UNIVERSITÀ DEGLI STUDI DI MILANO
Dipartimento di Informatica e Comunicazione

RAPPORTO INTERNO Nº RT 15-06

ORION – Ontology-based Routing of querIes in Overlay Networks

Elena Pagani, Gian Paolo Rossi,
Enrico Pertoso

ORION – Ontology-based Routing of querIes in Overlay Networks

Elena Pagani, Gian Paolo Rossi, Enrico Pertoso
Information Science and Communication Department
Università degli Studi di Milano, Italy
E-mail: {pagani,rossi}@dico.unimi.it;enrico.pertoso@gmail.com

TECHNICAL REPORT RT15-06

1 Introduction

Modern search engines are moving apart from current keyword-based mechanisms to retrieve information and are embracing more powerful mechanisms that are capable of reasoning on semantic attributes of contents in a distributed repository. On one side, this is motivated by the growing needs to enable machine-based services of information retrieval. On the other side, the huge of available information is demanding for shared data models or common conceptualizations, also referred to as ontology, to achieve the integration of semantically homogeneous data coming from Internet communities and separate sub-fields. This evolution is directly affecting the components of modern content retrieval and searching systems and is driving the innovation of their underlying communication infrastructure, namely the Web and peer-to-peer platforms, P2P. So far, most efforts have been addressed inside the W3C to define mark-up languages capable to capture some content meaning and code it to obtain a machine understandable description. A mark-up language enables defining a domain ontology and the semantic browsing through this ontology can be performed without substantial modification of the web by simply interposing a middleware between the browser and the network layers. Purpose of the middleware is the extraction, from the search keywords issued by the user, of a set of affine concepts that are then sent as input to a traditional search engine. Similarly, responses are filtered by the middleware using the ontology before to be displayed to the user. This approach is possible because the web has maintained clean separation between search engine and communication platform. By contrast, in traditional peer-to-peer systems, the content sharing is achieved through an interwoven relation between the search policy and the communication overlay. As the consequence, the adaptation of P2P systems to provide semantic search requires radical changes and poses several challenging

problems to the research. Among them, one of the most critical and, so far, less explored problem is the design of the communication infrastructure capable to support the maintenance of and the retrieving in a distributed ontology, while continuing to ensure efficient and scalable operations. It is worth to notice that the centralized approach, i.e. to adopt centralized ontology and search engine, is a dead-end approach from this point of view. In common P2P application scenarios, in fact, each node, or peer, maintains local capabilities of reasoning onto the local ontology and is able to access the entire distributed knowledge, obtained as the aggregation of the set of peers, by exploiting the features of the communication infrastructure. It is clear that such a distributed approach has the twofold design advantage of limiting the node processing load and granting the capability of reasoning on the entire distributed ontology (completeness) in a scalable and efficient way. To achieve this result, however, local search engines need the aid of a distributed query processing whereby the concepts and their semantic relations are stored and retrieved from hosting nodes within the network and queries are routed over a connectivity infrastructure that mirrors the semantic relations among the searched objects. From the architectural point of view, the above arguments lead to the design of an overlay network, i.e. the middleware infrastructure on top of the TCP/IP protocol stack, that efficiently support the required diffusion of queries to retrieve distributed concepts and related content. Because the overlay network provides a connectivity that should derive from the semantic relations among concept of the ontology, we use the term of ontology-based routing and addressing to identify the basic functionalities it provides.

The performance and the functionality of the desired query processing system is significantly influenced by the node to node communication mechanisms provided by the overlay network. Despite its critical role in the entire architecture, the overlay network has so far received a limited attention from the research community and the available proposals mainly address this issue by adopting variation of flooding [14] or placement mechanisms [10], that will be briefly described in the next section. The main contribution of this paper is the co-design of a query processing and of an overlay network to enable efficient and scalable self-organized content retrieval in a complete way. To achieve this goal, the paper adopts the concept of Semantic Overlay Network (or SON) which has been firstly introduced by [6]. SONs are here used to aggregate sets of semantically homogeneous nodes in a community, i.e. nodes owning concepts bound by some degree of affinity. The overlay is a hierarchical infrastructure of SONs whose topology is self-organized and where all nodes have autonomous capability of joining, leaving and navigating SONs in the overlay through totally distributed operations. The adopted architecture represents the trade-off between completeness and efficiency: queries are routed over an overlay describing only hierarchical affinity while nodes can exploit the full power of local semantic engines to ensure completeness in solving the queries.

The paper describes the ORION (Ontology-based Routing of querIes in Overlay Networks) system for the construction of the overlay infrastructure and the routing

of queries through its topology. In this section, related works in the literature are reviewed in order to motivate the design choices of ORION. In sec.2, the assumptions on the ontology structure and the properties of the engine for semantic reasoning are described, and the functional architecture of peers is presented. In sec.3, the principles for the characterization of a hierarchical topology of the overlay are discussed, and the protocols for hierarchy formation and for query processing are described in detail. Sec.4 concludes the work.

1.1 Related works

In the literature, there are a lot of peer-to-peer systems proposed for content sharing. One of the first attempts was to use a **centralized** system, such as Napster [8]. Besides of obvious drawbacks – such as, the central server is a bottleneck and a single point of failure – this approach is unsuitable for semantic-based knowledge sharing because it implicitly assumes that the central server is able to reason with every semantic formalism possibly used to represent knowledge in the peers.

In [3], content sharing policies are divided into **flooding**-based policies (e.g. [14]) and policies adopting **deterministic placement** of contents (e.g. [10]). In the former case, peers do not know anything about content location, the overlay has a flat topology, and search is performed by flooding queries, with very high cost in large systems. If flooding is bounded with the aim of limiting costs, then the probability of retrieving some responses – not to mention that of finding *all* existing responses – is decreased. In the latter case, hashing is used to establish a relationship between a content and its location: a content is placed in the peer whose identifier results from hashing the content's key. Deterministic placement has the drawback of forcing peers to maintain information for contents not locally originated, which can be not desirable or viable. As for the centralized approach, if adopted for knowledge sharing, a node could not have the semantic tools to reason about knowledge represented with a formalism different from that locally adopted. Moreover, in order to achieve good performance, the hashing function is required not only to guarantee fair load distribution among the peers, but also to place semantically close concepts into geographically close peers. This goal is achieved by pSearch [13], which merges the CAN [10] approach with semantics. The document's semantic is generated by *latent semantic indexing*. With this formalism, a document is represented by a vector containing, for each term in a reference vocabulary, the degree of importance of that concept in the document. A document thus represents a vector in a multi-dimensional space, and is mapped into the peer responsible for managing the region of space including that point, using a hash function. Semantically affine documents are close in the space, and are mapped on the same or neighbor nodes. Relationships among documents may be derived also from co-occurrence of terms.

