], Irini Fundulaki[1], Maria Michou[1], and Grigoris Antoniou[1,2]

[1] Institute of Computer Science, FORTH, Greece
[2] Computer Science Department, University of Crete, Greece
{fgeo, fundul, michou, antoniou}@ics.forth.gr

Abstract. One of the current barriers towards realizing the huge potential of Future Internet is the protection of sensitive information, i.e., the ability to selectively expose (or hide) information to (from) users depending on their access privileges. In this work we discuss the experiments conducted with our repository independent, portable across platforms system that supports fine-grained enforcement of RDF access control.

1 Introduction

The potential of RDF as a data representation standard for the Future Internet is undermined by the lack of an effective mechanism for controlling access to RDF data. In light of the potentially sensitive nature of RDF information, the issue of *securing* RDF content and *ensuring the selective exposure of information* to different classes of users depending on their access privileges is an important issue. The building blocks of an access control system are the *specification language*, that allows the expression of access control permissions and policies, and the *enforcement mechanism*, responsible for applying the latter to the data, by denying access to data that the policy has deemed as non-accessible.

In this work, we enforce access control by proposing a solution which is repository independent, portable across platforms, and in which *fine-grained access control* (protection at the level of RDF triple) is enforced by a component built on top of the RDF repository. In this poster we report on experiments performed with our system; the full description and formal semantics of our language, as well as more details on the approach can be found in [6].

2 RDF Access Control Framework

We concentrate on *fine-grained* RDF access control for *read-only queries*. An *access control permission* is used to explicitly grant or deny to/from a given user the ability to access an RDF triple, or a set of RDF triples, and can be viewed as a query whose evaluation over an RDF graph results in a set of triples which are accordingly granted or denied access. Access control permissions are expressed using SPARQL [7] *triple patterns* and *value constraints*, and are of the form: $\mathcal{R} = \mathsf{include/exclude}\ (x,\ p,\ y)\ \mathsf{where}\ \mathcal{TP}, \mathcal{C}$ with $(x,\ p,\ y)$ a triple pattern, $\mathcal{TP}$ a conjunction of triple patterns and $\mathcal{C}$ a conjunction of value constraints on the variables appearing in the triple patterns. Explicit access rights are not set for all triples in an RDF graph, and permissions are not always unambiguous (i.e., a triple could be marked as both accessible and inaccessible). To determine whether such triples should be accessible, we use the notion of *access control policy*, which includes a set of positive and a set of negative permissions, as well

as two boolean flags (*default semantics* and *conflict resolution* – ds, cs resp.), which determine whether triples with missing (resp. ambiguous) permissions are accessible or not. A full description of the formal semantics of access control permissions and policies can be found in [6].

3 Implementation and Experiments

Architecture: We implemented a main memory platform which serves as an additional access control layer on top of an arbitrary RDF repository. Our goal was for our system to be portable across platforms, so it was designed in a repository-independent way. The system's architecture is shown in Fig. 1. It is comprised of the following modules, all implemented in Java: the RDF Dataset Loader, responsible for loading the complete RDF dataset in the underlying repositories, the RDF Access Control Policy Manager that loads in memory the access control policies and the RDF Access Control Enforcement Module, which translates the access control policies into the appropriate programs that compute the accessible triples of an RDF dataset and annotates accordingly the data in the repositories with accessibility information.

