

RDF-based Peer-to-Peer-Networks for Distributed (Learning) Repositories

Wolfgang Nejdl¹, Mario Schlosser¹, Wolf Siberski¹, Martin Wolpers¹, Bernd Simon², Stefan Decker³, Michael Sintek⁴

¹ Learning Lab Lower Saxony, University of Hannover, Germany,
e-mail: {nejdl, schlosser, siberski, wolpers}@learninglab.de

² Department of Information Systems, Vienna University of Economy, Austria,
e-mail: bernd.simon@wu-wien.ac.at

³ Information Sciences Institute/USC, Marina del Rey, USA, e-mail: stefan@isi.edu

⁴ German Research Institute for Artificial Intelligence,
Kaiserslautern, Germany, e-mail: sintek@dfki.de

December 15, 2002

Abstract Metadata for the World Wide Web are important, but metadata for Peer-to-Peer (P2P) networks are absolutely crucial. In this paper we discuss the open source project Edutella, which combines semantic web and peer-to-peer technologies in order to make distributed learning repositories possible and useful. We describe the main services of the Edutella network infrastructure and its architecture based on the exchange of RDF metadata, and specify the query service (based on the Edutella Common Data Model and Query Exchange Language RDF-QEL) as one of the core services of Edutella. As a schema-based peer-to-peer network, Edutella extends conventional peer-to-peer networks by allowing different and extensible schemas to describe peer content, a necessary feature for information rich peer-to-peer networks. We describe and discuss the HyperCuP topology, which implements an optimal broadcast topology, and show how it can be used to form the backbone of a super-peer-based topology suitable for schema-based peer-to-peer networks, including appropriate routing indices for super-peer/super-peer connections as well as for super-peer/peer connections. We also discuss an extension of this architecture providing additional capabilities for inferences and model transformation based on the TRIPLE language. Finally, we discuss the use of the Edutella infrastructure in the EU/IST ELENA project which aims to create smart spaces for learning.

Key words Semantic Web, Peer-to-Peer, Schema-Based Routing, Distributed RDF Repositories, Learning Services

1 Introduction

While in the client/server-based environment of the World Wide Web metadata are useful and important, for *Peer-to-Peer (P2P)* environments metadata are absolutely crucial. Information Resources in P2P networks are no longer organized in hypertext like structures, which can be navigated, but are stored on numerous peers waiting to be queried for these resources if we know what we want to retrieve and which peer is able to provide that information. Querying peers requires metadata describing the resources managed by these peers, which is easy to provide for specialized cases, but non-trivial for general applications.

P2P applications have been successful for special cases like exchanging music files. However, retrieving a song like “Material Girl from Madonna” does not need complex query languages nor complex metadata, so special purpose formats for these P2P applications have been sufficient. In other scenarios, like exchanging educational resources, queries are more complex, and have to build upon standards like IEEE/IMS LOM [15, 16] metadata with up to 100 metadata entries, which might even be complemented by domain specific extensions. Furthermore, by concentrating on domain specific formats, current P2P implementations appear to be fragmenting into niche markets instead of developing unifying mechanisms for future P2P applications. There is indeed a great danger (as already discussed in [10]), that unifying interfaces and protocols introduced by the World Wide Web get lost in the forthcoming P2P arena.

Edutella Infrastructure The Edutella project [11, 24, 25] addresses these shortcomings of current P2P applications by building on the W3C metadata standard RDF [21, 5]. The project is a multi-staged effort to scope, specify, architect and implement an RDF-based metadata infrastructure for P2P-networks based on the recently announced JXTA framework [12]. The initial Edutella services will be *Query Service* (standardized query and retrieval of RDF metadata), *Replication Service* (providing data persistence / availability and workload balancing while maintaining data integrity and consistency), *Mapping Service* (translating between different metadata vocabularies to enable interoperability between different peers), *Mediation Service* (define views that join data from different metadata sources and reconcile conflicting and overlapping information) and *Annotation Service* (annotate materials stored anywhere within the Edutella Network).

The Edutella infrastructure aims to provide the metadata services needed to enable interoperability between metadata of heterogeneous applications. The main application area we discuss within this paper are P2P networks for the exchange of educational resources (using schemas like IEEE/IMS LOM, and ADL SCORM [2] to describe course materials). However, as this infrastructure is independent of specific schemas, it is suitable for the connection of distributed RDF knowledge bases in other application areas, too.

As the query service is the fundamental service in Edutella, upon which other services are built, we specify in Section 2 the Edutella Common Data Model (ECDM) as basis for the Edutella query exchange language and format (see [24, 25]). In Section 3 we describe a new P2P routing topology called HyperCuP. The

HyperCuP topology minimizes broadcast traffic and distance between the peers in such a network, allowing it to scale to large numbers of peers. Section 4 shows how this topology can be used as a super-peer backbone topology for the Edutella network, enhanced with appropriate super-peer/super-peer and super-peer/peer routing indices, taking important schema characteristics of the connected peers into account to optimize query distribution.

Section 5 discusses how the basic Edutella data model can be extended with functional terms, model identifiers and RDF-appropriate syntax to implement enhanced model transformation and inference capabilities. Section 6 describes the use of the Edutella infrastructure in the EU/IST project ELENA which aims to create smart spaces for learning, building upon the Edutella infrastructure.

2 Edutella Query Service

The Edutella Query Service is a standardized query exchange mechanism for RDF metadata stored in distributed RDF repositories and serves both as query interface for individual RDF repositories located at single Edutella peers as well as query interface for distributed queries spanning multiple RDF repositories (storing RDF statements based on arbitrary RDFS schemata). One of the main purposes is to abstract from various possible RDF storage layer query languages (e.g., SQL) and from different user level query languages (e.g., RQL, TRIPLE): The Edutella Query Exchange Language and the Edutella Common Data Model provide the syntax and semantics for an overall standard query interface across heterogeneous peer repositories for any kind of RDF metadata. The Edutella network uses the query exchange language RDF-QEL (based on Datalog semantics) as standardized query exchange language format which is transmitted in an RDF/XML-format.

We will start with a simple RDF knowledge base and a simple query with the following RDF XML Serialization:

```
<lib:Book about="http://www.xyz.com/sw.html">
  <dc:title>Software Engineering</dc:title>
</lib:Book>

<lib:Book about="http://www.xyz.com/ai.html">
  <dc:title>Artificial Intelligence</dc:title>
</lib:Book>

<lib:AI-Book about="http://www.xyz.com/pl.html">
  <dc:title>Prolog</dc:title>
</lib:AI-Book>
```

Evaluating the following query (plain English)

Return all resources that are a book having the title 'Artificial Intelligence' or that are an AI book.

we get the query results shown in Figure 1, depicted as RDF-graph.

Edutella peers can be highly heterogeneous in terms of the functionality (i.e., services) they offer. A simple peer has RDF storage capability only. The peer has

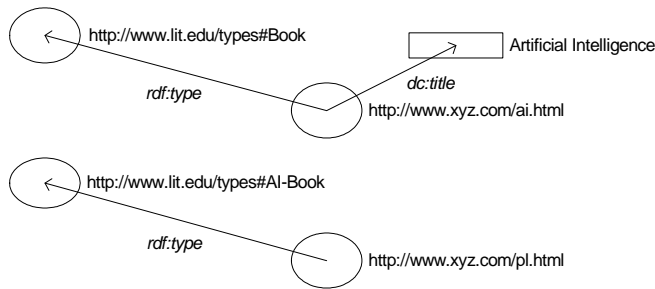


Fig. 1 Query Results as RDF Graph

some kind of local storage for RDF triples (e.g., a relational database) as well as some kind of local query language (e.g., SQL). In addition the peer might offer more complex services such as annotation, mediation or mapping. To enable the peer to participate in the Edutella network, Edutella wrappers are used to translate queries and results from the Edutella query and result exchange format to the local format of the peer and vice versa, and to connect the peer to the Edutella network by a JXTA-based P2P library. To handle queries, the wrapper uses the common Edutella query exchange format and data model for query and result representation. For communication with the Edutella network the wrapper translates the local data model into the Edutella Common Data Model ECDM and vice versa, and connects to the Edutella Network using the JXTA P2P primitives, transmitting the queries based on ECDM in RDF/XML form.

In order to handle different query capabilities, Edutella defines several capability levels, describing which kind of queries a peer can handle (conjunctive queries, relational algebra, transitive closure, etc.) The same internal data model is used for all levels.