A **hybrid** approach, adopted for instance by FreeNet [5], consists in dynamically replicating either a content's location or the content itself near the peers that most

frequently access it; if no information is available in a peer's neighborhood, flooding is used. In the case of knowledge sharing, besides of the flaws related with deterministic placement, caching also implies dynamically replicating ontologies, which can be both bandwidth and memory consuming. It is worth to notice that in many real peer-to-peer systems the high costs discourage users from participating for times longer than the minimum needed to download the data they are interested in; this results in poor content sharing and thus in very low rate of success in retrieving information.

The idea of using *Semantic Overlay Networks* (SONs) was first proposed in [6], to implement an **indexing** approach. That work focused on policies for characterization of SONs and their hierarchy, on policies the peers may adopt to choose what SONs to join to, and on policies to characterize the most appropriate SONs to which a query should be forwarded. The hierarchy of SONs is characterized *a priori*; it must then include all concepts that may eventually appear at peers, as well as concepts that may belong to peers not currently included in the system but that can join in the future. Communication aspects are not considered. When a peer enters the system, it floods a request to obtain the hierarchy description, basing on which it decides to what SONs it should belong according to the locally owned contents. A query should be forwarded to just the nodes belonging to the appropriate SON(s), without bothering other peers. For both joining a SON and forwarding a query to the appropriate SON, the lookup of nodes belonging to that SON is performed via flooding. A membership should be maintained inside SONs, although [6] does not discuss this issue. Nor it is clear whether "links" (i.e. mutual knowledge among peers) are maintained between father/child pairs of SONs.

Edutella [20] uses the JXTA platform for communications, and the RDF [18] language as semantic tool, to implement a semantic network for educational purposes. Peers register with a query service the query schemes they are able to reply. A query is sent to the peers registered as able to reply. As far as the communication aspects are concerned, Edutella adopts a super-peer approach with super-peers connected in a hierarchical hypercube structure, which is efficient for message broadcasting and more resilient to failures than tree structures. Super-peers maintain routing indexes containing metadata information, and the identifiers of peers using those metadata schemes. The drawback of the hypercube topology is that, if no enough peers are available to build a hypercube, then some peers must occupy several vertices in the infrastructure. Each super-peer is connected to a group of peers, and knows the schemes they use to reply to queries; each super-peer is responsible to forward a query to the peers it controls that are able to process the metadata in the query. Routing of queries among super-peers is either a broadcast, or routing indexes may be maintained also at this level, which are summaries of the super-peer indexes. Peers can connect either to arbitrary super-peers or to a super-peer controlling similar peers; in the latter case, several similarity notions can be used. In both cases, if a peer join provokes an update of its super-peer index, the super-peer broadcasts the update to the other super-peers.

In [15], SONs are used to ease content search, and SON membership is maintained with a lightweight policy. Each peer p monitors what other peers have been more useful in replying to its recent queries on a certain concept c. Those peers are considered as belonging to the same SON as p as far as c is concerned. The SON membership is implicit: if a peer p considers a peer q member of its same SON for concept c, the reverse does not necessarily hold. This mechanism has been refined in [16], where peers use gossiping to exchange information about their interests, in order to supply each peer with knowledge about *every other peer*. A peer uses this information to maintain a *semantic list* of affine peers. Interests are represented by file names; no ontological reasoning is used. Queries are routed to peers in the semantic list of the querier. However, the work assumes that a peer generates queries only about contents/concepts affine to those it already owns. Otherwise, flooding is used.

The Semantic Web initiative [19] aims at facilitating knowledge sharing through the deployment of tools able to derive information from data, to integrate data, and to automatically reason about this information, using semantic theory and ontologies. The RDF [18] and OWL [17] languages are currently recommended to represent knowledge in the semantic web. Communication aspects involved with navigating and querying the semantic web have yet to be dealt with. A possible solution lies in the use of software agents: a retrieval engine discovers all contents containing the searched terms; agents then reason about those documents to refine the results [12].

In contrast with the previous solutions, ORION adopts a distributed approach. Every peer only maintains knowledge extracted from local contents, and reasons with the semantic formalism locally adopted; no deterministic placement or caching is used. Peers belong to as many SONs as many their "areas of interest" are. SONs are connected in a hierarchical overlay, such that the parent/children relationships mirror a hierarchical relation between the concepts represented by the SONs. The hierarchy is *not* pre-determined, but rather its structure depends on the concepts held by the peers currently belonging to the system, and may dynamically vary according to changes in the peer knowledge. Different subsets of peers may use different semantic formalisms. As a consequence, the overlay structure is more likely a forest than a tree. Peers do not know the global hierarchy structure; they know the membership of the SONs they belong to, and have partial knowledge of the tree everyone of such SONs is linked to, in terms of parent and children SONs.

2 System model

2.1 Assumptions on ontology structure

In this work, the focus is on the problem of distributed knowledge maintenance and retrieval. Knowledge is represented through ontologies. Informally, an ontology is a set of concepts linked by semantic relations. It can be represented as a graph, whose