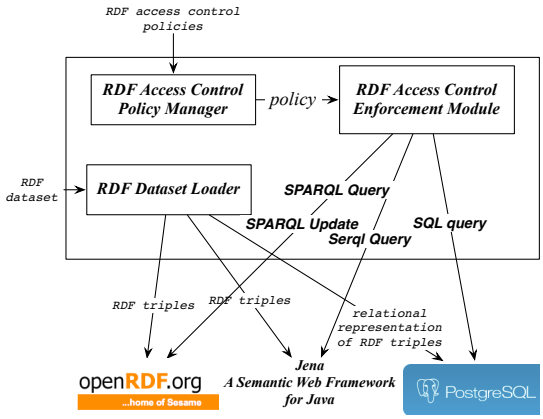
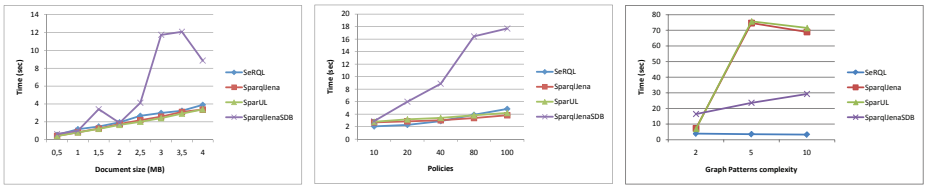


Fig. 1. System Architecture

To enforce an access control policy we produce a query which implements the semantics of the policy and is expressed in the language supported by the underlying repository. The triples in the result of the evaluation of this query on the RDF graph are exactly the accessible triples which are then *annotated* as such. Conceptually, annotations can be represented by adding a fourth column to an RDF triple (hence obtaining a quadruple), denoting whether the triple is accessible or not; if several different user roles need to be supported, one column per role should be added. Annotations can be stored using the *named graphs* mechanism of RDF repositories [5], or, in the case of a relational backend, by extending the triple table that stores the RDF triples with a fourth column.

Experiments: Our experiments measured the time required to annotate the set of RDF triples, using the above methodology, in state-of-the-art RDF repositories (Sesame [3], Jena [1]) or relational backends (Postgres [2]). We used the SP2Bench [8] data generator to obtain the input RDF graphs. We implemented our approach on top of Jena v2.6.2, Sesame v2.3.1 and Postgresql v8.4. For Jena we tested the SparqlJenaModule and SparqlJenaSDBModule (processing SPARQL queries) as well as the SPARULModule (processing SPARQL Update

queries) modules. SparqlJenaModule and SPARULModule load the datasets into main memory whereas the SparqlJenaSDBModule stores the datasets to a Postgresql database. For Sesame we used the SeRQLModule, which processes SPARQL [7] and SeRQL [4] queries in memory.



(a) $gp = 2$,$ps = 80$ (b) $gp = 2$,$doc = 4MB$ (c) $ps = 80$, $doc = 4MB$

Fig. 2. Experiments

We measured the time required for the annotation as a function of four different parameters: *(i) document size* (*doc*), i.e., the size of the input RDF graph (size ranging between 500KB-4MB with a 500KB increase); *(ii) policy size* (*ps*), i.e., the number of permissions in the access control policy (for sizes of 10, 20, 40, 80 and 100, with an equal share of positive/negative permissions in each case); *(iii) permission size* (*gp*), i.e., the number of triple patterns and constraints in the where clause of each access control permission (values considered: 2, 5, 10); and *(iv) policy parameters*, i.e., the values of the ds, cr parameters of the input policy (all 4 combinations considered).

Evaluation: Fig. 2 shows a subset of the results of our experiments: we run each experiment 5 times, and took the average time. In each graph, the annotation time is presented as a function of one of the parameters *(i)-(iv)*, for fixed values for the other parameters. We report here on policies with *"deny"* as default semantics (*ds*) and conflict resolution policy (*cr*) because it is the most common one. The results show that our approach scales along the considered parameters. All the platforms that we ran our experiments on demonstrated a linear behavior as document, policy sizes and permission complexity increased (except the Jena SPARUL and SPARQL Modules).

References

1. Jena A Semantic Web Framework for Java. `http://jena.sourceforge.net/`.
2. PostgreSQL. http://www.postgresql.org/.
3. Sesame: RDF Schema Querying and Storage. http://www.openrdf.org/.
4. J. Broekstra and A. Kampman. SeRQL: A Second Generation RDF Query Language. In *Workshop on Semantic Web Storage and Retrieval*, 2003.
5. J. J. Carroll, C. Bizer, P. J. Hayes, and P. Stickler. Named Graphs. *JWS*, 3(4), 2005.
6. G. Flouris, I. Fundulaki, M. Michou, and G. Antoniou. Controlling Access to RDF Graphs. In *FIS*, 2010.
7. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. `www.w3.org/TR/rdf-sparql-query`, January 2008.
8. M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel. SP2Bench: A SPARQL Performance Benchmark. Technical report, arXiv:0806.4627v1 cs.DB, 2008.

