On Combining a Semantic Engine and Flexible Network Policies for P2P Knowledge Sharing Networks [*]

S. Castano, A. Ferrara, S. Montanelli, E. Pagani, G.P. Rossi, S. Tebaldi
Università degli Studi di Milano
DICO - Via Comelico, 39, 20135 Milano - Italy
{castano,ferrara,montanelli,pagani,rossi,tebaldi}@dico.unimi.it

Abstract

Peer-to-Peer (P2P) systems have recently become popular for content sharing, and a number of different approaches have been studied to perform effective content retrieval in such networks. In this paper, we discuss how conventional P2P network policies for content retrieval could be combined with a semantic engine component to enforce peer-based knowledge sharing. The semantic engine exploits peer ontologies for providing a semantically rich representation of peer knowledge, and matching techniques for comparing ontologies of different peers for knowledge sharing and discovery purposes.

1. Introduction

Peer-to-Peer (P2P) systems have recently become popular for content sharing [2, 5, 6, 11, 13], and a number of different approaches have been studied to perform effective content retrieval in such networks. Because of its desirable qualities, the P2P paradigm can be effectively extended to the challenging problem of knowledge sharing in distributed contexts [4, 7, 8, 9, 10].
This paper analyzes three content management approaches, with the aim of highlighting the differences between content management and knowledge management. Pros and cons of content retrieval policies are evaluated, also by means of simulations. The peculiarities of distributed knowledge sharing are investigated, discussing the research issues involved in combining a semantic engine with an underlying P2P overlay network. The semantic engine exploits peer ontologies for providing a semantically rich representation of peer knowledge, and matching techniques for comparing ontologies of different peers for knowledge sharing and discovery purposes. We study how the above mentioned content management policies could be adapted for supporting knowledge retrieval according to a P2P approach, with the purpose of learning concept in order to enrich the ontology locally maintained by every peer.

[*] This paper has been partially funded by “Wide-scalE, Broadband, MIddleware for Network Distributed Services (WEB-MINDS)” FIRB Project funded by the Italian Ministry of Education, University, and Research and by NoE INTEROP, IST Project n. 508011 - 6th EU Framework Programme.

2. P2P policies for content distribution and query forwarding

Several policies have been proposed to distribute contents and forward queries among peers [2, 5, 13]. Each one of these policies adopts peculiar mechanisms for content distribution and query forwarding. The policies differ in both their assumptions about the system and the impact they have on the network and applications behavior.
The three policies studied in this section are characterized by a common two-layer architecture. The content search layer takes in charge the content upload, replication and download. The overlay network layer takes in charge the routing of content queries and replies through the overlay network connecting the peers. The modules included in these layers differ for different policies.

Flooding policy. Flooding (see Figure 1) is adopted for instance by Gnutella [2]. Upon receiving a query, a peer P replies to it if the searched content is locally available. Otherwise, the query is forwarded to part or all the known peers except the one it has been received from. Query forwarding is controlled by adding time-to-live (TTL) information to the query message. This policy is stateless, as each peer only needs to know the locally maintained contents, and the list LP_p of the known peers. To discard duplicates and prevent loops in query forwarding, a peer should additionally maintain a history of the formerly processed queries. Not remembering where concepts are located makes the approach independent of content placement.