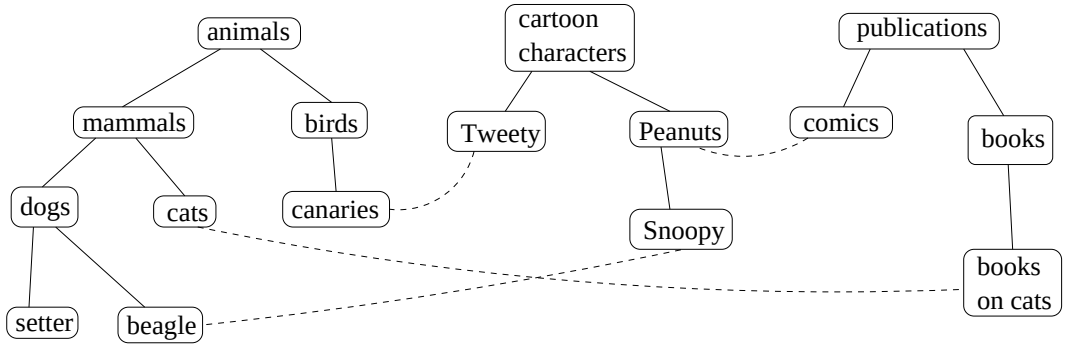


Figure 1: Example of ontology

nodes are concepts while links are affinity relationships between pairs of concepts. Affinity relationships can be characterized in several ways. The Semantic Web initiative recommends the RDF and OWL languages to build ontologies. In RDF [18], metadata used to represent data properties are in the form of a triple $A(O, V)$ meaning that the object O has an attribute A with value V. Predefined types of objects allow to define a *hierarchy* of classes of objects, and of classes of properties. *OWL* is a language for ontology definition and management based on description logic; it is an evolution of the DAML and DAML+OIL [7] languages. All these languages are based on description logics, thus allowing richer expressive power and supporting automatic reasoning. In [7], an ontology is defined as *"a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept."* DAML+OIL has – amongst many others – predefined axioms to determine whether a concept is subclass (more specific) than another (`subClassOf`), or whether two concepts are equivalent (`sameClassAs`). In fact, these are fundamental axioms, as all the others can be reduced to them. *WordNet* [9] is an English lexicon developed at the Princeton University, composed by five syntactic categories: nouns, verbs, adjectives, adverbs, and function words. *Opencyc* [11] is a large knowledge base whose terms are axiomatized by assertions expressed with predicate calculus. It includes several algorithms and theorem provers to derive inferences from the ontology. Opencyc includes predicates for term mapping that allow to establish whether two terms are synonyms (`synonymousExternalConcept`), or whether one of the two is a subclass of the other (`isa`, `genls`). Both parts of WordNet and ontologies expressed in DAML have been merged with OpenCyc. In *Helios* [4], four degrees of affinity can be detected between two concepts, through different techniques of *matching* among concepts. Matching can be based – in order of increasing semantic richness – (*i*) on *names*, that is, two concepts are related if their names are synonyms; (*ii*) on *attributes*, that is, two concepts are related if they are described by comparable structures; (*iii*) on *relationships* that consider two concepts as affine if they either are used in the same context or are related with the same concepts; and finally (*iv*) on *instances*, that is, two concepts are affine if they have instances in common.

In the simulations discussed in sec.**??**, WordNet is adopted as an example ontology.

In this work, a concept is represented by a name and a list of attributes describing the concept structure. Since the proposed system could be adopted to retrieve contents using semantic reasoning, the description of a concept in the ontology may also include a link to a content locally held; this point will be made clear in the next subsection. An ontology is a directed weighted graph whose edges between two affine concepts are represented by couples $\langle$*type of edge, affinity degree*$\rangle$. The former field indicates the kind of affinity relationship between the two concepts; the latter field has value between 0 and 1. A *semantic engine* is a set of tools able to extract concepts from contents, and to supply affinity relationships amongst concepts through reasoning with a certain semantic formalism. *Full concept matching* indicates the operations the semantic engine performs to derive all possible relationships between two concepts. A *cluster* is a part of ontology including a set of related concepts and all their semantic relationships. A cluster for a concept c includes all concepts related with c and their relationships, possibly within a certain degree of affinity. Hence, a cluster is an ontology. We assume that the affinity relations satisfy a *transitivity property*, such that if a concept A is affine to a concept B with weight w_{AB}, and B is affine to a concept C with weight w_{BC}, then A and C are affine with a weight $w_{AB} \cdot w_{BC}$.

We do not make any assumption about how concept affinity is determined. For the purposes of ORION, we assume that, whatever are the principles on which the semantic engine is based, it is able to derive hierarchical relations between concepts according to a notion of *semantic complexity*. These relations are a subset of the affinity relations the semantic engine can find out through full concept matching. Let α be the affinity notion that characterizes hierarchical relations between pairs of concepts; the *dist* function is defined as a measure of the minimum number of edges composing a path of type α connecting two concepts A and B, and of the hierarchic relationship between the two concepts, as follows:

- $dist(A, B) = 0$. In this case, the two concepts are comparable and equivalent ($A \equiv B$). This can be the case of concepts that are synonyms if comparison is performed with linguistic criteria, or that have the same structure and are then recognized as representing the same object if comparison is performed with structural criteria;

- $dist(A, B) < 0$. In this case, concept A is semantically less complex than B ($A \prec B$). For instance, A could be a hyperonym of B according to a linguistic comparison, a more generic term according to reasoning with description logics, a super-class in a RDF tree or a category of higher level than B according to a hierarchy such as those considered by Yahoo! or Google.

- $dist(A, B) > 0$. In this case, B is semantically less complex than A ($A \succ B$).

- $dist(A, B) = \perp$. In this case, the two concepts are not comparable, that is, no relation between the two concepts exists according to the α affinity.[1] Anyway, other semantic relations may exist between A and B.

In the first case $\alpha(A, B) = 100\%$; in the second and third cases $0 < \alpha(A, B) < 100\%$; while in the last case $\alpha(A, B) = 0$. As a consequence, the α affinity characterizes hierarchies of concepts, which can be represented as a *forest*, where concepts that are not comparable belong to different trees. As an example, let us consider the ontology shown in fig.1: edges in solid lines are those produced by the α affinity, while dashed edges are obtained from other semantic relationships. Hence: "`books`" is a concept more complex than "`publications`", while "`Tweety`" and "`dogs`" are not comparable either with α or with other semantic relations; $dist($"`setter`", "`animals`"$) = 3$, $dist($"`animals`", "`canaries`"$) = -2$, and $dist($"`birds`", "`comics`"$) = \perp$.

The hierarchy of SONs is obtained applying the *dist* matching function to the knowledge held by peers. It is worth to notice that using the *dist* function alone for building the overlay topology does *not* make the ontologies held by peers poorer. Rather: for the sake of building the overlay topology, ORION considers only a subset of the relations among concepts, so that the graph among concepts can be pruned to a forest. But each peer maintains the whole – richest – graph that can be built with the concepts it knows using all the affinity relations the semantic engine is able to find.

2.2 Architecture of peers

The functional architecture of a peer participating in ORION is composed of two plans, each involving two logical layers (fig.2). The *data plan* is composed by the *content layer* and the *ontology layer*. In the former, the databases and content repositories are maintained, that is, the collection of all documents known by the peer. The latter includes the ontology known by the peer. The *control plan* provides the mechanisms to maintain both contents and ontology, and to build the overlay and provide the communication primitives over it. At the *communication layer*, the suite of ORION protocols is executed, which performs overlay maintenance, and message transmission for both knowledge and content retrieval from other peers in the system. At the *semantic layer*, a semantic engine operates, able to perform semantic reasoning on both the contents locally held and contents and knowledge learnt from other peers along the system lifetime. The query manager takes in charge the processing of messages received from the *communication layer*, to supply it with indications about further message forwarding. The ontology manager maintains the local ontology according to the results obtained by the semantic engine in processing concepts and contents locally owned.

[1] This may also be the case of concepts described with different formalisms, such that the semantic engine is not able to find any relation among them.