Edutella Common Data Model (ECDM) The ECDM is based on Datalog, which is a well-known non-procedural query language based on Horn clauses without function symbols. A Datalog program can be expressed as a set of rules/implications (where each rule consists of one positive literal in the consequent of the rule (the head), and one or more negative literals in the antecedent of the rule (the body)), a set of facts (single positive literals) and the actual query literals (a rule without head, i.e., one or more negative literals). Literals are predicates expressions describing relations between any combination of variables and constants such as `title(http://www.xyz.com/book.html, 'Artificial Intelligence')`. Disjunction is expressed as a set of rules with identical head. Additionally, we can use negation as failure in the antecedent of a rule, with the semantics that such a literal cannot be proved from the knowledge base. A Datalog query then is a conjunction of query literals plus a possibly empty set of rules [31]. Datalog queries easily map to relations and relational query languages like relational algebra or SQL. In terms of relational algebra Datalog is capable of expressing selection, union, join and projection and hence is a relationally complete query language. Additional features include transitive closure and other recursive definitions.

The example knowledge base in Datalog reads

```
title(http://www.xyz.com/ai.html, 'Artificial
      Intelligence').
type(http://www.xyz.com/ai.html, Book).
title(http://www.xyz.com/sw.html, 'Software
      Engineering').
type(http://www.xyz.com/sw.html, Book).
title(http://www.xyz.com/pl.html, 'Prolog').
type(http://www.xyz.com/pl.html, AI-Book).
```

Each RDF repository can be viewed as a set of ground assertions either using binary predicates as shown above, or as ternary statements “s(S,P,O)”, if we include the predicate as an additional argument. In the following examples, we use the binary surface representation.

Example Query in (binary) Datalog notation.

```
aibook(X) :- title(X, 'Artificial Intelligence'),
            type(X, Book).
aibook(X) :- type(X, AI-Book).
?- aibook(X).
```

Since our query is a disjunction of two (purely conjunctive) subqueries, its Datalog representation is composed of two rules with identical heads. The literals in the rules’ bodies directly reflect RDF statements with their subjects being the variable X and their objects being bound to constant values such as ‘Artificial Intelligence’. Literals used in the head of rules denote derived predicates (not necessarily binary ones). The query expression “aibook(X)” asks for all bindings of X, which conform to the given Datalog rules and our knowledge base, with the results:

```
aibook(http://www.xyz.com/ai.html)
aibook(http://www.xyz.com/pl.html)
```

Internally Edutella Peers use a Datalog based model to represent queries and their results. Figure 2 visualizes this data model as UML class diagram. The Edutella Wrapper API includes the ECDM as well as wrappers for different query languages, and is available as source code from the Edutella Project Page¹. In section 5 we describe an extension of this model based on the TRIPLE language [32], adding model IDs (to explicitly distinguish between different models/peers) and function symbols (useful to construct new IDs for models, resources and for complex values expressed as terms instead of strings).

Our current prototype environment features a set of different peers to demonstrate various aspects of the translation from ECDM to local query languages. It contains the QEL query exchange mechanism, a simple mediator and the wrapping of different repository peer types, including an OLR (Open Learning Repository) based peer [9] using a subset of IEEE/IMS LOM RDF metadata stored in a relational database, a dbXML-based peer [27] as a prototype for an XML repository

¹ <http://edutella.jxta.org/>

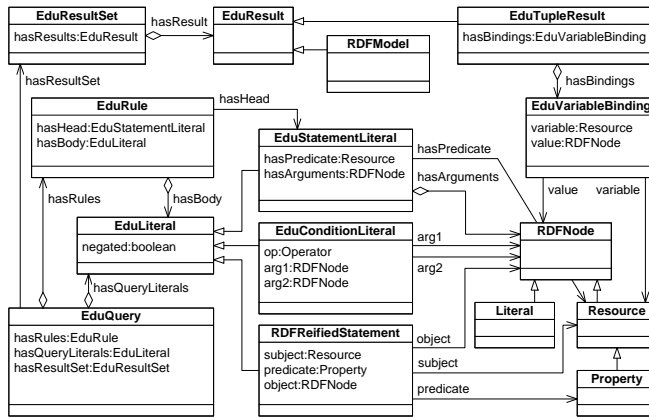


Fig. 2 Edutella Common Data Model (ECDM)

using a simple mapping service to translate from RDF-QEL queries (subset of conjunctive queries) to Xpath queries over the appropriate XML-LOM schema, AMOS-II-based peers [28] with local repositories, KAON-based peers [22] allowing remote annotation [14] using an RDF-based ontology format, and an O-Telos-Peer [17] and [34]. The environment also supports the design and integration of other tools which make use of RDF metadata. The Ont-O-Mat editor and annotation tool allows distributed annotation of documents within the Edutella infrastructure, as described in [25], the concept browser Conzilla [26] provides a graphical query representation.

3 HyperCuP Peer-to-Peer Routing

Obviously, routing in P2P networks is essential for scaling up the network. Most P2P networks evolve in an unorganized manner, and are prone to suffer from serious scalability problems, limiting the number of nodes in the network, creating network overload and pushing search times to unacceptable limits. In almost all P2P networks requests for services use flooding algorithms which are based on inefficient broadcast mechanisms. We achieve efficiency by organizing peers into a hypercube (or, more general, a Cayley graph) topology.

3.1 Organizing Peers into a Hypercube Graph

Figure 3a depicts a hypercube for a base $b = 2$, a topology that has been studied before in the area of multiprocessor machines [18], but under different assumptions. A complete hypercube graph consists of $N = b^{L_{max}+1}$ nodes and is defined by the fact that all nodes have $(b - 1) \cdot (L_{max} + 1)$ neighbors, $(b - 1)$ in each 'dimension' - where $L_{max} + 1$ is essentially the number of dimensions spanned by the cube (in Figure 1, the cube has three dimensions, and L_{max} is 2). The network diameter, defined as the shortest path between most distant nodes in terms of node

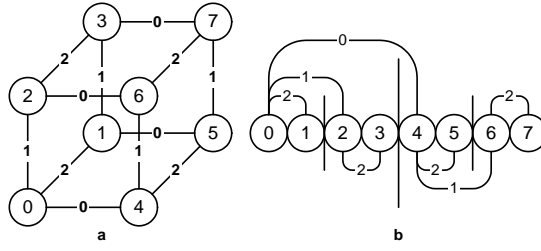


Fig. 3 a. Hypercube graph b. Serialized notation (links incomplete)

hops, is $\Delta = \log_b N$. As visible, this structure is symmetric, i.e. no node incorporates a more prominent position than others. This is crucial for load balancing in the network: Every node can become the source of a broadcast (the root of a spanning tree of the network), yet the load will always be shared equally. The topology provides redundancy - its connectivity (the minimum number of nodes to be removed in order to partition the graph) is optimal, i.e. equal to $\text{node degree} - 1$. Power-law networks such as Gnutella can easily be partitioned by bringing down highly connected nodes in the network through denial of service attacks, the hypercube topology is far less vulnerable to such attacks. The hypercube base b can be chosen to adjust the network diameter and node degree. Note at this point that the construction algorithm works well with node numbers that are not equal to those in complete hypercubes, allowing for any number of peers in the network. To describe the topology of a graph $G = (V, E)$, we state some definitions. In the following, we will deal with hypercubes with a binary base for brevity. Edges in the graph are labeled: Node Y is dubbed i -neighbor of node X or $Y = iN(X)$ iff node Y is X 's neighbor in dimension i . For example, in Figure 3, node 5 is the 2-neighbor of node 4. Node 5 is also dubbed 4's neighbor in dimension 2. Edges in the graph are undirected, i.e. node 4 is also 5's 2-neighbor. A node can have extended neighbors $Y = N(X) = \{x_0, x_1, \dots\}(X)$, where N is termed neighbor link set, and it denotes the sequence of i -neighbors one would have to follow in the complete hypercube graph to reach node Y from node X and vice versa. In our example, the neighbor link set $\{0, 1\}$ leads from node 1 to node 7 and back, i.e. $1 = \{0, 1\}(7)$ and $7 = \{0, 1\}(1)$. Edge labels start at $i = 0$. The maximum dimension of a node is termed L_{max} . A node associates each of its neighbors with a link set and a transport network address.