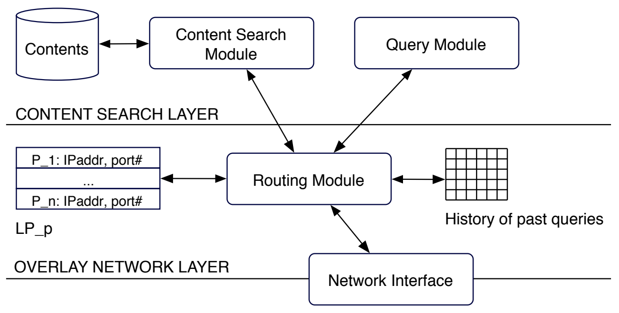


Figure 1. Functional architecture of a peer in the *flooding* policy

Deterministic placement policy. According to this policy (see Figure 2) – adopted for instance by CAN [13] – contents are placed in pre-determined peers. A *hash function* is defined, which associates a key with each content name. A distance function between two keys must be defined. Each key represents a point on a k-dimensional toroid. Each peer involved in the system is in charge of the management of the contents whose keys are in a given range; the range characterizes an area of the toroidal surface. A peer p also stores the addresses of the peers managing adjacent areas (list LP_p in the figure). The routing module forwards a query point-to-point, by exploiting a greedy approach: the query is addressed to the peer, among the adjacent ones, owning the range of keys nearest to the key of the searched content. Chord [11] adopts a similar approach. The deterministic placement policy may require that contents are placed in peers different from their originators. All peers must know the adopted hash function.

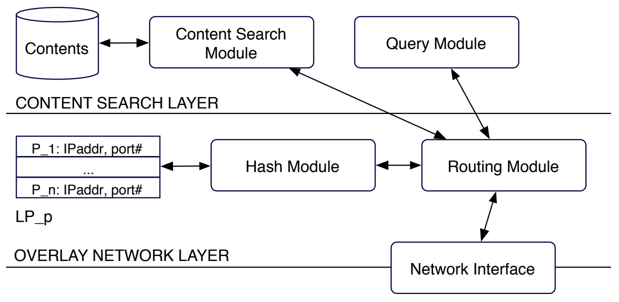


Figure 2. Functional architecture of a peer in the *deterministic* policy

Deterministic placement with learning policy. This policy (see Figure 3) is adopted by FreeNet [1, 5]. According to this policy, contents are associated with keys, as before, and a content is deterministically placed in the peer taking in charge the management of the related key. Yet, peers are also able to learn content keys and to locally replicate contents. Learning is performed as follows: when a peer generates a reply to a query, the reply is forwarded along the reverse path followed by the query, thus traversing peers that manage keys similar to the searched one. The *learning module* analyzes the received responses, and it may decide to increase the local knowledge by either recording the name of the content included in the response together with the address of the peer owning that content, or replicating the full content as well. Along its route a query could thus traverse a peer P that is not in charge of the management of the searched content, but that either owns the content because of a previous replication, or knows a peer Q owning the content. In the former case, the peer P replies to the query. In the latter case, the peer P learning module sends the peer Q address to the routing module for query forwarding, thus expediting query processing. Otherwise, the learning module hands the content key over to the hash module and the system performs greedy forwarding as in the previous case.

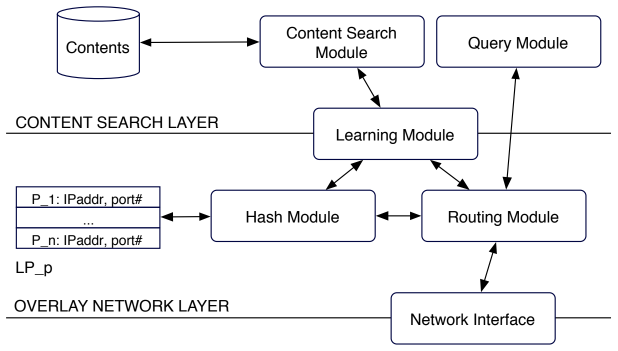


Figure 3. Functional architecture of a peer in the *deterministic-with-learning* policy

Comparison of the policies. In Table 1, we show the characteristics of the policies we described in the previous section. It partially summarizes some performance evaluations we performed through simulation techniques. Besides of the approach overheads, we consider the ability of fairly distributing the work among peers, and of adapting to the pattern of query generation – for instance by replicating highly requested documents and moving them near the requesting peers.