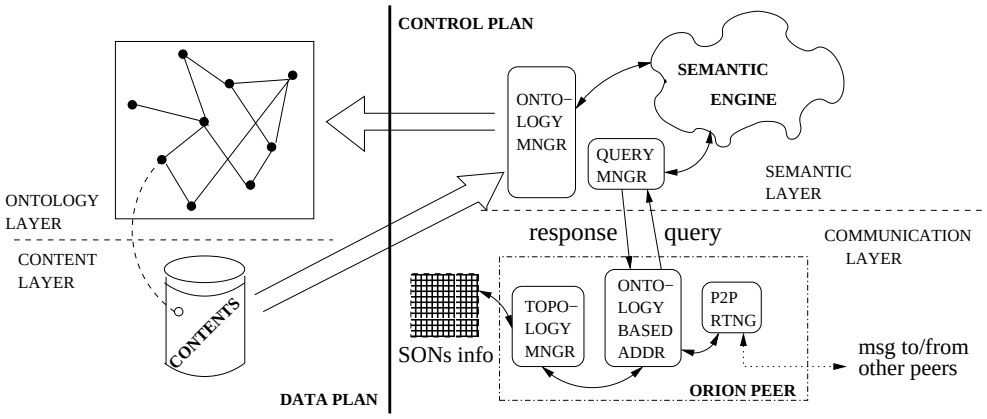


Figure 2: Functional architecture of peers

In this work, the focus is on the *communication layer*, provided that the semantic engine satisfies the assumptions discussed in sec.2.1. The *communication layer* involves a P2P routing module that takes in charge the forwarding of messages using the overlay links among peers, and interacts with underlying network protocols. The topology manager maintains the information about the overlay topology, that is, the identities of adjacent peers. The ontology-based addressing exploits the semantic engine to discover whether a query is arrived at the destination, that is, a response can be generated locally, or it must be further forwarded and to what neighbors.

As a preliminary operation at a peer bootstrap, the ontology manager uses the semantic engine to extract concepts from the local contents and to produce the local ontology. Concepts are linked to the local contents concerning them. The semantic engine is used for two tasks:

1. maintenance of the local ontology: this is performed every time a concept is either added to or removed from the local ontology. The former event happens when new contents – describing concepts not included so far in the peer's ontology – appear at the *content layer*, or when the peer learns new knowledge from other peers. The latter event may happen when all contents related with a certain concept are removed from the peer. Yet, a peer could also choose to not loose knowledge also if no contents related with it are maintained locally. Upon one of those events, the ontology manager uses the semantic engine to adapt the ontology graph;

2. cooperation with the *communication layer*: for overlay adaptation upon join/leave requests from peers, and for query processing.

As far as the latter point is concerned: each message received by a peer is handed over by the P2P routing to the ontology-based addressing, which in turn exploits the semantic engine through interaction with the query manager. The reply of the query

manager is used by the ontology-based addressing to decide the next action. In case of join/leave events of peers, the ontology-based addressing interacts with the topology manager to maintain up-to-date neighborhood information: the position in which a peer is connected to the overlay topology depends on the knowledge it owns. In case of queries – to retrieve either knowledge or contents – the query manager tells the ontology-based addressing whether a reply can be generated locally and how it is, or whether to perform further routing. In the latter case, the response of the query manager is interpreted by the ontology-based addressing module – also using the information of the topology manager – to supply the P2P routing with the address of the appropriate next hop peer. In this respect, the *semantic layer* operates as a sort of routing protocol.

Through the synergy of the *semantic* and the *communication layers*, ORION organizes the overlay topology in order to route queries – for both contents and concepts – right to the peers able to satisfy it. This achieves a twofold goal: (*i*) bandwidth is saved, by avoiding the routing of queries to all peers; (*ii*) load is reduced, as query processing is performed only at peers having high probability of being able to reply. Indeed, the processing required for query routing is more lightweight than full concept matching. We describe in the next section how this is accomplished.

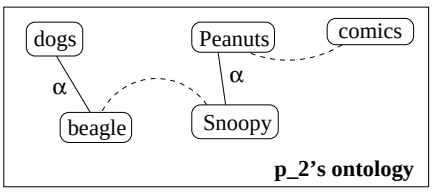
3 ORION

In this section, an overview of the hierarchical structure of the overlay topology is provided. The protocol for overlay construction and the protocol for query routing and processing are then described.

3.1 Overview of the hierarchical structure

The overlay network has a forest topology, with root nodes of trees connected to one another. Nodes of the trees are SONs [6]. A SON root of a tree is denoted as *first level* SON. A SON groups peers owning affine concepts, within a certain degree of affinity. A SON is characterized by a *manifesto*, which is an ontology – possibly composed of a unique concept – representing the "interests" the peers in the SON have in common. A peer may belong to several SONs, as many as manyfold are its interests. Each SON $\mathcal{S}$ has a *root peer* $rp_{\mathcal{S}}$, which usually is the SON's founder. SONs are linked together depending on the result of the *dist* function applied to their manifestos. For instance, in the example ontology shown in fig.1, three trees could be characterized by neglecting the dashed links among concepts. The SONs members of the forest are *not* predetermined. The forest is built up dynamically from the concepts held by peers; a SON exists only if at least one peer owns the concept(s) represented by its manifesto. Changes in the knowledge held by peers may raise changes in either the membership of peers to SONs, or to the forest topology. The former case happens when a peer either acquires new concepts not related with any other concept known before, or deletes all

p_1 : {animals, dogs, Tweety, comics}
p_2 : {beagle, dogs, comics, Peanuts, Snoopy}
p_3 : {mammals, cats, books on cats, publications}
p_4 : {birds, canaries, books, Tweety}
p_5 : {setter, cartoon characters, Tweety, Snoopy}



SON	membership
animals	p_1
mammals	p_3
birds	p_4
dogs	p_1, p_2
cats	p_3
canaries	p_4
beagle	p_2
setter	p_5

SON	membership
cartoon characters	p_5
Tweety	p_1, p_4, p_5
Peanuts	p_2
Snoopy	p_2, p_5
publications	p_3
books	p_4
books on cats	p_3
comics	p_1, p_2

Figure 3: Example of system and overlay topology

concepts that determine its membership in a certain SON. The latter case happens when either a new SON must be created or a SON becomes empty of peers, as a consequence of changes in peers' ontologies. It is worth to notice that, although the overlay network is built using a subset of the semantic relationships among concepts,[2] each peer maintains a complete ontology. For instance, the ontology of peer p_2 in fig.3 maintains edges richer than those produced by the hierarchical relation α alone.

SONs are exploited to ease information retrieval and to reduce the overhead involved with query processing. Let us consider the peers in fig.3: peer p_1 belongs to four different SONs, while SON "`Tweety`" includes three peers. The overlay produced by ORION is that represented in fig.1 with solid lines, and with additional links connecting the first level SONs. If a query is generated by peer p_3 about the concept "`books`" and related contents, this query does not need to be processed by all peers in the system. It is forwarded through the overlay till it reaches the appropriate SON. There, the query is processed by every peer in the SON, using the whole ontology each peer owns. Navigation in the overlay is performed following the links between peers that are semantically significant with respect to the query. In the example, p_3 belongs to the SON "`publication`"; by comparing the manifesto of the SON with the searched concept, p_3 discovers that the query must be forwarded to the downstream SON "`books`", where the query is processed by p_4.

Peers do not know the overall overlay topology. Each peer only knows to what SONs it belongs, and the identity of the root peers of these SONs. Root peers additionally know the root peers of the upstream and downstream SONs, and the membership of their SONs. This is a minimal requirement, adopted when ORION operates *stateless*. A *stateful* mode of operation is also possible, with root peers knowing the manifestos of upstream and downstream SONs. The *stateful* mode is more expensive in terms of memory usage at peers, but helps speeding up query processing. A root peer of a *first level* SON knows the root peers of the other first level SONs. In stateful mode, it also knows the manifestos of those SONs. In fig.4, an

[2]Namely, only the relations due to α affinity.