Experimental Testing in the Future Internet PERIMETER Project

Eileen Dillon[1], Gemma Power[1] and Frances Cleary Grant[1]

[1]Telecommunications Software and Systems Group, Waterford Institute of Technology, Cork Road, Ireland
[1]{edillon, gpower, fcleary}@tssg.org

Abstract. This submission will provide interested parties with the opportunity to learn more about the FP7 PERIMETER project. Details of the testing and experimental methodologies and the role of testbed activities involved in this Quality of Experience network mobility project will be given. A demonstration and explanation of the Quality of Experience management system of PERIMETER will be provided and will allow visitors to interact with the system using a Google Android device with the PERIMETER middleware installed. This will be illustrated with the aid of the purpose built PERIMETER Visualisation Tool.

Keywords: Future Internet, Testbeds, Experimental Testing and Methodologies, PERIMETER, Demonstration.

1 Introduction

This submission accompanies the PERIMETER FIS paper [1] which examines the requirements and challenges of the Future Internet and demonstrates how they are dealt with in the Telecommunications Software & Systems Group research centre using a case study of the Future Internet project; PERIMETER [2]. This submission will provide a demonstration, poster and explanation of the PERIMETER middleware. Here, an overview of the PERIMETER project is provided before a description of the demonstration is given.

2 PERIMETER Overview

The PERIMETER project [2] entitled "User-Centric paradigm for Seamless Mobility in Future Internet" is a three-year project funded by the FP7 programme. PERIMETER is listed as one of eight 'Experimentally driven advanced research" projects in the first wave of the FIRE Initiative [3].

PERIMETER's main objective is to establish a new paradigm of user centricity for advanced networking. Putting the user in the centre rather than the operator enables the user to control their identity and preferences (e.g. cost, security, network

selection). This enables the mobile user to be "Always Best Connected" (ABC) in multiple-access multiple-operator networks of the Future Internet.

To achieve this, PERIMETER develops and implements protocols and a middleware to address requirements for privacy, security, resilience and transparency. The network selection is done based upon a parameter called Quality of Experience (QoE) [4]. Analysis of QoE parameters allows the evaluation of network connections by user perceivable indexes and parameters (i.e. cost, security, connection performance). Network selection can be handled automatically by PERIMETER's QoE management system for generic QoE definition, QoE signalling, and QoE based content adaptation [5], thus creating a single-sign-in experience.

The architecture of PERIMETER consists of Terminal Nodes and Support Nodes. The PERIMETER Terminal Nodes are users' handheld devices which have some storage and computational restrictions. Currently mobile phones and netbooks based upon Google's Android platform [6] are supported. The Support Nodes are physical devices that can act as data sources and gateways and are connected using a peer-to-peer approach both to other PERIMETER Support Nodes and PERIMETER Terminal Nodes. The Support Nodes do not have any storage or computational restrictions and are therefore used as the main storage for the system and for expensive computational activities. The Support Nodes reside in PERIMETER's state of the art federated QoE testbed between Waterford Institute of Technology, Ireland and Technical University of Berlin, Germany while the Terminal Nodes are connected wirelessly to the federated testbed infrastructure [7]. The PERIMETER system is supported by a number of identified applications which also run across the federated testbed and support the verification of the user centric scenario based approach adopted in the PERIMETER project.

3 PERIMETER Demonstration

The PERIMETER demonstration will encompass the QoE management system aspect of the PERIMETER middleware. This will be showcased using a purpose built PERIMETER Visualisation Tool (PVT). The PVT will demonstrate the internal functionality of the PERIMETER middleware from the users' perspective in a graphical manner, making it intuitive for visitors to the stand. Visitors will be given the opportunity to interact with PERIMETER using Google Android handheld devices with the PERIMETER middleware pre-installed, as depicted in Figure 1. The demonstration will be supported by information on the system and an accompanying poster.