Flooding generates the highest amount of traffic, which is randomly forwarded. As a side effect, many query copies could be deleted along the way because of TTL expiration,

	communication overhead	memory overhead	processing overhead	load balancing	adaptability
flooding	high	low	high	low	no
placement	medium	low	medium	high	no
placement with learning	low	high	medium	high	yes

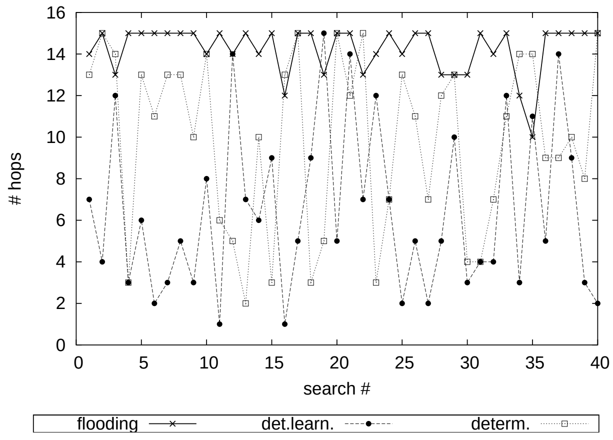
Table 1. Comparative overview of the characteristics of content distribution policies

thus wasting network resources without accomplishing useful work. Due to the stateless approach, peers are not able to remember where frequently searched concepts are located. Moreover, a content could be not retrieved although it exists in the system. Processing a query is not expensive but the processing overhead is high because of the high overall number of queries to be processed. The random distribution of contents does not allow to achieve load balancing, nor to adapt to the request pattern. When using the flooding approach to manage knowledge, a concept is not guaranteed to be ever learnt by other peers besides of the one owning it. In case semantic queries are heavy to process, running a semantic engine could be affordable only by resource-rich devices. On the other hand, several replies could be obtained for a given query, thus boosting the knowledge dissemination process.

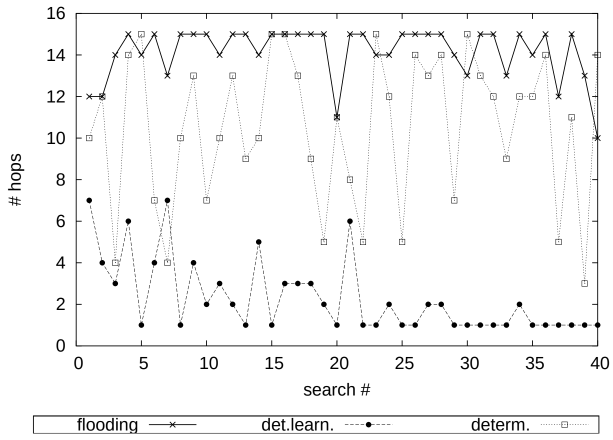
By contrast, the policies that adopt deterministic placement guarantee that an upper bound exists on the time needed to retrieve a content, if the content exists in the system. To adopt this approach for knowledge management, a placement technique should be devised that mirrors the affinity among concepts into the peers proximity in the overlay topology – so as to shorten query routes – while still guaranteeing load balancing.

The learning capability allows to adapt to network dynamics and changes in the content distribution, thus progressively yielding better performance in terms of latency to obtain a reply, number of responses collected for a given query, and number of generated messages. On the other hand, performance depends on both the amount of memory a peer uses to maintain locally replicated contents and content names, and the time needed to perform a remote copy of the content.

We have studied the policies behavior by means of simulations (see Figure 4). The reported results have been obtained considering a network composed by 1024 peers. The figure shows the mean query route length; a query stops when either the TTL expires, or the query is received at a peer owning the searched document. Each document has a unique key; each peer owns 3 documents. In the deterministic placement case, peers are mapped onto a 32×32 square grid wrapped up in a toroidal structure. In the placement with learning case, each peer can record up to 10 neighbors. Queries may concern either keys uniformly chosen from the key space (Figure $4(a)$), or popular documents (Figure $4(b)$). In the latter case, half of the queries concern the popular document, while the other queries are uniformly distributed on the remaining keys. In both cases the flood-



(a)



(b)

Figure 4. Comparison of the performance of the three studied policies for $(\mathbf{a})$ uniformly distributed queries on the key space, and $(\mathbf{b})$ hot spot searches

ing policy shows the worst performance, as queries are randomly forwarded. The deterministic placement policy guarantees that a known upper bound exists on the path length. However, it is not able to adapt to the document request pattern, thus showing a comparable behavior under both query generation policies. By contrast, the placement with learning approach can dynamically replicate keys and documents, and this characteristic allows to achieve shorter delays to find the documents. This behavior is more evident in case most of the requests concern a popular document, whose replicas are maintained by several peers.