3.2 Broadcast and Search Algorithm

Based on this terminology, we can define a broadcast scheme which guarantees that the set of nodes traversed strictly increases during a forwarding process, i.e. nodes receive a message exactly once. It is guaranteed that exactly $N - 1$ messages are required to reach all nodes in a topology. Furthermore, the last nodes are reached after $\log_b N$ forwarding steps. Any node can be the origin of a broadcast in the network, satisfying a crucial requirement. The algorithm works as follows:

A node invoking a broadcast sends the broadcast message to all its neighbors, tagging it with the edge label on which the message was sent. Nodes receiving the message restrict the forwarding of the message to those links tagged with higher edge labels. As an example, refer to the serialized notation of the network graph in Figure 3b (for clarity, only the links used in the example are depicted - however, one can just copy all links in 3a into this notation to arrive at the full picture): Node 0 sends a broadcast - at first to all its own neighbors, viz. nodes 4, 2 and 1. Node 4 receives the message on a link tagged as a dimension 0 link, i.e. it forwards the message only to its 1- and 2-neighbors, namely 6 and 5. At the same time, node 2 which has received the message on a dimension 1 link forwards it to its 2-neighbor, node 3. In the third forwarding step, node 6 relays the message to node 7, again its 3-neighbor. The characteristic path length [29] in this scheme can be calculated as $L = \frac{1}{L-1} \cdot \sum_{i=1}^{\log_b N} \frac{(b-1)^{\log_b N - i + 1}}{(\log_b N - i)!} \cdot \prod_{j=0}^{\log_b N - i} (i + j)$ which is about $0.5 \cdot \log_b N$. A search in a hypercube is essentially a broadcast with a time-to-live, i.e. a broadcast with a limited scope. It also has a monotonically increasing neighbor set which means that the maximum number of nodes is reached with a given number of messages.

3.3 Building and Maintaining Hypercube Graphs

In the following, we outline a distributed algorithm which allows nodes to build a hypercube topology. Here, the major challenges in P2P networks are as follows: To maintain network symmetry, crucial for P2P networks, any node in the network should be allowed to accept and integrate new nodes into the network. Furthermore, joining and leaving the network are to consume a reasonable amount of message transmission to limit the traffic imposed on the transport network. Clearly, a joining node should not have to register with all nodes in the network, i.e. we would like the protocol to beat a message number of $O(n)$ for node joins and removals.

In the following, we will describe our construction and maintenance of a hypercube P2P topology. The formal description of the algorithm and a proof of its completeness can be found in [29]. The algorithm can be used to build a general class of graphs, the so-called Cayley graphs [3]. Hypercubes are Cayley graphs, as well as a graph called star graph [3] which features even better (sub-logarithmic) properties than a hypercube. Yet, the algorithm can be more easily explained with hypercubes, hence we focus on them through the course of this paper. We will walk through an example by having 9 peers joining a network, and one peer leaving during the process, to elaborate the basic idea of the construction and maintenance algorithm. The construction and maintenance algorithm is based on the notion that nodes in an evolving hypercube graph take over responsibility for more than one position in the hypercube. The idea is to have the hypercube topology of the next biggest complete hypercube graph already implicitly present in the current topology state, i.e. in the link sets of all participating nodes. Upon arrival of new nodes, the complete hypercube topology unfolds as needed. Upon removal of nodes, other nodes jump in to cover the positions previously covered by the

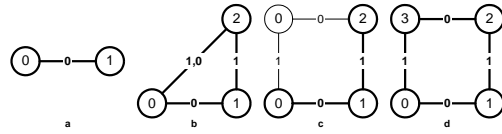


Fig. 4 Network Topology Construction

node that left the topology, prepared to give these positions up again as new nodes join. Since the complete hypercube topology is implicitly preserved, the broadcast and search algorithms do not have to change either - still, every peer receives a broadcast message exactly once.

Start. At the beginning, only peer 0 is active.

Step a. Peer 0 is contacted by node 1 which wants to join the P2P network. Peer 0 integrates peer 1 as 0-neighbor since it does not currently have any other neighbor: The peers establish a link between each other which is tagged with the neighbor set $\{0\}$, as depicted in Figure 4. Generally, a peer integrates a joining peer in its first vacant dimension, the dimensions are ordered such that lower dimensions always come first.

Step b. Peer 2 contacts one of the two peers (here, we assume that it contacts peer 1) to join the network. The first vacant dimension of peer 1 is 1 since it already maintains a 0-neighbor, peer 0. Essentially, peer 1 opens up a new dimension for the hypercube, as depicted in Figure 4b. Peer 1 becomes the so-called integration control node for the complete integration of peer 2 into the network: It is responsible for providing peer 2 with all necessary links - at the end of the integration process, peer 2 has to have neighbor links connecting it all currently existing dimensions, in order to be able to carry out complete broadcasts. Since peer 1 currently has two neighbors, a 0- and a 1-neighbor, it knows that it has to provide peer 2 with a 0- and a 1-neighbor, too. Peer 1 itself has become peer 2's 1-neighbor. Since there is currently no alternative, peer 1 selects peer 0 as the new 0-neighbor for peer 2. However, peer 0 can only become a temporary 0-neighbor for peer 2 because it already has another 0-neighbor, namely peer 1 - and we said before that a peer can only have one neighbor per dimension. Essentially, peer 0 now covers a vacant position in the hypercube, i.e., it acts as if it occupies two positions in the hypercube, as depicted by the thin copy of peer 0 in Figure 2c. To mark the link between peers 2 and 0 as temporary relationship, it is tagged with the link set $\{0, 1\}$ (instead of $\{0\}$): This link set denotes the path from peer 2 via the position at which the link set is originally aiming to peer 0, the peer which currently covers this position. (This path is also well visible in Figure 4c.) Temporary link sets are always constructed by this rule.

Step c. Peer 3 wants to join the network. We compare three cases, viz. peer 3 contacting peer 0, 1 or 2 to join the network. In case peer 3 contacts peer 0 to join, peer 0 follows the general rule to integrate the peer in its first vacant dimension - which is 1, since peer 0 has a 0-neighbor, but no 1-neighbor. As its new 1-neighbor, peer 3 will now cover the temporary position that peer 0 used to maintain in the hypercube: Hence peer 0 can pass on links that are associated with this position to peer 3. Due to the construction rule of edge labels for temporary link sets, peer 0

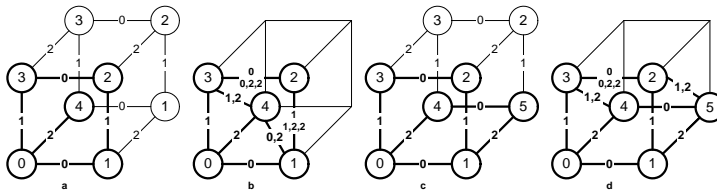


Fig. 5 Network Topology Construction Continued

is able to determine that link $\{0, 1\}$ to peer 2 is a link that is to be passed on to peer 3. Peer 3 then establishes a link tagged by link set $\{0\}$ to peer 3, as depicted in Figure 4d. In case peer 3 contacts peer 2 to join, peer 2 decides to integrate peer 3 as its new (and non-temporary) 0-neighbor. However, it does not carry out the integration itself: Since peer 0 currently covers the position that will soon be occupied by peer 3, the integration control responsibility has to be forwarded to peer 0. Peer 2 can do so via peer 0. Note that it is always possible for peers in the network to reach the node to which they have to forward the control integration, if necessary, in one hop. We prove this in [29]. Peer 0 carries out the integration just as described above, arriving at Figure 4d. In case peer 3 contacts peer 1, peer 1 will integrate peer 3 in dimension 2, i.e., it opens up a new dimension for the hypercube. This leads to a momentary imbalance in the hypercube with some peers maintaining more links than others. To preserve network balance, joining nodes carry out a random walk (at maximum of length $\log_b N$) with increasing probability of choosing the currently visited node as integration champion. Simulations show that e.g. power-law join behavior observed in Gnutella-style networks leads to a fairly balanced network using this technique.