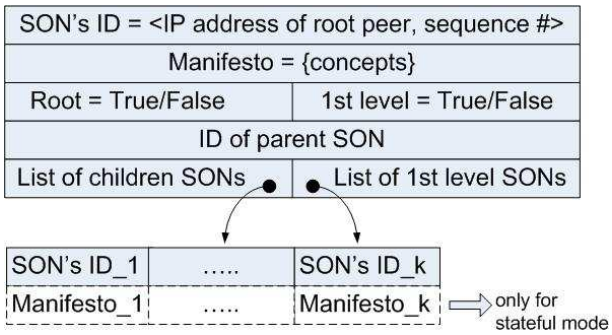


Figure 4: Data structures maintained at peers

entry of the SON table maintained by peers is shown; the dashed fields are maintained only for the stateful mode. The SON identifier is a couple $\langle$*founder ID, SON tag*$\rangle$, where the latter field is a sequential number produced by the founder, in order to differentiate SONs created by the same peer. The founder ID is a network address: when a message must be forwarded to an upstream or downstream SON, the SON ID supplies the address of its root peer, to be put into the message. An entry is recorded for each SON to which the peer belongs; the SON table is maintained by the topology manager. The policies for overlay construction, and query processing and routing, are implemented by the ontology-based addressing module. In particular, as far as overlay construction is concerned, the ontology-based addressing module takes in charge the processing of control messages exchanged in order to characterize the SONs a peer should join; this phase is discussed in sec.3.2. In sec.3.3, the procedures for contents and knowledge lookup are described.

3.2 Construction phase

When a peer p wants to join the system, it subscribes to ORION by sending a `JoinReq` to a ORION server responsible for managing the membership.[3] The server provides p with k addresses of peers, $q_1 \dots q_k$, already belonging to the system, which will help p in finding the appropriate SONs to join to. The goal is to include p into the existing SONs that best fit its interests; in case this fails, new SONs can be created. p sends to $q_1 \dots q_k$ a `ManReq` message, asking for the manifestos of the SONs they belong to. For each concept c that p has in its own ontology and for each received manifesto, p computes $dist(c,\text{manifesto})$, in order to find the manifesto nearest to the considered concept, that is, the one producing the smallest result (in absolute value). Let q be the peer that sent such a manifesto, and $\mathcal{S}$ the SON to which the manifesto ($man_{\mathcal{S}}$) belongs. Then p sends an `InterestQuery` to q for the concept c.

The `InterestQuery` navigates through the overlay, driven by the semantic links

[3]The server address can be written in the ORION executable. We do not address in this work aspects related with server management.

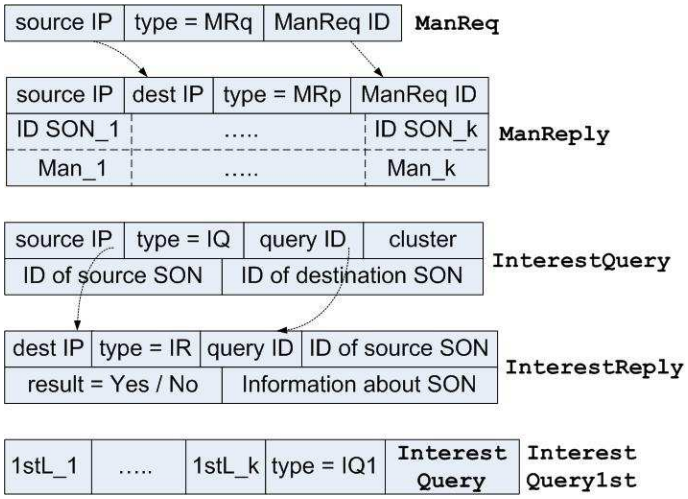


Figure 5: Structures of `ManReq`, `InterestQuery`, `InterestReply`, and `InterestQuery1st`

among SONs, with the purpose of finding the most appropriate SON p should join as an owner of concept c. SONs membership is decided according to a $radius$ parameter: if the absolute value of $dist$ between a concept and the manifesto of a SON is $< radius$ then the peer owner of the concept can become a member of that SON.

The `InterestQuery` is firstly sent by q to the root peer of $\mathcal{S}$. The `InterestQuery` is received by the P2P routing that forwards it to the ontology-based addressing, which in turns obtains from the topology manager the manifesto of the local SON with which the concept in the `InterestQuery` must be compared, and passes it to the query manager waiting for a reply. The semantic engine of the root peer applies the $dist$ function and performs the following actions:

- if $0 \leq |dist(c, man_{\mathcal{S}})| < radius$ then an `InterestReply` is sent back from the root peer to p, saying that p should become a member of the SON;
- otherwise, if $dist(c, man_{\mathcal{S}}) \leq -radius$ then the `InterestQuery` is sent to the upstream root peer;
- else (case $dist(c, man_{\mathcal{S}}) \geq radius$) the `InterestQuery` must be sent downstream.

In the first case, when a peer p receives a positive `InterestReply`, the message is handed over by the P2P routing to the ontology-based addressing module, which checks whether `InterestReply.result` has value `Yes`. In this case, the ontology-based addressing module extracts from the message the semantic information, which is passed to the topology manager to create a new entry in the SON table, with the format shown in fig.4, initialized with the data from the message. Each `InterestQuery`

produces the connection to only one SON; a peer may belong to as many SONs as the number of concepts it knows.

In the second case, no upstream root peer could exist if the current SON is a first level SON. In stateful mode, the current root peer computes the matching between c and the manifestos of all the other first level SONs; the interactions among modules are the same as described above. The `InterestQuery` is forwarded to the root peer of the SON whose manifesto minimizes the value of *dist* with c. In stateless mode, the `InterestQuery` is encapsulated into a first level Interest Query, `InterestQuery1st`, by pre-pending a header containing the list of IDs of first level SONs. The root peers of these SONs are visited in order: each root peer computes $dist(c, man_{\mathcal{S}})$ with $man_{\mathcal{S}}$ its own manifesto; if the result is ≥ 0 then the `InterestQuery` must be processed in the local tree and its processing is performed as before after dropping the added header. Otherwise, the `InterestQuery1st` is forwarded to the next root peer in the list, after deletion from the list of the current SON ID. It may occur that no tree is appropriate for p. The `InterestQuery` arrives at a root peer such that its SON does not match with the `InterestQuery`, the other trees in the forest have already been explored and the root peer's descendants do not match either. In this case, the root peer sends back a *negative* `InterestReply` to the peer source of the `InterestQuery`, which must create a new SON using the procedure described in sec.3.2.1.