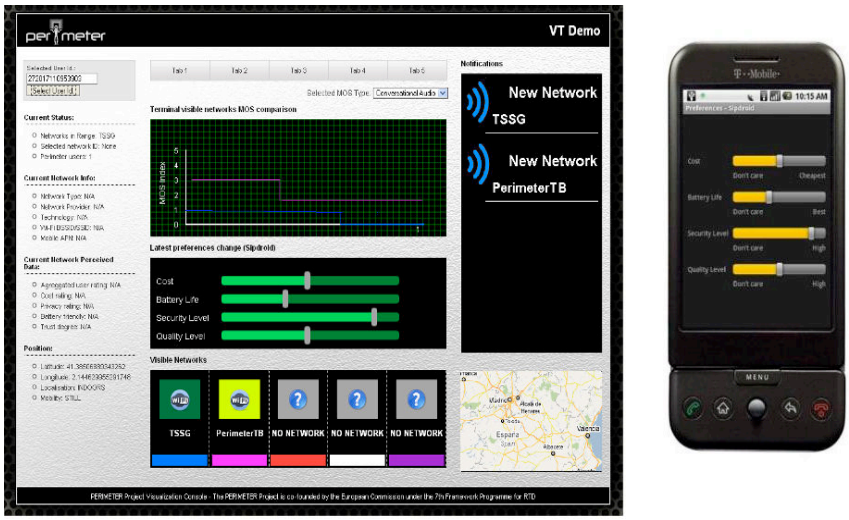


Figure 1 PERIMETER Visualisation Tool and on the Android G1 Device

4 Summary

This submission provides the opportunity for interested parties to learn more about the PERIMETER system, to see a demonstration of the QoE management system aspect of the PERIMETER middleware, and to provide their feedback and comments.

Acknowledgments. This research activity is funded under the EU ICT FP7 project, PERIMETER (Project No.: 224024).

References

1. E. Dillon, G. Power and F. Cleary Grant. Experimental Testing in the Future Internet PERIMETER Project. FIS 2010 [accepted].
2. PERIMETER, User-centric Paradigm for Seamless Mobility in Future Internet, Available: http://www.ict-perimeter.eu/
3. The Future Internet Research and Experimentation (FIRE) Initiative, Available: http://cordis.europa.eu/fp7/ict/fire/
4. DSL Forum Technical Report TR-126, Triple-play Services Quality of Experience (QoE) Requirements, December 2006, Produced by Architecture & Transport Working Group.
5. E. Dillon, G. Power, M. O. Ramos, M. A. Callejo Rodríguez, J. R. Argente, M. Fiedler, D. S. Tonesi, PERIMETER: A Quality of Experience Framework, Future Internet Symposium (FIS) 2009, Berlin (Germany), Sep. 1-3 2009.
6. Google Android Operating System, Available http://www.android.com/
7. F. Cleary Grant, E. Dillon, G. Power, T. Kaschwig, A. Cihat Toker, and C. Hammerle, QoE Testbed Infrastructure and Services: Enriching the End User's Experience, TridentCom, May 2010.

NoTube: Making Television More Personal

The NoTube project consortium[1]

Abstract. The NoTube European project looks at creating the future of TV, in which the TV user will be placed back in the driver's seat by having a personalized TV experience with rich interaction possibilities. For this, new technologies like Linked Data, Semantic Web and Social Web data analysis are applied to the TV domain.