Step d. Peer 4 arrives and contacts peer 0. Now, the network crosses a threshold - a hypercube with 2 dimensions cannot accommodate 5 peers, hence a third dimension is opened up. Peer 0 integrates peer 4 in its first vacant dimension as its new 2-neighbor. Peer 4 needs 3 neighbors, one in each dimension - but neither peer 0's 1-neighbor, peer 3, nor peer 0's 0-neighbor, peer 1, are linked to their own 2-neighbor which they could provide as a new neighbor to peer 4. Thus, peer 3 acts as temporary 1-neighbor for peer 4, whereas peer 1 acts as temporary 0-neighbor for peer 4, indicated once again by the link sets $\{0, 2\}$ and $\{1, 2\}$ among these peers (see Figure 5b). Figure 5a shows the existing peers in the network in bold style and the positions that are additionally covered by them in thin style. Once again, note that the positions that are additionally covered by peers determine the temporary connections the peers have to maintain, plus their edge labels. Figure 5a also demonstrates another basic rule: Peers that are 'closest' to a vacant position in the hypercube structure are always chosen to cover it. Here, 'closest' means that the peer in the highest dimension to a vacant position covers it. (In the more complicated case when a peer has to cover several positions, a peer covers the power set of its vacant dimensions.) Among the other peers in the network, adding another dimension to the graph means the multiplication of existing links, too: For example, peers 1 and 2 could now both integrate 2-neighbors, which would then be

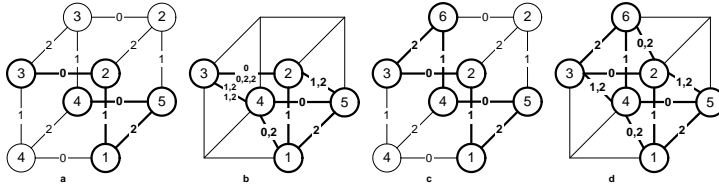


Fig. 6 Network Topology Construction Continued

linked in dimension 1. Thus, they tag their already existing $\{1\}$ link additionally as $\{1, 2, 2\}$ link. So do peers 2 and 3 with their already existing $\{0\}$ link.

Step e. Peer 1 is contacted to integrate the newly arriving peer 5. Peer 1 is still lacking a 2-neighbor, thus peer 5 will be integrated on this position (Figure 5d). Now, peer 1 can get rid of its $\{1, 2, 2\}$ link to peer 2: It is moved to peer 5. However, since 2 is not peer 5's final 1-neighbor either, the link stays temporary: Peers 2 and 5 now maintain a $\{1, 2\}$ link among them. Peer 5 takes over peer 1's temporary $\{0, 2\}$ link to peer 4, which still lacks its final 0-neighbor. It has found one now, namely peer 5.

Step f. Let us assume that peer 0 suddenly leaves the network. In the maintenance protocol, it is obliged to carry out a peer removal process: Basically, it decides which existing peers that it knows will be chosen to take over responsibility for the positions it gives up. In our example, peer 0 leaves only one position vacant, its original position in the graph - however, a node which covers multiple positions will have to find successors for each of its positions in the graph. Peer 4 takes over peer 0's position, establishing temporary links to the former neighbors of peer 0, peers 1 and 3. Figure 6a shows the new distribution of covering responsibilities, Figure 6b depicts the link structure that arises from this network state.

Step g. Peer 4 is contacted by peer 6 and decides to integrate it as its new 1-neighbor. This position is currently covered by peer 3, hence peer 4 forwards the integration control to peer 3, just as described in step c. In the example, all temporary links that are currently owned by peer 3, but originally belong to the new position of peer 6, are restored and passed on to peer 6. Additionally, peer 3 integrates peer 6 as its new 2-neighbor, arriving at Figure 6d.

Step h. Peer 6 is contacted by peer 7, leading to peer 7's integration as peer 6's new 0-neighbor. Figure 7a and Figure 7b depict the state of the network: Almost all positions of a complete hypercube graph with 3 dimensions are held by active

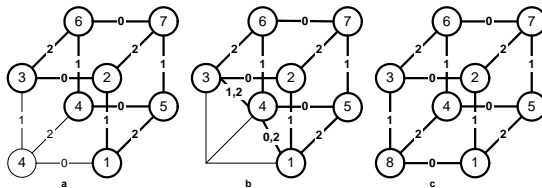


Fig. 7 Network Topology Construction Continued

peers, only peer 4 still covers two positions in the hypercube. What if several peers want to join the network simultaneously? We are currently working on turning our protocol into a real-time protocol, dealing with simultaneous node joins and departures. For example, parallel joins can be executed easily when join actions are time-stamped.

Step i. On integrating peer 8, peer 4 pushes its links $\{1, 2\}$ and $\{0, 2\}$ to its new 2-neighbor, arriving at a complete hypercube topology again.

Link failures. A link failure in the network leads to a node's immediate departure from the P2P topology, not being able to send any departing messages. If that happens, the topology must be able to recover and head back to a normal state. In the hypercube graph, we can always recover from a sudden node loss. The procedure can be found in detail in [29]: The node that is closest to the vanished node (in terms of a metric we call graph hop distance which uses the dimension order to compute a distance value between positions in the hypercube) contacts the vanishing node's neighbors by asking its own neighbors for them. The node then carries out the node departure routine on behalf of the vanished node.

3.4 Complexity

Assuming a relatively balanced graph structure, the algorithm as described above yields an $O(\log_b N)$ complexity in terms of messages that have to be sent in order to join or leave the network. More precisely, this holds when there are only nodes in the graph that have $\lfloor \log_b N - 1 \rfloor$ or $\lfloor \log_b N \rfloor$ non-missing dimensions. Note that this allows for any number of nodes in the graph. To arrive at this complexity order, the algorithm uses optimizations not explained in detail in our walk-through. For example, if a new hypercube dimension is opened up by integrating an additional peer (as has happened above in step d), this information is not broadcast to all peers in the network - instead, it is propagated only when necessary, i.e. once again when nodes communicate on the issue of removing or integrating a peer. Networks that reach a large number of nodes can scale down again to a small number of nodes (as long as this takes place relatively balanced, see Section 3.3): Higher dimensions that are added during the construction process are removed again if no peer in the network has any neighbor in a dimension any more. Once again, this information is not broadcast in the network but locally inferred by every peer by observing its set of neighbor link sets it maintains.

4 Schema-Based Routing

In the last section, we have discussed a very efficient and scalable topology for broadcast networks. To take the semantic heterogeneity of schema-based P2P networks into account, we now extend this topology to a super-peer topology, where the HyperCuP network constitutes the "backbone" of our P2P network, and where additional routing indices take care of message routing and integration / mediation of metadata, based on the schemata used in our P2P network. A new super-peer is able to join the network by asking any other, already integrated super-peer which

then carries out the peer integration protocol (as shown in 3.3). For broadcasts, each node can be thought of as the root of a specific spanning tree through the P2P super-peer network, and the topology guarantees efficient communication and message forwarding among super-peers. As a path of $\log_2 N$ length exists between any two super-peers (see 3.2, any two distinct schemas can be reached within a short number of hops from each other. Peers then connect to the super-peers in a star-like fashion, providing content and content metadata, as shown in Figure 8.

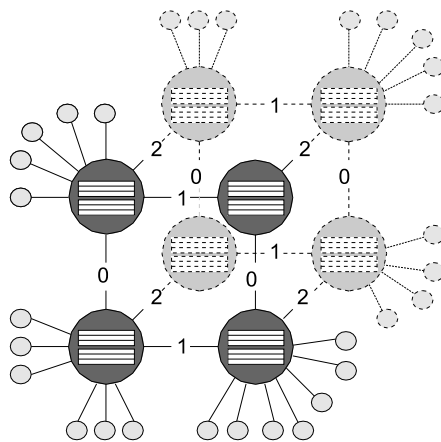


Fig. 8 Peers Connected to the Super-Peer “Backbone”

The introduction of super-peers in combination with routing indices reduces the workload of peers significantly by distributing queries only to the appropriate subset of all possible peers (see also [7] who discusses routing indices based on various aggregation strategies of content indices). In the next sections we will discuss these routing indices in more detail.

4.1 Routing Super-Peer/Peer Queries and Responses

The first kind of indices needed in super-peers are so-called super-peer/peer routing indices (SP/P-RIs). In these indices each super-peer stores information about metadata usage at each peer. This includes schema information such as schemas or attributes used, as well as possibly conventional indices on attribute values. On registration the peer provides the super-peer with its metadata information by publishing an advertisement. This advertisement encapsulates a metadata based description of the most significant properties of the peer. As this may involve quite a large amount of metadata, we build upon the schema-based approaches which have successfully been used in the context of mediator-based information systems (e.g. [33]).