In the third case above, `InterestQuery` processing is different basing on the mode of operation. In stateless mode, the `InterestQuery` is sent to the root peers of all descendant SONs. The current root peer waits for their replies, consisting in the value of the *dist* function computed on c and their manifestos: the current root peer delegates further `InterestQuery` processing to the best child – the one minimizing *dist* – basing on the obtained results. If many children return the same value of *dist*, one of them is chosen randomly. In stateful mode, the current root peer performs the function computation on all of its children manifestos in order to choose the appropriate child to which to forward the `InterestQuery`.

Two aspects must be highlighted. `InterestQuery` processing is performed only by root peers, thus limiting load on other peers. Furthermore, processing at root peers does not require to perform full concept matching – which could be very expensive in terms of latency depending on the semantic formalisms adopted – but only the much more lightweight computation of the *dist* function, thus speeding up the insertion of a peer in the overlay.

3.2.1 Creation of a SON

A SON is created by a peer when a negative reply is obtained for an `InterestQuery`. This happens for the first peer in the system, and when the knowledge held by the peer is not affine to any existing SON. A new SON could be created in one out of two positions. It could be a first level SON, or a leaf SON. The first case occurs when the `InterestQuery` does not match with the manifesto of any first level SON.

The second case occurs when a root peer discovers that an `InterestQuery` does not match with the local manifesto and is semantically more complex, but either it has not any child or none of its children has a better match.

In both cases, the procedure is as follows: the peer p that received a negative `InterestReply` for a concept c computes the cluster for c of scope $radius$, that is, the set of concepts c' in its ontology such that $|dist(c, c')| < radius$.[4] The peer then records in its data structures the existence of a SON $\mathcal{S}$ having such a cluster as manifesto, and the couple $\langle p$'s address, p's SON counter$\rangle$ as identifier.

$\mathcal{S}$ is inserted into the forest in different ways, depending on its position. If $\mathcal{S}$ is a first level SON, then its ID is sent to all other first level SONs. The root peers of those SONs update their list of first level SONs. If $\mathcal{S}$ is a leaf SON, p sends to the root peer of the parent SON – that generated the *negative* `InterestReply` – a request of becoming its child and its own ID. The data structures of the two root peers are updated so as to record the parent/child relation. In both cases, when operating in stateful mode, the manifesto $man_{\mathcal{S}}$ is also sent.

When operating in stateful mode, the overlay can be re-organized. If $\mathcal{S}$ is a first level SON, then the root peers of the other first level SONs – upon receiving $man_{\mathcal{S}}$ – compare their own manifestos with that of $\mathcal{S}$. In case the root peer r of a SON notices that $\mathcal{S}$ should be its parent, r asks the root peer of $\mathcal{S}$ to become its child, deletes its data structures as a first level SON and notifies the remaining first level SONs that it must be removed from their list. If $\mathcal{S}$ is a leaf SON and its parent SON has other children, then the root peer of the parent SON $\mathcal{S}_p$ compares $man_{\mathcal{S}}$ with the manifestos of the other children. The children of $\mathcal{S}_p$ that should become children of $\mathcal{S}$ – having a manifesto semantically more complex that $man_{\mathcal{S}}$ – are notified of the needed re-organization and pruned from their current position. Their root peers must require a re-join to the root peer of $\mathcal{S}$. Reorganization is not performed in stateless mode, because the reduced state information available at peers makes it too expensive.

With the aim of better understanding the search policy described in the next section, a characteristic of the forest must be pointed out. Given a set of peers having certain sets of concepts, the structure of the forest is not uniquely determined. Indeed, it depends on the peer addresses initially supplied to the peer, and on the navigation path through the forest, which changes along with its structure. As a consequence, if two peers own the concept represented with a shadow circle in fig.6, one of them can join the SON with manifesto $M1$ and the other peer can join the SON with manifesto $M2$. The following property can be proved.

Property 1 *The manifesto $man_{\mathcal{S}}$ of a SON $\mathcal{S}$ is the center of the concepts represented in $\mathcal{S}$. The minimum distance between the manifestos of two adjacent SONs is radius. The intersection between two adjacent SONs might be not empty.*

[4]Concepts related with c with other affinity notions may exist, but for them p must be included in different SONs.

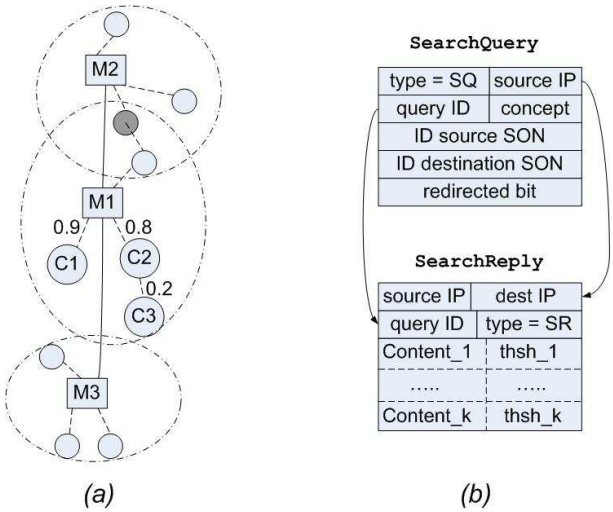


Figure 6: (a) Example of forest structure. (b) Structures of `SearchQuery` and `SearchReply`

Proof. The first claim immediately derives from the consideration that a positive `InterestReply` is generated to a peer owner of a concept c such that $|dist(c, man_{\mathcal{S}})| < radius$, that is, $-radius < dist(c, man_{\mathcal{S}}) < radius$. Hence, concepts in a SON are either semantically more complex or semantically less complex than $man_{\mathcal{S}}$. The second claim is proved in fig.6. Let us suppose that a SON $\mathcal{S}_1$ with manifesto $M1$ is first formed, and that $radius = 3$. When an `InterestQuery`($M2$) is generated by a peer p, with $dist(M1, M2) = 3$, p must create a new SON $\mathcal{S}_2$ with manifesto $M2$ upstream. The third claim is proved by the arguments above. ♣

The nondeterminism evidenced by the property is the price to pay to allow peers to not know anything about the system apart for their own local knowledge. If all the knowledge that could ever appear in the system were known in advance, then clearly separated SONs could be characterized a-priori, at the expenses of lower flexibility.

3.3 Lookup phase

Peers belonging to the system may look for concepts or contents held by other peers. Content search is performed using semantics, that is, a user specifies the concept s/he is interested in and the system provides him/her with a list of contents concerning that concept.

In fig.6(b), the structures of both `SearchQuery` and `SearchReply` are shown. They differ in a few fields from the query and reply to join the system. In a `SearchQuery`, the source peer specifies the concept c it is looking for and the threshold $thsh$ determining the search scope. A concept c' satisfies the search if the affinity relationship between c' and c has a weight $\geq thsh$.