The ultimate goal of the EU project NoTube is to develop an adaptive end-to-end architecture, based on semantic technologies, for personalized creation, distribution and consumption of TV content. The project takes a user-centric approach to investigate fundamental aspects of consumers' content-customization needs, interaction requirements and entertainment wishes, which will shape the future of the "TV" in all its new forms. To achieve all that NoTube is working on the following challenges:

- *Integration of TV & Web with help of semantics:* Currently there are many practical software and hardware hurdles for users to handle such integration. NoTube is using semantics (e.g. linked open data) to open and interlink TV content in a Web fashion.
- *Putting the user back in the driving seat*: Personalized services are now common, but the user data is still under the control of separate applications, and the users are faced with multitude of distributed personal data, hidden in tons of inaccessible cookies
- *TV is not bound to one device:* We currently witness multiple device usage scenarios, e.g. use of computer as TV & vice versa and use of mobile device as remote control. In this context, NoTube architecture aims to support device independence.

The main challenges in the television domain are the scale of the content available and the need for filtering and personalization of the content. These challenges are explored from different content and user perspectives within three representative scenarios for future television enabled by semantic technology:

[1] http://www.notube.tv/project/partners

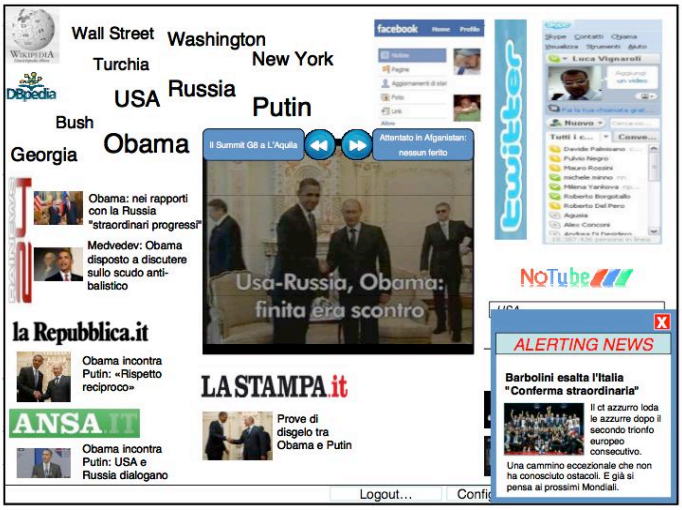


Figure 1. RAI demonstrator for personalized news

- *Personalised News [1].* This demonstrator (Figure 1), using archival content from RAI, shows how news programs can be enriched with concepts (people, places, themes) that allow easy browsing to additional information. The screenshot shows (above) an alert, which allows the user to be informed when a important news story enters in his home ambient.

Figure 2. Stoneroos demonstrator of a personalized TV guide.

- *iFanzy[2] Personalised TV Guide.* The iFanzy demonstrator (Figure 2), developed by Stoneroos, enables the user to build in a simple fashion a profile which can be used to generate different types of recommendations for personal EPG. This screen shot (above) shows iFanzy in action with Dutch programs and Dutch EPG data.

[2] http://www.ifanzy.nl

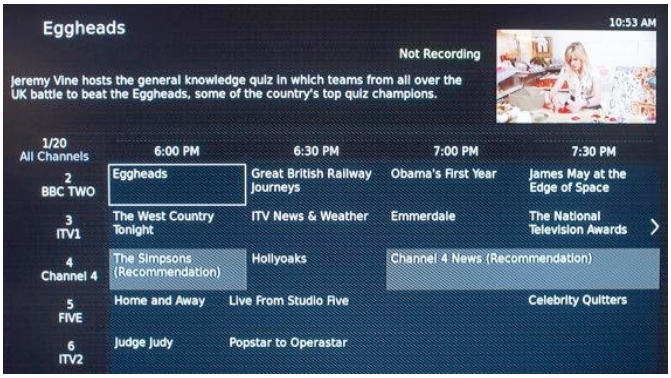


Figure 3. BBC demonstrator of TV programme recommendations

- *TV and the Social Web [2].* This demonstrator (Figure 3), driven by BBC use cases and content, shows how TV watching can be personalised using Social Web data and facilitating a personalised TV experience without an intrusive user profiling process. It illustrates how TV can be linked to your own or you friend's social-web data, such as bookmarks and Facebook profiles. The screen shot shows a recommendation based on such data using a MythTV[3] front end.