To ensure that the indices are always up-to-date, peers notify super-peers when their content changes in ways that trigger an update of the index. For example, if

Granularity	Query	
Schema	dc, lom	
Property	dc:subject, dc:language, lom:context	
Property Value Range	dc:subject	ccs:sw'engineering
Property Value	lom:context dc:language	"undergrad" "de"

Table 1 Contents of the Sample Query at Different Granularities

a peer had announced that it uses the Dublin Core schema `dc` during the last connection to its super-peer, but now also uses the Learning Object Metadata schema `lom` to describe resources, it needs to announce this to the super-peer. If a peer leaves the network, all references to this peer are removed from the indices. In contrast to some other approaches, our indices do not contain content elements but peers (as in CHORD). At each super-peer, elements used in a query are matched against the SP/P-RIs in order to determine local peers which are able to answer the query (see also [1] for a similar approach). A match means that a peer understands and can answer a specific query, but does not guarantee a non-empty answer set. The indices can contain the information about other peers or super-peers at different granularities: schema identifiers, schema properties, property value ranges, and individual property values.

To illustrate index usage, we will use the following sample query: *find lectures in German language from the area of software engineering suitable for undergraduates*. In the Semantic Web context this query would probably be formalized using the `dc` schema for document specific properties (e.g. title, creator, subject) and the `lom` schema which provides learning material specific properties, in combination with classification hierarchies (like the ACM Computing Classification System, ACM CCS) in the subject field. In line with RDF/XML conventions, we will identify properties by their name and their schema (expressed by a namespace): "`dc:subject`" therefore denotes the property "subject" of the DC schema. So, written in a more formal manner, the query becomes:

Find any resource where the property `dc:subject` is equal to `ccs:softwareengineering`, `dc:language` is equal to "de" and `lom:context` is equal to "undergrad".

Table 1 shows the values requested in the query at the different granularities; e.g. the query asks for DC and LOM at the schema level, while it requests a `lom:context` value of "undergrad" at the property value level, etc.

In order to further clarify things we consider the scenario shown in figure 9. In this network, various resources are described on different peers, which in turn are attached to super-peers. Peer P_0 sends the sample query mentioned above to its super-peer SP_1 . In our example, this query could be answered by the peers P_1 and P_4 , attached to SP_1 and SP_4 , respectively. These contain metadata about resources r and s which match the query. The following paragraphs will explain how the routing indices at the different granularities facilitate routing the query to the right peers.

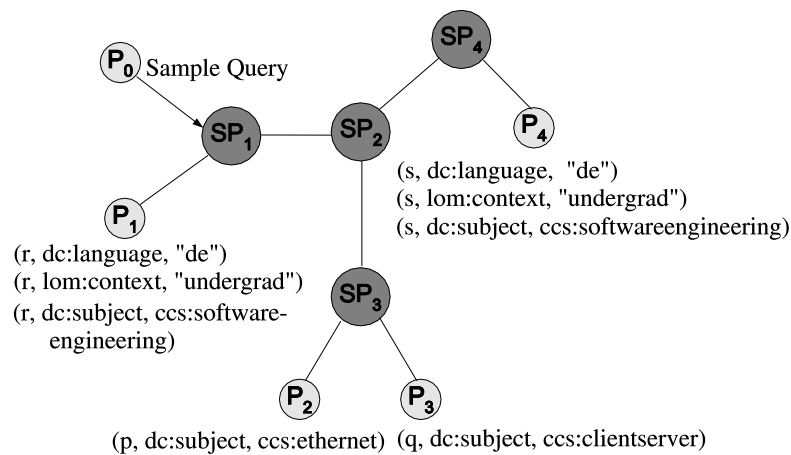


Fig. 9 Routing Example Network

Schema Index We assume that different peers will support different schemas and that these schemas can be uniquely identified (e.g. the `dc` and `lom` namespaces are uniquely identified by an URI). The routing index contains the schema identifier as well as the peers supporting this schema. Figure 10 shows a sample of such an index. Queries are forwarded only to peers which support the schemas used in the query. Super-peer SP_1 will forward the sample query to attached peers which use DC and LOM to annotate resources (peer P_1 in Figure 10). Support for mediation [6] between different schemas will be added in the future.

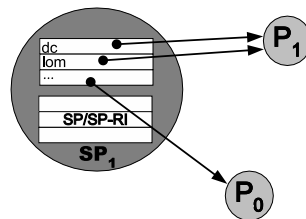


Fig. 10 Sample Super-Peer/Peer Routing Index of SP_1

Property/Sets of Properties Index Peers might choose to use only parts of (one or more) schemas, i.e. certain properties, to describe their content. While this is unusual in conventional database systems, it is more often used for data stores using semi-structured data, and very common for RDF-based systems. In this kind of index, super-peers use the properties (uniquely identified by namespace/schema ID plus property name) or sets of properties to describe their peers. Our sample query will be sent to peers using at least `dc:subject`, `dc:language` and `lom:context` (e.g. SP_1 will send the query to P_1 , as P_1 contains all of these properties). Sets

of properties can be useful to characterize queries (i.e. we might use a “sets-of-properties index” to characterize and route the most common queries).

Property Value Range Index For properties which contain values from a predefined hierarchical vocabulary we can use an index which specifies taxonomies or part of a taxonomy for properties. This is a common case in Edutella, because in the context of the semantic web quite a few applications use standard vocabularies or ontologies. In our example, peers could be characterized by their possible values in the `dc:subject` field, and the query would not be forwarded to peers managing “`ccs:networks`” or “`ccs:artificial_intelligence`” content (as these sub-hierarchies are disjoint from the `ccs:software_engineering` sub-hierarchy), and will not be forwarded to peers which use the MeSH vocabulary (because these peers manage medical content). Note that the subsumption hierarchy in a taxonomy such as ACM CCS can be used to aggregate routing information in order to reduce index size.

Property Value Index For some properties it may also be advantageous to create value indices to reduce network traffic. This case is identical to a classical database index with the exception that the index entries do not refer to the resource, but the peer providing it. This index contains only properties that are used very often compared to the rest of the data stored at the peers. In the example, this is used to index string valued properties such as `dc:language` or `lom:context`.

4.2 Routing among Super-Peers based on Routing Indices

As with peers, we want to avoid broadcasting queries to all super-peers. To achieve this goal we introduce super-peer/super-peer routing indices to route among the super-peers. These SP/SP indices are essentially extracts and summaries (possibly also approximations thereof) from all local SP/P indices. They contain the same kind of information as SP/P indices, but refer to the (direct) neighbors of a super-peer. Queries are forwarded to super-peer neighbors based on the SP/SP indices, and sent to connected peers based on the SP/P indices.

Table 2 gives a full example of the SP/SP routing index of SP_2 at the different granularities. For example, SP_2 knows at the schema level that all of its neighbors (SP_1 , SP_3 , SP_4) use the DC namespace, but only SP_1 and SP_4 contain information described in the LOM schema. Thus, the sample query will not be routed to SP_3 , as it requires both DC and LOM. The same applies for the other levels of granularity. A special case is the Property Value Range level; note that `ccs:networks` is a common super concept of `ccs:ethernet` and `ccs:clientserver` in the ACM CCS taxonomy. Making use of the topic hierarchy, the routing index can contain aggregate information like this in order to reduce index size.

Update of SP/SP indices is based on the registration (or update) messages from connected peers. We assume for the moment that a peer can connect to an arbitrary super-peer and define the index update procedure as follows: when a new peer registers with a super-peer, it announces the necessary schema (and possibly content)

Granularity	Index of SP_2		
Schema	dc lom	SP_1, SP_3, SP_4 SP_1, SP_4	
Property	dc:subject dc:language lom:context	SP_1, SP_3, SP_4 SP_1, SP_4 SP_1, SP_4	
Property Value Range	dc:subject dc:subject	ccs:networks ccs:software- engineering	SP_3 SP_1, SP_4
Property Value	lom:context dc:language	“undergrad” “de”	SP_1, SP_4 SP_1, SP_4

Table 2 Sample SP/SP Index of SP_2 at Different Granularities

information to the super-peer. The super-peer matches this information against the entries in its SP/P index. If new elements have to be added in order to include the peer into the SP/P index, the super-peer broadcasts an announcement of the new peer to the super-peer network (according to the HyperCuP protocol, so that it reaches each super-peer exactly once). The other super-peers update their SP/SP indices accordingly.

Although such a broadcast is not optimal, it is not too costly either. First, the number of super-peers is much less than the number of all peers. Second, if peers join the super-peer frequently, we can send a summary announcement containing all new elements only in pre-specified intervals instead of sending a separate announcement for each new peer. Third, an announcement is necessary only if the SP/P index changes because of the integration of the new peer. As soon as the super-peer has collected a significant amount of peers (hopefully with the same characteristics, see our discussion on clustering in the next section), these announcements will rather be an exception. Similarly, indices have to be updated when peers disconnect from their super-peers.