Processing a `SearchQuery` is much the same as processing an `InterestQuery`. A

peer p generating a `SearchQuery`$(c, thsh)$, matches c against the manifestos of all the SONs it belongs to, to find out the manifesto that minimizes $dist$. Let $\mathcal{S}_0$ be such a SON, with root peer rp_0. Then, p sends the query to rp_0, that routes the query according to the same policy adopted for routing an `InterestQuery`, up to the most appropriate SON in the forest, let it be $\mathcal{S}_X$. However, processing a `SearchQuery` differs from processing an `InterestQuery` in the scope of diffusion once arrived in $\mathcal{S}_X$. Some properties can be derived about where to look for the replies to a `SearchQuery`, depending on whether the searched concept is in the manifesto of a SON or not, and on the value of $thsh$.

It is worth to notice that not all peers included in $\mathcal{S}_X$ may be able to generate replies to the query. The peers in $\mathcal{S}_X$ having concepts such that their affinity degree d with c is $d < thsh$ do not reply. In the example of fig.6(a), an overlay with $radius = 3$ is shown. Let us suppose that a `SearchQuery`$(C1, 0.7)$ is processed. The `SearchQuery` is forwarded up to the SON with manifesto $M1$ and there it is broadcast to all peers. However, the peers owning concepts $C1$ and $C2$ generate a `SearchReply`, but the peers owning concept $C3$ do not generate replies because, although $dist(C3, M1) = 2 < radius$, the affinity degree between $C1$ and $C3$ is 0.144.[5]

Query processing differs from `InterestQuery` processing because in some cases the replies can be located in different SONs. Because of the nondeterminism in the forest construction discussed in sec.3.2.1, peers with the same concept may belong to different SONs. In different peers a certain concept could be linked to different contents. With the aim of finding all the contents related with a concept, a *portion of a tree* suitable to find responses must be characterized. The following properties can be easily proved.[6]

Property 2 *If a* `SearchQuery`$(M, thsh)$ *is generated, with* M *manifesto of a SON* $\mathcal{S}$*, and* $\forall c \in \mathcal{S}$ *such that* $|dist(c, M)| = radius - 1$*,* $affinity(c, M) < thsh$*, then all possible replies can be found in* $\mathcal{S}$*, in the parent SON and in the children SONs.*

Proof. As shown in fig.6, and by Property 1, the intersection between two adjacent SONs may be not empty. As a consequence, a concept affine to M within $thsh$, and semantically less complex than M, can be either in $\mathcal{S}$ or in its parent SON. Peers with such a concept could join anyone of the two SONs, that must thus be both visited. On the other hand, the hypothesis about the concepts neighbors of M within $radius$ guarantees that the manifesto of the parent SON and its other descendants do not satisfy the query; hence, the research must not be continued in ancestors of $\mathcal{S}$ farer than its parent, nor in the other children of its parent, thanks to the transitivity property. Similar symmetrical arguments apply for the children SONs of $\mathcal{S}$. ♣

[5] If no other affinity relations between the two concepts exist besides of those depicted.

[6] For simple ontologies where edges are not weighted – such as RDF trees or WordNet – simplified properties can be proved. They are reported in Appendix A.

Property 3 *If a* `SearchQuery`$(C, thsh)$ *is generated, with* C *a concept included in a SON* $\mathcal{S}$ *and* $dist(C, M) < 0$*, and* $\forall c \in \mathcal{S}$ *such that* $dist(M, c) = -radius + 1$*,* $affinity(C, c) < thsh$*, and* $\forall c \in \mathcal{S}_P$ *parent SON of* $\mathcal{S}$ *such that* c *is semantically less complex than* M_P*,* $affinity(C, c) < thsh$*, then all possible replies can be found in* $\mathcal{S}$ *and* $\mathcal{S}_P$*. If* $dist(C, M) > 0$*, and* $\forall c \in \mathcal{S}$ *such that* $dist(M, c) = radius - 1$*,* $affinity(C, c) < thsh$*, and* $\forall c \in \mathcal{S}_C$ *child of* $\mathcal{S}$*, such that* c *is semantically more complex than* M_C*,* $affinity(c, C) < thsh$*, then all possible replies can be found in* $\mathcal{S}$ *and its children SONs.*

Proof. The first claim is proved in fig.6. By hypothesis, all concepts descendant of $M1$ do not satisfy a query on the shadowed concept; hence, by the transitivity property, no reply can be found in downstream SONs. Similarly, no concept semantically less complex than $M2$ satisfies the query; hence, no reply can be found upstream by the transitivity property. Similar symmetrical arguments are used to prove the second claim. ♣

Let us indicate with $\mathcal{S}_U$ a generic upstream SON of $\mathcal{S}$, and with $\mathcal{S}_D$ a generic downstream SON of $\mathcal{S}$.

Property 4 *If a* `SearchQuery`$(C, thsh)$ *is generated, with* C *a concept included in a SON* $\mathcal{S}$*, then all possible replies can be found in* $\mathcal{S}$*, and*

- *in the ancestor SONs up to* $\mathcal{S}_U$ *satisfying* $\forall c \in \mathcal{S}_U : dist(c, man_U) = -radius + 1, affinity(c, C) < thsh$ *and in the parent SON of such* $\mathcal{S}_U$*,*
- *in the descendant SONs up to* $\mathcal{S}_D$ *in each branch, satisfying* $\forall c \in \mathcal{S}_D : dist(c, man_D) = radius - 1, affinity(c, C) < thsh$ *and in all children of all such* $\mathcal{S}_D$*.*

Proof. The limitation on the set of ancestors to be visited derives from the extension of the first claim of Property 3. The limitation on the set of descendants derives from the extension of the second claim of Property 3. The additional levels to be visited are a consequence of the non-empty intersection between adjacent SONs (Properties 1 and 2). ♣

As a consequence of the properties above, processing of a `SearchQuery` is performed as follows: once the most appropriate SON $\mathcal{S}_X$ has been found, the `SearchQuery` is broadcast by the root peer rp_X of $\mathcal{S}_X$ to all the members of $\mathcal{S}_X$. If a peer in $\mathcal{S}_X$ having a concept c such that $|dist(c, man_X)| = radius - 1$ can reply to the `SearchQuery` and thus has $affinity(c, C) \geq thsh$, such a peer sends a notification to rp_X that the `SearchQuery` must be furtherly forwarded to upstream/downstream SONs, depending on whether c is semantically less complex or more complex than man_X respectively. Upon receiving such notifications, rp_X forwards the `SearchQuery` to the parent and children root peers as appropriate, marking it as `re-directed`. Marking is needed to force processing of the `SearchQuery` in a SON such that the searched concept could be too far from the manifesto, but sufficiently near to concepts held in the SON. This

procedure is recursively repeated in upstream and/or downstream SONs till reaching SONs where the equations in Property 4 hold. Each root peer receiving such a re-directed `SearchQuery`, broadcasts it in its SONs waiting for possible notifications.