3. Requirements for knowledge sharing in P2P systems

In contrast to content sharing in conventional P2P systems, a number of important additional requirements are to be addressed for knowledge sharing in peer-based systems. In particular, a semantically rich description of data sources to be shared is required. To this end, each peer provides a formal representation of its knowledge by means of a *peer ontology*. Furthermore, appropriate ontology matching techniques are required to identify semantic correspondences among similar concepts in different ontologies, in order to identify the neighbors of a peer based on established semantic correspondences. Finally, expressive query languages are needed for managing knowledge discovery processes.

Ontology knowledge representation. In conventional P2P systems, files to be shared are identified by means of their names. In a P2P system with knowledge sharing purposes, structured resources can be shared (e.g., structured data, semi-structured data), and a rich semantic knowledge representation has to be provided. This is accomplished through a peer ontology, thus exploiting Semantic Web techniques. Using a peer ontology, the resources shared by each peer can be represented in terms of concepts, properties, and semantic relations.

Ontology matching techniques. Conventional matching techniques for data retrieval are based on a perfect matching strategy. Having a target object to be retrieved, each node compares name and metadata specified for the target with the corresponding information about the objects it owns. In P2P systems for knowledge sharing, knowledge is represented by means of a peer ontology and semantic matching techniques are required to match target concept(s) specified in a query against peer ontology concepts. In this context, each peer is interested in comparing incoming requests against its peer ontology, in order to discover whether it can provide concepts matching the target. In order to address this requirement, appropriate matching models are required capable to cope with different levels of detail in concept descriptions. Matching techniques are exploited to perform dynamic matching at different levels of depth, with different degrees of flexibility and accuracy of results by taking into account various metadata elements (e.g., concept name, concept properties) separately or in combination. The matching process returns a ranked list of concepts semantically related to the target, in the range $[0, 1]$. A threshold t is specified to select the concepts in the ranked list with the affinity value greater than t. In a peer ontology, the meaning of ontology elements depends basically on the names chosen for their definition and on the relations they have with other elements in the ontology. In the matching process, an important requirement is related to the fact that these features can have a different impact in different ontology structures. In particular, the aim of matching techniques is to allow a dynamic choice of the kind of features to be considered in the matching process, with the goal of providing a wide spectrum of metrics suited for dealing with many different matching scenarios. A detailed description of an ontology matching algorithm we have developed for peer-based systems can be found in [3].

Query/answer representation. In P2P systems for knowledge sharing, two query models are defined to distinguish between two different kind of searches: the probe query model and the search query model.
Probe query model. The probe query model is used by a peer interested in extending its knowledge acquiring knowledge related to target concept(s) from other peers. Each peer having concepts matching the target concept(s) of a probe query can answer to the requesting peer. The answer to a probe query is constituted by metadata about concepts having semantic affinity with the target. A requesting peer can use the answer to a probe query to extend its knowledge on target concept(s) with new properties or relations.
Search query model. The search query model is used by a peer in order to find in the network data related to one or more concepts of interest. Each peer storing data matching the target concept(s) of a search query can answer to the requesting peer.

Probe queries, conceived for allowing knowledge sharing between nodes, represent a real add-on with respect to conventional search query models. For knowledge sharing, we require an expressive probe query representation to support the specification of concepts, properties, and semantic relations, as well as the specification of the matching model and the matching threshold required for the query processing. A reference template for probe queries is shown in Figure 5. It is composed of the following clauses:

- *Find:* list of target concept(s) names.
- *With:* (optional) list of properties of the target concept(s).