The process of super-peers joining the network consists of two parts, namely, taking the appropriate position in the HyperCuP topology and announcing themselves to their neighbors, so that these can update their SP/SP indices accordingly. The HyperCuP protocol handles the proper positioning and bookkeeping with regard to the topology of the super-peer network. Announcing a new super-peer to its neighbors and updating their SP/SP indices works similarly as the construction of SP/P indices described in section 4.1.

If a super-peer fails, its formerly connected peers must register with another super-peer chosen at random (or find one that contains similar peers as described in the next section). The respective SP/SP indices entries at other super-peers are removed based on the dynamic optimizations described in the next chapter. Obviously, with an arbitrary distribution of peers to super-peers, the majority of all queries would still have to be sent to most super-peers. Therefore we have to investigate clustering techniques based on peer characteristics, which we will discuss in the next section.

4.3 Dynamic Routing Indices

In the last section we described how queries can be routed using schema-based routing indices. This routing still has the problem that most queries must be broadcast if the peer distribution is arbitrary. In order to avoid broadcasting as much as possible the SP/P and SP/SP routing indices have to be extended with additional frequency information about queries. This allows us to adapt the network topology and peer clustering based on this frequency information, as discussed in the following paragraphs.

Similarity-Based Clustering of Peers. Clustering in our super-peer network is based on the idea of integrating peers into locations already populated with peers of similar characteristics. In contrast to randomly assigning peers to super-peers this will reduce the amount of messages sent in the network. HyperCuP is a deterministic topology which partitions and sub-partitions the network in a regular way. Connections from a super-peer to its neighbors can be viewed as connections into other partitions and sub-partitions. Forwarding a query to a subset of neighbors therefore results in the distribution of the query within a subgraph. As discussed before, HyperCuP partitions are connected redundantly using links with different dimensions, and broadcast messages which arrive along a dimension k are forwarded only along links with dimension $i > k$. If we assume for example that the super-peers 4 to 7 in Figure 3 all manage peers with the same characteristics, which are not present at super-peers 0 to 3, a query using these characteristics just has to be forwarded once to the right cluster using a dimension 0 link, and then broadcast to all super-peers in this partition. If a query uses the characteristics of the peers attached to super-peers 0 and 1, we just have to reach this cluster and then distribute the query to the appropriate peers of these two super-peers (based on their SP/P indices).

Obviously we still have to define what kind of similarity measures we want to use for our partitions. In [30] we have discussed how to partition a HyperCuP-based P2P network based on a topic ontology shared by all peers. Another, more dynamic way is to take the characteristics of queries into account when deciding on the clustering parameters. As query characteristics in a large and possibly heterogeneous P2P network cannot be defined in advance, we propose to use frequency counting algorithms on streams such as the ones discussed in [23] to identify the most commonly used schemas, properties and sets of properties in the sent queries. This means that we have to add a frequency property in our SP/SP routing indices, that we focus on including the most frequently used index items in our indices, and that we use these statistics to define our similarity measures responsible for clustering peers to super-peers. Queries not covered by these (possibly incomplete) SP/SP indices can be broadcast in the super-peer HyperCuP network, and then forwarded to the appropriate peers based on the SP/P indices.

The algorithms for estimating the frequencies of specific items discussed in [23], such as sticky sampling and lossy counting, can both be used in this context, as they scan the input stream only once, and do not need much temporary storage to hold the frequency counts. In our scenario, items to be counted could

be schemas, schema properties or value entries from a taxonomy, thus clustering the peers according to the schemas or schema properties used, or to the taxonomy entries (or their super-topics) most commonly used in the queries. The algorithm for estimating frequencies of item sets (instead of single items) from [23] is more difficult to use because of its space requirements, though it would be useful to characterize queries as sets of used properties and value ranges and cluster the peers accordingly.

5 Model Transformation and Querying using the TRIPLE Language

To use RDF as the underlying syntactic representation enables one to define a common infrastructure, which is usable for every kind of data, regardless of the specific data model used to express semantics. This, however, does not solve the problem of integrating data based on different data models entirely. Although many query languages and inference engines for RDF exist (e.g., SiLRI [8], RQL [19], SQUISH²) none of them is capable of supporting multiple semantics as required by the different heterogeneous data models. Support for different kind of semantics in the query engines is required e.g., when querying UML data the *Generalization* relation should be treated as a transitive relationship, as well as the `rdfs:subClassOf` relationship in RDF Schema. None of the cited query languages have the abilities to query different data models with different kind of semantics, and also for RDF-QEL we have assumed no built-in semantics (RDF data are queried as they are stored, no additional inferences are performed). Although some of them (RQL and SiLRI) have a built-in semantics for RDF Schema (transitivity of `rdfs:subClassOf`, etc.), this does not generalize to other data models.

To address the issues of multiple semantics and multiple distributed RDF models we propose TRIPLE [32]. TRIPLE supports most features of RDF with the exception of containers and defines a logic able to reason with RDF and RDF models and to transform RDF models taking a specified semantics into account, thus going beyond the pure Datalog-based abilities of RDF-QEL.

5.1 Features of TRIPLE

In the following, the main features of TRIPLE (i.e., those extending Horn logic and Datalog) are informally described.

Namespaces and Resources TRIPLE has special support for namespaces and resource identifiers. Namespaces are declared via clause-like constructs of the form *nsabbrev* := *namespace*., e.g.

```
rdf := "http://www.w3.org/1999/02/22-rdf-syntax-ns#".
```

² See <http://ilrt.org/discovery/2000/10/swsql/>

Resources are written as *nsabbrev:name*, where *nsabbrev* is a namespace abbreviation and *name* is the local name of the resource. Resource abbreviations can be introduced analogously to namespace abbreviations, e.g.

isa := rdfs:subClassOf.

Statements and Molecules An RDF statement (triple) is—inspired by F-Logic object syntax—written as

subject[*predicate* → *object*]

Several statements with the same subject can be abbreviated as “molecules”:

xyz:ai.html[title → Artificial Intelligence; type → Book; . . .]

RDF statements (and molecules) can be nested, eg.:

xyz:ai.html[author → xyz:emp1[grade → PhD]]

Models RDF models, i.e., sets of statements³, are made explicit in TRIPLE (“first class citizens”). Statements, molecules, and also Horn atoms that are true in a specific model are written as *atom@model* (similar to Flora-2 module syntax), where *atom* is a statement, molecule, or Horn atom and *model* is a model specification (i.e., a resource denoting a model), e.g.

xyz:pl.html[title → *Prolog*]*@factsAboutBooks*

TRIPLE also allows Skolem functions as model specifications. Skolem functions can be used to transform one model (or several models) into a new one when used in rules (e.g., for ontology mapping or integration):

$O[P \rightarrow Q]@sf(m1, X, Y) \leftarrow \dots$

If all (or many) statements/molecules or Horn atoms in a formula (see Section 5.1) are from one model, the following abbreviation can be used: *formula@model*. All statements/molecules and Horn atoms in *formula* without an explicit model specification are implicitly suffixed with *@model*. Instead of constants, variables, and Skolem functions also boolean combinations can be used, eg.: (*model*₁ ∩ *model*₂) specifying the intersection of two models, (*model*₁ ∪ *model*₂) specifying the union of two models, and (*model*₁ \ *model*₂) specifying the set-difference of two models.

Reified Statements Reified statements are written as <*statement*> and can be used inside other statements, allowing “modal” statements like

stefan[believes → <Ora[isAuthorOf → homepage]>]

³ The notion of *model* in RDF does not coincide with its use in (mathematical) logics.

Path Expressions For navigation purposes, path expressions have proven to be very useful in object oriented languages. TRIPLE allows the usage of path expressions instead of subject, predicate, or object definitions (and at all other places where terms are allowed). Path expressions are dot-delimited sequences of resources, e.g.:

stefan.spouse.mother

denotes Stefan's mother in law.

Logical Formulae TRIPLE uses the usual set of connectives and quantifiers for building formulae from statements/molecules and Horn atoms, i.e., \wedge , \vee , \neg , \forall , \exists , etc. All variables must be introduced via quantifiers, therefore marking them is not necessary (i.e., TRIPLE does not require variables to start with an uppercase letter as in Prolog).