To enable this vision, sets of services are developed for users, metadata and TV content, which are described semantically and mediated by a broker. Specific applications are also developed to make use of those services to provide the desired functionalities, e.g. user activity capture, content recommendation. This open TV and Web infrastructure is illustrated in Figure 4. Major features in the NoTube service architecture are:

- User, Metadata and Content oriented services: the complete content selection, adaptation and delivery process from content provider to home ambient is supported.
- Service brokering: Automated configuration of services based on explicit semantic descriptions of the services, which are gathered in the repository.
- Multiple devices: Functionality of NoTube should not be limited to single devices. The demonstrators run on a TV with a setup box, on computers and on mobile devices, as well as one combinations of these (e.g. mobile device as remote control showing recommendations).

[3] http://www.mythtv.org/

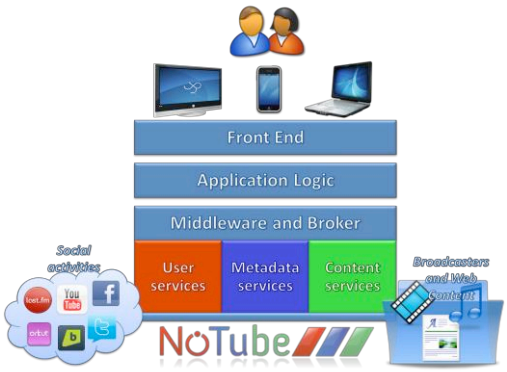


Figure 4. NoTube high level architecture.

For more information about the future of television, please refer to the project Website http://www.notube.tv and follow the project blog http://blog.notu.be

References

[1] "Personalised Semantic News: combining Semantics and Television", Robert Borgotallo, Roberto del Pero, Alberto Messina, Fulvio Negro, Luca Vignaroli, Lora Aroyo, Chris van Aart and Alex Conconi at the 1st International ICST Conference on User-centric Media - UCMedia 2009. Dec 2009.
[2] "Linking TV And Web Using Semantics, A NoTube Application", Balthasar Schopman, Davide Palmisano, Ronald Siebes, Chris van Aart, Véronique Malaise, Michele Minno and Lora Aroyo. In 1st NoTube workshop on Future Television, in the Adjunct Proc. EuroITV 2010. Jun 2010.

The Information Workbench
Interacting with the Web of Data

Peter Haase[1], Andreas Eberhart[1], Sebastian Godelet[1], Tobias Mathäß[1],
Thanh Tran[2], Günter Ladwig[2], Andreas Wagner[2]

[1]fluid Operations GmbH, Walldorf, Germany
{firstname.lastname}@fluidops.com
[2]Institute AIFB, University of Karlsruhe, Germany
{gla,dtr,awa}@aifb.uni-karlsruhe.de

Abstract. We present the Information Workbench, an application for interacting with the Web of data. The Information Workbench manages large amounts of structured and unstructured information, which may be imported and integrated from existing sources, but also allows end users to annotate, complete and update information in a collaborative way. New paradigms for accessing information include hybrid search across the structured and unstructured data, keyword search combined with facetted search, as well as semantic query completion and interpretation, which assists the user in expressing complex information needs by an automated translation of keyword queries into hybrid queries. A *Living UI* based on widgets for the interaction with the data enables a homogeneous, seamless, continuous and personal experience.

1 Main Features

In recent years, we have observed a tremendous success of the paradigms of the Web 2.0 in Web applications. The Web has developed from a platform in which information is published by few providers to an interactive and collaborative medium for producing, consuming and sharing information. As the most prominent example, Wikipedia has grown to one of the central knowledge sources.

At the same time, the amount of structured data available on the (Semantic) Web has been increasing rapidly. Currently, there are billions of triples published and connected in web data sources of different domains. The benefits of this linked data are obvious (at least to our community), but there are still very few applications that actually make use of it. In particular, the potential of applications complementing the Web of data with the characteristics of the Web 2.0 has been largely unrealized: Existing applications are mainly limited to generic linked data browsers, they typically assume that the data is published by data providers – and thus read only – as apposed to user generated content. In other words, the means for the interaction with the Web of data are still in its 1.0 state.