- *Where:* (optional) list of conditions to be verified by the property values, and/or (optional) list of concepts related to the target by a semantic relation.
- *Matching model:* (optional) specification of the matching model asked by the requesting peer to process the query.
- *Matching threshold:* (optional) specification of the threshold value t, with $t \in (0, 1]$ to be used for the selection of matching concepts based on the semantic affinity value determined by the matching process. If a matching threshold is not specified in the query, the answering peer adopts its own default threshold.

Query template	
Find	Target concept name [, ...]
[*With*	⟨Property name⟩ [, ...]]
[*Where*	Condition, ⟨related concept, semantic relation name⟩ [, ...]]
[*Matching model*	Matching model to be used]
[*Matching threshold*	$t \in (0, 1]$]

Figure 5. Probe query template

When a peer receives a query from another peer, the query is processed in order to extract the target concept(s) and the matching model to use. In particular, the query is transformed into an ontological description of the target concept(s) (using information in the Find, With, and Where clauses) for comparison against the peer ontology.
Once concepts matching a target concept have been identified using the matching algorithm, they are returned to the requesting peer through a query answer, which contains the list of concepts matching the target. Each concept is described by the following answer template (see Figure 6):

- *Concept:* name of the matching concept.
- *Properties:* (optional) list of properties of the matching concept.
- *Adjacents:* (optional) list of concepts related to the matching concept by a semantic relation.
- *Matching:* set of pairs ⟨target concept, affinity value⟩, specifying the target concept with which the matching concept matches, together with the corresponding affinity value.
- *Location relations:* (optional) list of location relations known by the answering peer for a matching concept.
- *Matching model:* specification of the matching model applied to process the query.
- *Matching threshold:* (optional) specification of the threshold value t, with $t \in (0, 1]$ used for the selection of matching concepts based on the semantic affinity value determined by the matching process.

Answer template	
{*Concept*	Concept name
[*Properties*	⟨Property name⟩ [, ...]]
[*Adjacents*	⟨related concept, semantic relation name⟩ [, ...]]
Matching	⟨Target concept, affinity value⟩ [, ...]
[*Location Relations*	Location relation [, ...]]
Matching model	Matching model name
[*Matching threshold*	$t \in (0, 1]$]}

Figure 6. Answer template

4. A semantic engine architecture for knowledge sharing

According to the requirements illustrated in Section 3, we present a reference semantic engine architecture to support knowledge sharing in P2P systems. Such an architecture provides a semantic infrastructure which enables each peer of a P2P network to evolve the *content search layer* of conventional P2P architectures to a *knowledge search layer* enriched with semantic functionalities. As shown in Figure 7, the semantic engine is composed of the following components:

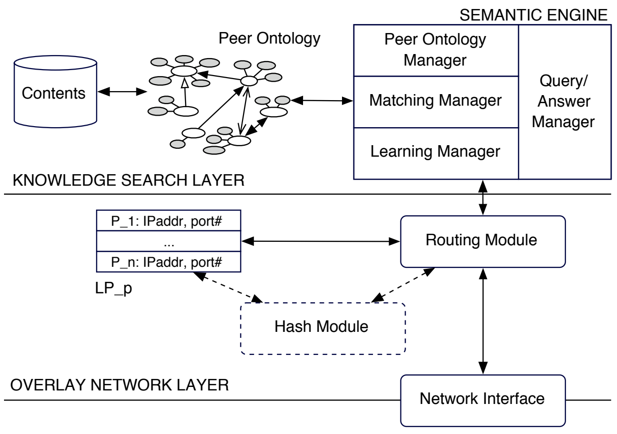


Figure 7. The semantic engine architecture

- **Peer ontology manager.** The peer ontology manager is responsible for maintaining the peer ontology which is organized into a *peer knowledge layer* and a *network knowledge layer*, respectively. The peer knowledge layer describes the knowledge a peer brings to the network. The network knowledge layer describes the knowledge that a generic peer P has of the knowledge owned by other peers in the network. When a peer P receives a content concept from another peer P1 in response to a probe query, the peer ontology manager stores in the network knowledge layer a description of the peer P1. Peer descriptions are given in form of *network concepts*, characterized by a set of properties describing the network features of P1 (e.g., IP-address, bandwidth). A *location relation* connects a content concept c with a network concept c' describing a node containing resource concepts matching c (i.e., concepts semantically related to c). A generic peer P can increase its knowledge by adding/extending content concepts and/or by adding new location relations and network concepts.