Clauses and Blocks A TRIPLE clause is either a fact or a rule. Rule heads may only contain conjunctions of molecules and Horn atoms and must not contain (explicitly or implicitly) any disjunctive or negated expressions. To assert that a set of clauses is true in a specific model, a model block is used:

$@model \{clauses\}$

or, in case the model specification is parameterized:

$\forall Mdl \ @model(Mdl) \{clauses\}$

5.2 Example: Querying Multiple Models

This section presents an example of how TRIPLE is useful with respect to multiple models and explicitly represented semantics. The Dublin Core Metadata Initiative [20] defines a set of elements for marking up documents with metadata like title, creator, date, subject, etc. An encoding of Dublin Core metadata in RDF is straightforward. The example in Figure 11 shows how data model semantics can be represented and used with TRIPLE using multiple models. An ontology is used to query a Dublin Core Model with respect to background knowledge, here represented by an ontology. The first part of figure 11 depicts the necessary rules for computing the semantics of RDF Schema. The set of rules requires one parameter. The parameter represents the model containing the ontology of which the RDF Schema semantics should be computed. The next set of RDF statements consists of a small ontology defining some sub-classes of Motor Vehicles, among them a Mini-Van. The next model then defines a set of Dublin Core statements, describing a book about Minivans. The next two sets of statements describe the actual query - the first model defines how to search a dublin core metadata set with respect to a certain terms. The last statements represents the query, asking for all books about Motor Vehicles. Since the ontology says that MiniVan is a sub-class of Motor Vehicle, the search rule returns the book. Each of the models could be located on a different peer in the P2P network.

```

rdf := 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
rdfs := 'http://www.w3.org/2000/01/rdf-schema#'.
FORALL Mdl @rdfschema(Mdl) {
  FORALL O,P,V O[P->V] <-
    O[P->V]@Mdl.
  FORALL O,P,V O[P->V] <-
    EXISTS S S[rdfs:subPropertyOf->P]
      AND O[S->V].
  FORALL O,P,V O[rdfs:subClassOf->V] <-
    EXISTS W (O[rdfs:subClassOf->W]
      AND W[rdfs:subClassOf->V]).
  FORALL O,P,V O[rdfs:subPropertyOf->V] <-
    EXISTS W (O[rdfs:subPropertyOf->W]
      AND W[rdfs:subPropertyOf->V]).
  FORALL O,T O[rdf:type->T] <-
    EXISTS S (S[rdfs:subClassOf->T]
      AND O[rdf:type->S]).
}
@cars {
  xyz := "http://www.w3.org/2000/03/example/vehicles#".
  xyz:MotorVehicle[rdfs:subClassOf -> rdfs:Resource].
  xyz:PassengerVehicle[rdfs:subClassOf -> xyz:MotorVehicle].
  xyz:Truck[rdfs:subClassOf -> xyz:MotorVehicle].
  xyz:Van[rdfs:subClassOf -> xyz:MotorVehicle].
  xyz:MiniVan[
    rdfs:subClassOf -> xyz:Van;
    rdfs:subClassOf -> xyz:PassengerVehicle].
}
rdf := "http://www.w3.org/1999/02/22-rdf-syntax-ns#".
dc := "http://purl.org/dc/elements/1.0/"
motor := "http://www.auto.com/#"
@motor:books {
  motor:d_01_01[dc:title -> "For Whom the Minivan Rolls";
    dc:creator -> "Jeffrey Cohen";
    dc:subject -> xyz:MiniVan].
}
FORALL DCMModel,Topic, Ontology interestingBooks(DCMModel, Topic,
Ontology){
  FORALL S,P,O,C S[dc:title -> O] <-
    C[rdfs:subClassOf -> Topic]@rdfschema(Ontology)
    AND S[dc:subjects -> C; dc:title -> O].
}
Query: FORALL S,T <- S[dc:type->T]@interestingBooks(motor:books,
xyz:MotorVehicle, cars)

```

Fig. 11 Example with multiple RDF models

6 ELENA: Smart Spaces for Learning

The recently started EU/IST project ELENA [4] builds upon the Edutella infrastructure and aims at creating a smart space for learning. A smart space for learning is a distributed system comprising of educational nodes delivering learning services. Smart spaces for learning support the “smart” mediation of learning services based on user profiling as well as evaluation and reputation services.

A learning service is defined as an event that is managed by a learning service provider in order to support the accomplishment of a specific educational objective. This is achieved by creating a learning environment consisting of educators, educational material, communication infrastructure, meeting places, etc. Examples for a learning service are the delivery of a course, the provision of a web-based training application or self-study material.

6.1 Architecture of Smart Spaces for Learning

In Elena Edutella P2P technology is used to connect heterogeneous kinds of educational nodes. An educational node refers to a system component of a smart space for learning, which facilitates the provision of a learning service. The P2P infrastructure is used to announce the availability of learning services, search for evaluation service providers or obtain the reputation rating of a particular learning service provider. On top of the P2P interfaces educational nodes offer interfaces for contracting and consuming learning services. These interfaces are part of the learning management network.

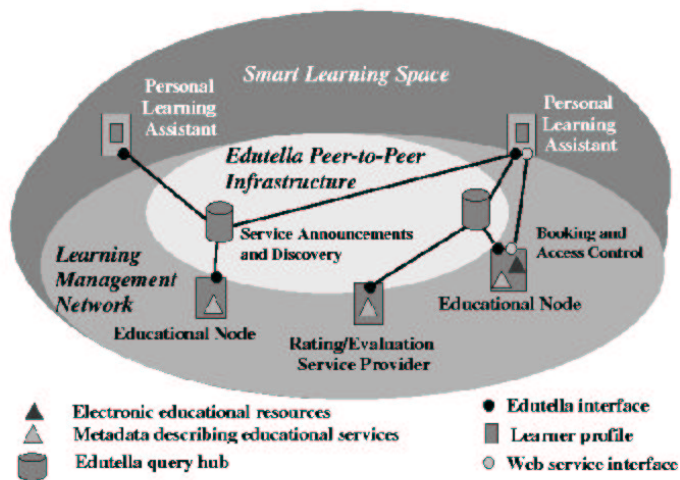


Fig. 12 Architecture of Smart Spaces for Learning

In smart spaces for learning, learning service providers need not to provide their services in a fully electronic manner. For example, some learning service providers just list a number of courses by writing an RDF file, as they did write a static HTML page to describe their courses on the Web. Others provide a full web-based training application and a web-service enabled interface, which allows automatic learner registration and reporting of learning achievements. Reputation service providers facilitate rating-like annotations of learning service providers, which are aggregated from evaluations of learning service deliveries.

On top of the learning management network, personal learning assistants interact with the connected peers in order to search for suitable learning services. A personal learning assistant is a component of a smart space for learning, which supports learners in searching for, selecting and contracting learning services. Personal learning assistants take advantage of the learner profile in order to augment queries and personalize query results. They recommend learning services based

on the profile and have rules implemented, which allow them to automatically perform processes such as course registration. Figure 12 depicts the various components of a smart learning space.

Reputation is the credibility that a community assigns to a learning service and its provider. Evaluation data is one source to determine reputation. Positive learning experiences, which are communicated within a community, increase the reputation of a learning service. The main purpose of reputation ratings is to support the selection process of learning services.

6.2 Ontologies for Learning Services

Ontologies have been identified as one of the most important ingredients in smart spaces for learning. Being defined as explicit, shared specifications of conceptualizations [13], they enable the various actors in these applications to communicate with each other on a high level of abstraction. In a smart space for learning, these actors are educational nodes.

A typical request sent to educational nodes would be "find a tutorial that explains the semantic web to a novice." Unlike in common search engines (like Google), such a request is not sent in its textual representation, but formalized with the help of various ontologies, to specify for example selections like `edu:sercivetype=tutorial`, `dc:subject=Semantic_Web` and/or `edu:level_of_knowledge=novice`.

Here, the document type takes values from an ontology of educational resource types that contains the various types of educational resources (e.g. Case Study, Course, Course Unit, Exam, Exercise, Experiment, Group Work, Lecture, Presentation, Tutorial, etc.). `dc:subject` is instantiated from a domain ontology, in this example, an ontology containing concepts of computer science. Novice comes from an ontology that models the learners' previous knowledge or, in general, learner profiles. An additional ontology that comes into play is a web services ontology that is used by the educational services to describe their capabilities. When creating or adding educational resources, these have to be annotated with the concepts from the above mentioned ontologies. This can either be done manually or semi-automatically, i.e., with the help of linguistic or statistical content analysis.