The Information Workbench provides the means to fill this gap – as an infrastructure for building applications for the interaction with the Web of data, combining Web 2.0 features such as collaboration with those of semantic technologies. The key features of the Information Workbench include:

– the ability to manage large amounts of structured and unstructured content, which may be imported and integrated from existing sources, but also may be generated by end users, who can annotate, complete and update the content,
– new paradigms for accessing information, including hybrid search across the structured and unstructured data, keyword search combined with facetted search, and semantic query completion and interpretation, automatically translating keywords into hybrid queries corresponding to the user intent,
– a *Living UI* to enable a homogeneous, seamless and personal experience, despite heterogeneous and dynamic data. Knowing what, when, and where to show is realized through an automated and customizable selection of widgets, which implement various paradigms for interacting with the data.

The technology of the Information Workbench is generic in the sense that it is independent of particular domain or data set, or application. In fact, the strength lies in the ease of building concrete applications. To demonstrate this, we have setup an instance of the Information Workbench to interact with a Semantic Wikipedia, publicly accessible at `http://iwb.fluidops.com/`. To bootstrap the system, we have taken the English Wikipedia and enriched it with structured data from the Open Linked Data Initiative, including the DBpedia data set.

While the data in the demonstrator spans many domains (in fact it covers a large fraction of the world knowledge) and thus potential applications are just as manifold, we further illustrate the benefits in a small application scenario.

2 An Application Scenario

Sebastian is a hobby astronomer, who is familiar with using computers, experienced in using Web 2.0 style wikis and forum software for managing pictures and observation reports. Especially in the domain of astronomy, information is abundant. For example, Wikipedia contains vast amounts of knowledge about astronomy. Most of this knowledge is available in unstructured form, but increasingly, structured data is published. As one example, DBpedia already contains some structured knowledge about astronomic entities extracted from Wikipedia. Using the Information Workbench, Sebastian is able to access and interact with the data aggregated from the various available sources. The data is presented in a resource-centric way, with a single page per resource. One such page may be that of the solar system (c.f. Figure 1). For displaying the Information, the application automatically selects appropriate widgets based on the data available. For example, cosmic objects might be associated with coordinates. Based on them, these objects are displayed using Google Sky. At the same time, Sebastian would like to personalize the interface to his preferences: Sebastian may want to have a Twitter feed included that displays live news about a particular resource, while some other user may prefer to see videos associated with that resource.

An important means of interacting for Sebastian is the ability to enrich the existing data, in the form of photographs, text, and also, more structured data. In a simple case, he may want to annotate the unstructured information within the Wikipedia. Simple annotations such as making the relationship between two entities explicit (e.g. stating `[[location::Solar System]]` for a planet)

Fig. 1. Screenshot of the Demonstrator

lead to immediate benefits: Promptly, Sebastian will be able to perform an ad-hoc structured query, e.g. asking for the mass and planets in the solar system arranged in the order of the distance from the sun. The Information Workbench will assist in formulating complex information needs by automatically translating the query from keywords into structured queries and making suggestions for completing and refining the query. Results are displayed using an appropriate visualization, e.g. in the form a bar diagram type.

Figure 1 shows a screenshot the Information Workbench in our scenario. The screen shows the page of the solar system, integrating information from the original Wikipedia page, the DBpedia data set, various external sources and the annotations added by Sebastian. The upper left widget shows a wiki-based interface to the resource, the widget below shows the associated structured as a graph. On the right side, we see external content in the form of video from YouTube and a Twitter news feed. The upper right widget shows the results of a structured query associated with the page, displaying the density of the planets in the solar system.

For more details on the search concepts implemented by the Information Workbench, we refer the reader to [1].

References

1. Thanh Tran, Peter Haase, and Rudi Studer. Semantic search - using graph-structured semantic models for supporting the search process. In *ICCS*, pages 48–65, 2009.