- **Matching manager.** The matching manager is responsible for performing semantic matching between a target concept in a query and concepts in the peer ontology. With respect to the content search engine of conventional P2P architectures, the matching manager provides a semantic improvement in concept comparison, by considering both the linguistic and the contextual features of ontology concepts in peer ontologies. A detailed description of the ontology matching algorithm we have developed for the matching manager is presented in [3, 4]
- **Query/answer manager.** The query/answer manager is responsible for query composition, processing, and answering. With respect to conventional P2P architectures, the functionalities of this component are provided by the query3 module. We have integrated the query/answer manager in the semantic engine because of the strong interaction with the matching manager, as described in Section 3.
- **Learning manager.** The learning manager is responsible for the acquisition of information about the placement of contents in the network in order to increase query distribution effectiveness. In contrast to conventional P2P architectures, the learning manager has been integrated in the knowledge search space because information about content distribution are maintained and accessed through peer ontologies.

The semantic engine enables peers to support knowledge sharing requirements, and constitutes a flexible infrastructure capable to cope with any P2P architecture implemented in the overlay network layer [1]. Nevertheless, the presence of semantic functionalities based on peer ontologies and matching techniques poses additional issues which are to be addressed when considering the interaction between the semantic engine and the Overlay Network. With respect to the P2P architectures presented in Section 2, we discuss how the Overlay Network components can be properly adapted to cooperate with the semantic engine components to support knowledge sharing functionalities.

1 In Figure 7, a comprehensive representation of the three overlay network architectures presented in Section 2 are summarized. The hash module is represented through broken lines because only the deterministic placement and deterministic placement with learning policies require this module.

Flooding with learning. The query distribution protocol adopted in the flooding policy, is wasteful due to the high generated traffic and the processing overhead. In order to improve the performance of such an approach, the semantic engine can be adopted to support a semantics-based query distribution protocol. Each peer exploits its peer ontology in order to maintain information about semantically related concepts stored in other peers by means of location relations. For instance, a location relation connecting a content concept CC with a network concept NC, is used to represent that the peer described through NC can provide concepts semantically related to CC. When a peer has to forward a probe query containing a target concept TC, the peer ontology is exploited by semantic matching techniques: (i) to discover concepts semantically related to TC, and (ii) to identify location relations connecting such concepts to a set of network concepts describing the peers to which the query is to be sent, by avoiding flooding. A learning manager has to manage and update the location relations referring to other peers knowledge.

Semantic placement. As the deterministic content placement, the semantic knowledge placement combines a predefined network structure with a hash function in order to implement an efficient query distribution protocol. Concepts are dynamically distributed in the network following similarity criteria, so that each peer maintains in its peer ontology concepts with a high affinity among them. When a target concept TC is to be placed in the network, each peer evaluates the semantic affinity of TC with the concepts actually stored in its peer ontology, by exploiting semantic matching techniques in order to decide if TC is to be inserted in the peer ontology or if it is to be forwarded for a better placement. As described in Section 2, simulation results show that deterministic knowledge placement can improve query distribution efficiency due to hashing. When semantic criteria are followed for knowledge placement, hashing improvements are possible only if the hash function adopted preserves the semantics of the affinity evaluation.