Finding the desired learning services for a request requires more than just simply matching it with their annotations: When the user requests an educational service on the topic Semantic Web, the system should also list services on e.g., RDF and other main stream Semantic Web technologies. This is accomplished by explicitly representing relationships in the (domain) ontologies like `is_technology_for` plus exploiting them with the help of inference engines (see 5.2 for an example in TRIPLE).

Especially in the case of domain ontologies, it is very likely that the various institutions develop their own ontologies for the same or similar domains independently (or use their existing library indexes to bootstrap them), resulting in largely incompatible ontologies. This requires the development of mappings between concepts from these ontologies. Such mappings can, for example be expressed as rules which are enacted by rule engines like TRIPLE.

7 Conclusions

In this paper we have discussed the open source project Edutella, which combines semantic web and peer-to-peer technologies in order to make distributed learning repositories possible and useful. We have described the basic Edutella infrastructure as an advanced, schema-based peer-to-peer network based on RDF metadata and a common query exchange language (RDF-QEL) enabling distributed queries over the Edutella network. Such a network extends conventional peer-to-peer approaches by allowing different and extensible schemas to describe peer content, an absolutely necessary feature for information rich peer-to-peer networks. We have described the HyperCuP topology, which implements an optimal broadcast topology, and showed how it can be used to form the backbone of a super-peer-based topology suitable for schema-based peer-to-peer networks. We further discussed TRIPLE, which extends the Datalog-based RDF-QEL by additional features to facilitate querying of information sources with heterogeneous semantics and model transformation. Finally, we showed how we use this infrastructure in our recently started EU/IST project ELENA where we are working to create smart spaces for learning.

Acknowledgements This paper discusses the Edutella infrastructure and necessary technologies for this network in more detail as well as its use for implementing smart learning spaces, developed in several projects. The authors gratefully acknowledge the numerous and invaluable contributions of all their co-workers (see also the references to various papers on these topics), and the many discussions we have had on these issues.

Specifically, we want to thank Rudi Studer, Steffen Staab, Julien Tane, Gerd Stumme and Christoph Schmitz from AIFB/Karlsruhe and L3S, Ingo Brunkhorst from KBS/Hannover and L3S, Ambjörn Naeve, Mikael Nilsson and Matthias Palmér from CID/Stockholm, Tore Risch from Uppsala, Gustaf Neumann and Zoltán Miklós from the Vienna University of Economics and Business Administration and Alexander Löser from Technical University Berlin.

References

1. Karl Aberer and Manfred Hauswirth. Semantic gossiping. In *Database and Information Systems Research for Semantic Web and Enterprises, Invitational Workshop*, University of Georgia, Amicalola Falls and State Park, Georgia, April 2002.
2. Technical Team ADL. SCORM Specification v1.2. <http://www.adlnet.org/Scorm/scorm.cfm>, October 2001.
3. S. B. Akers and B. Krishnamurty. A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–566, April 1989.
4. S. Brantner, N. Henze, S. Gunnarsdottir, B. Kieslinger, T. Klobucar, T. Kuechler, W. Nejdl, G. Neumann, J. Quemada, W. Siberski, B. Simon, and G. Vrabic. Elena - creating a smart space for learning. In *Proceedings of the 1st International Semantic Web Conference*. Springer, LNCS 2342, Sardinia, Italy, 2002.
5. Dan Brickley and R. V. Guha. W3C Resource Description Framework (RDF) Schema Specification. <http://www.w3.org/TR/1998/WD-rdf-schema/>, March 2000. W3C Candidate Recommendation.

6. Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of IPSJ Conference*, Tokyo, Japan, October 1994.
7. Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings International Conference on Distributed Computing Systems*, July 2002.
8. Stefan Decker, Dan Brickley, Janne Saarela, and Jurgen Angele. A query and inference service for RDF. In *QL'98 — The Query Languages Workshop*, Boston, USA, 1998. WorldWideWeb Consortium (W3C).
9. Hadhami Dhraief, Wolfgang Nejdl, Boris Wolf, and Martin Wolpers. Open learning repositories and metadata modeling. In *International Semantic Web Working Symposium (SWWS)*, Stanford, CA, July 2001.
10. Rael Dornfest and Dan Brickley. The power of metadata. <http://www.openp2p.com/pub/a/p2p/2001/01/18/metadata.html>, January 2001. excerpted from the book "Peer-to-Peer: Harnessing the Power of Disruptive Technologies".
11. The Edutella Project. <http://edutella.jxta.org/>, 2002.
12. Li Gong. Project JXTA: A technology overview. Technical report, SUN Microsystems, April 2001. <http://www.jxta.org/project/www/docs/TechOverview.pdf>.
13. T. R. Gruber. Conceptual analysis and knowledge representation. In *Toward Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology*. Kluwer Press, 1993.
14. Siegfried Handschuh, Steffen Staab, and Alexander Maedche. CREAM - creating relational metadata with a component-based, ontology-driven annotation framework. In *Workshop on Knowledge Markup and Semantic Annotation at the First International Conference on Knowledge Capture (K-CAP'2001)*, Victoria, BC, Canada, October 2001.
15. IEEE-LTSC. IEEE LOM Working Draft 6.1. <http://ltsc.ieee.org/wg12/index.html>, April 2001.
16. IMS. IMS Learning Resource Metadata Specification v1.2.1. <http://www.imsproject.org/metadata/index.html>, 2001.
17. M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal on Intelligent Information Systems*, 4(2):167 – 192, 1995.
18. S. L. Johnsson and C.-T. Ho. Optimum Broadcasting and Personalized Communication in Hypercubes. *IEEE Transactions on Computers*, 38(9):1249–1268, September 1989.
19. G. Karvounarakis, V. Christophides, D. Plexousakis, and S. Alexaki. Querying community web portals, 2001. Available at <http://www.ics.forth.gr/proj/isst/RDF/RQL/>, 2001. Submitted for publication.
20. Stefan Kokkelink and Roland Schwänzl. *Expressing Qualified Dublin Core in RDF/XML*. DCMI, August 2001. <http://dublincore.org/documents/2001/08/29/dcq-rdf-xml/>.
21. Ora Lassila and Ralph R. Swick. W3C Resource Description framework (RDF) Model and Syntax Specification. <http://www.w3.org/TR/REC-rdf-syntax/>, February 1999. W3C Recommendation.
22. Alexander Maedche, Steffen Staab, Rudi Studer, York Sure, and Raphael Volz. Seal — Tying up information integration and Web site management by ontologies. *IEEE Data Engineering Bulletin*, March 2002. <http://www.research.microsoft.com/research/db/debull/>.

23. Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, China, August 2002.
24. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. EDUTELLA: a P2P Networking Infrastructure based on RDF. In *Proceedings of the 11th International World Wide Web Conference*, Hawaii, USA, May 2002. <http://edutella.jxta.org/reports/edutella-whitepaper.pdf>.
25. Wolfgang Nejdl, Boris Wolf, Steffen Staab, and Julien Tane. Edutella: Searching and annotating resources within an rdf-based p2p network. In *Proceedings of the Semantic Web Workshop, 11th International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.
26. Mikael Nilsson and Matthias Palmér. Conzilla - towards a concept browser. Technical Report CID-53, TRITA-NA-D9911, Department of Numerical Analysis and Computing Science, KTH, Stockholm, 1999. http://kmr.nada.kth.se/papers/ConceptualBrowsing/cid_53.pdf.
27. Changtao Qu and Wolfgang Nejdl. Towards interoperability and reusability of learning resources: A SCORM-conformant courseware for computer science education. In *Proceedings of the IEEE International Workshop on Knowledge Media Networking (IEEE KMN'02)*, Kazan, Tatarstan, Russia, 2002.
28. T. Risch and V. Josifovski. Distributed data integration by object-oriented mediator servers. *Concurrency and Computation: Practice and Experience*, 13(11):933 – 953, 2001.
29. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. HyperCuP. Technical report, Stanford University, April 2002.
30. Mario Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. In *International Workshop on Agents and Peer-to-Peer Computing*, Bologna, Italy, July 2002.
31. Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database Systems Concepts*. McGraw-Hill Higher Education, 4 edition, 2001.
32. Michael Sintek and Stefan Decker. TRIPLE—An query, inference, and transformation language for the semantic web. In *Proceedings of the 1st International Semantic Web Conference*. Springer LNCS, June 2002.
33. G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38 – 49, 1992.
34. Martin Wolpers, Wolfgang Nejdl, and Ingo Brunkhorst. An o-telos provider peer for the rdf-based edutella p2p-network. In *Proceedings of Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM 2002) at 15th European Conf. on Artificial Intelligence*, Lyon, France, 2002.