Semantic placement with learning. Combining learning techniques and semantic knowledge placement, an adaptive P2P architecture can be defined. As described in semantic placement, knowledge is distributed in the network according to similarity criteria. When a target concept TC is to be placed in the network, each peer applies semantic matching techniques in order to evaluate the semantic affinity between TC and its peer ontology concepts. According to affinity evaluation results, more than one peer can decide to maintain TC in its peer ontology. As in conventional adaptive networks, concepts have not a static placement and move in the network according to requests. A learning strategy, implemented through location relations as in the flooding with learning policy, can be adopted to maintain information about other peer contents enforcing the effectiveness of query distribution. An ontology merging policy is required in order to preserve the correctness of peer ontology descriptions when new concepts are to be dynamically inserted.

Considerations. Evolving content sharing to knowledge sharing networks is a challenging issue which is being addressed by other research projects providing different solutions [7, 8, 9, 10, 12]. With respect to these approaches, the main contribution of the semantic engine architecture we propose is related to the presence of an ontology matching manager which allows dynamic knowledge discovery and sharing without a centralized authority (e.g., SuperPeer) and/or integrated knowledge descriptions. Moreover, the semantic engine architecture is suitable for knowledge sharing in dynamic scenarios and can be properly adapted to be placed on different P2P infrastructures.

5. Concluding remarks and future work

Several issues have to be investigated for the development of an effective knowledge sharing P2P system based on a semantic engine. Porting policies from a content search context to a knowledge search context is not such immediate. Differences lie in the concept management. For policies involving a dynamic replication mechanism, the pace of the ontology growth must be analyzed, as well as possible effects for query processing depending on the limitation of the peer ontology size. Concepts dynamically placed in a peer ontology may yield an overlay (virtual) topology more connected than in case of content management, thus possibly generating more traffic for query forwarding. We are working in the direction to evaluate the impact of different learning strategies on query processing in order to identify for each peer the correct trade off between concepts to include in the peer ontology and concepts to distribute in the network.

Performance indexes to be considered for a knowledge search context differ from those usually considered for content search. Knowledge management, namely, concept matching and concept insertion into a peer ontology, can be expensive in terms of processing. Their costs depend on the policy chosen for concept replication in different peers and for query forwarding, which can be controlled by adopting policies that implement deterministic placement. In these cases, policies could be devised to distribute query processing among several peers each one of which carries on the search for a subset of related concepts, thus avoiding work duplication. To guarantee an effective load balancing, concept keys assignment should place only a few highly generic concepts or frequently requested concepts on a certain peer so as to avoid that it becomes a bottleneck.

According to the considerations presented in Section 4, our future research work will be devoted to evaluate the performance of different P2P architectures for knowledge sharing based on the proposed semantic engine.

References

[1] The Freenet website. http://freenet.sourceforge.net.

[2] The Gnutella website. http://www.gnutella.com.

[3] S. Castano, A. Ferrara, and S. Montanelli. H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In *Proc. of the 1st SWDB VLDB Workshop*, Berlin, Germany, 2003.

[4] S. Castano, A. Ferrara, S. Montanelli, and G. Racca. Matching Techniques for Resource Discovery in Distributed Systems Using Heterogeneous Ontology Descriptions. In *Proc. of ITCC 2004*, Las Vegas, Nevada, USA, 2004.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Lecture Notes in Computer Science*, 2009:46, 2001.

[6] A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In *Proc. of ICDCS 2002*, Vienna, Austria, 2002.

[7] J. B. et al. A Metadata Model for Semantics-Based Peer-to-Peer Systems. In *Proc. of the 1st SemPGRID WWW Workshop*, Budapest, Hungary, 2003.

[8] W. N. et al. EDUTELLA: a P2P Networking Infrastructure Based on RDF. In *Proc. of WWW 2002*, Honolulu, Hawaii, USA, 2002.

[9] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *Proc. of ICDE 2003*, Bangalore, India, 2003.

[10] R. Huebsch, J. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Proc. of VLDB 2003*, Berlin, Germany, 2003.

[11] I. Stoica et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, February 2003.

[12] A. Kementsietsidis, M. Arenas, and R. J Miller. Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In *Proc. of ACM SIGMOD Int. Conference on Management of Data*, San Diego, California, USA, June 2003.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of the ACM SIGCOMM 2001*, San Diego, CA, USA, 2001.