A peer generating a `SearchReply` includes in it the name of the contents locally held, linked to the concepts that allow the `SearchReply` generation. Content name and peer address allow the querier to download the contents it is interested in.

3.4 Discussion

In the ORION description no assumption has been done with respect to the value of *radius*, which affects the degree of aggregation of peers into SONs. The larger *radius*, the more aggregate are peers, the lower is the overhead for overlay management, the more probability a search with broad scope has of finding all the responses in a certain SON, the more peers in the SON have to uselessly process a search with narrow scope without being able of generating a reply. Hence, the value of *radius* should be chosen also basing on the expected scope of user searches.

For the sake of simplicity, in the previous description we neglected the possibility that a peer belong to multiple SONs. Indeed, this case can be exploited to optimize query routing. A root peer belonging to multiple SONs can match the concept contained in a query with the manifestos of all SONs it is a member of, and choose the most appropriate SON. This policy allows a query to jump through the forest structure, taking a sort of short cut to reach the appropriate SON. As a consequence, the more manifold and heterogeneous are the interests of the peers, the better is the system behavior.

So far, only the case of content lookup has been considered. The system could be as well used for knowledge retrieval. In [3], `ProbeQuery` messages can be used to discover concepts related with knowledge already held by a peer, and peers with related knowledge. Using these messages in ORION would allow to enrich peer ontologies with edges pointing to other peers possibly included in different SONs, thus also enriching content search results. However, ORION should be slightly modified in order to allow the dissemination of a `SearchQuery` into remote SONs where affine peers are connected.

To cope with leave requests from root peers, a solution could be that of recording in each peer the same data structures maintained at the root (fig.4), as far as the tree topology is concerned. By contrast, the SON membership can be transparent to peers, in case a routing infrastructure is built within each SON to perform intra-SON query broadcast. The topology of the infrastructures could be chosen in order to optimize either network aspects or application aspects. In the former case, low number of duplicates and robustness could be achieved by adopting for instance a hypercube topology. In the latter case, the peers in a SON could be organized in a tree structure with peers more similar to the manifesto, or more useful in terms of replies generated in the past, nearer to the root peer.

As an alternative, SONs could be unstructured and their membership unknown, and epidemic techniques [1] could be used to diffuse `InterestQuery` messages. These techniques allow to save bandwidth with respect to flooding, at the expenses of higher latency in reaching all nodes.

4 Conclusions

In this work we describe the ORION infrastructure for the construction of an overlay network in peer-to-peer systems in order to ease semantic retrieval of contents and concepts. The overlay topology is based on affinity amongst the concepts held by peers; peers are grouped in Semantic Overlay Networks according to their interests. ORION does not make any assumption about the concepts held by peers, nor an a-priori knowledge of those concepts is needed. The procedures to build the infrastructure and to route queries for concepts and content retrieval are described. Properties are derived about the structure of the overlay, to provide guarantees about the capability of satisfying queries. The implementation of ORION is currently ongoing in the framework of a simulation environment, with the aim of measuring its performance in comparison with other solutions proposed in the literature.

References

[1] Akdere A., Bilgin C., Gerdaneri O., Korpeoglu I., Ulusoy A., Cetintemel U., "A Comparison of Epidemic Algorithms in Wireless Sensor Networks". *Computer Communications*, 29(13), 2006.

[2] Castano S., Ferrara A., Montanelli S., Pagani E., Rossi G.P., "Ontology-Addressable Contents in P2P Networks". *Proc. 1st Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID'03)*, May 2003, pp. 55-68.

[3] Castano S., Ferrara A., Montanelli S., Pagani E., Rossi G.P., Tebaldi S., "On Combining a Semantic Engine and Flexible Network Policies for P2P Knowledge Sharing Networks". *Proc. 1st International Workshop on Grid and Peer-to-Peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems (GLOBE 2004)*, Aug. 2004.

[4] Castano S., Ferrara A., Montanelli S., Racca G., "From Surface to Intensive Matching of Semantic Web Ontologies". *Proc. 3rd DEXA Int. Workshop on Web Semantics (WEBS 2004)*, IEEE Computer Society, Aug.2004.

[5] Clarke I., Sandberg O., Wiley B., Hong T. "Freenet: A Distributed Anonymous Information Storage and Retrieval System". *Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[6] Crespo A., Garcia-Molina H., "Semantic Overlay Networks for P2P Systems". *Technical Report*, Computer Science Department, Stanford University, Oct. 2002. http://citeseer.ist.psu.edu/garcia02semantic.html

[7] Horrocks I., "DAML+OIL: A Description Logic for the Semantic Web". *IEEE Bulletin of the Technical Committee on Data Engineering*, 2002. http://www.cs.man.ac.uk/~horrocks/Publications/download/2002/ieeede2002.pdf

[8] Napster. www.napster.com

[9] Princeton University Cognitive Science Laboratory. "WordNet - a Lexical Database for the English Language". http://wordnet.princeton.edu/

[10] Ratnasamy S., Francis P., Handley M., Karp R., Shenker S., "A Scalable Content-Addressable Network". *Proc. of the ACM SIGCOMM 2001*, 2001.

[11] Reed S.L., Lenat D.B., "Mapping Ontologies into Cyc". 2002. http://www.cyc.com/doc/white_papers/mapping-ontologies-into-cyc_v31.pdf

[12] Scott Cost R., Finin T., Joshi A., Mayfield J., Shah U., "Information Retrieval on the Semantic Web". *Proc. 11th Intl. Conf. on Information and Knowledge Management*, Nov. 2002. http://umbc.edu/~finin/papers/cikm02/cikm02.pdf

[13] Tang C., Xu Z., Dwarkadas S., "Peer-to-peer information retrieval using self-organizing semantic overlay networks". *Proc. ACM SIGCOMM 2003*, pp. 175-186.

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

[15] Voulgaris S., Kermarrec A., Massoulié L., van Steen M., "Exploiting Semantic Proximity in Peer-to-Peer Content Searching". *Proc. 10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, Nov.2001. http://citeseer.ist.psu.edu/voulgaris04exploiting.html

[16] Voulgaris S., van Steen M., "Epidemic-style Management of Semantic Overlays for Content-Based Searching". *Proc. EuroPar 2005*, Aug. 2005.

[17] W3C Consortium. "OWL Web Ontology Language". W3C Recommendation, Feb.2004, http://www.w3.org/TR/owl-semantics/

[18] W3C Consortium. "Resource Description Framework (RDF)". http://www.w3.org/RDF/

[19] W3C Consortium. "Semantic Web Activity". http://www.w3.org/2001/sw/

[20] Wolfgang N., Lser A., Wolpers M., Siberski W., Schmitz C., Schlosser M., Brunkhorst I., "Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks". *Proc. WWW'03*.

A Properties for simple ontologies

In this appendix, we prove some properties about the overlay infrastructure built by ORION in case of ontologies with edges not weighted.

Property 5 *If a* `SearchQuery`$(M, thsh)$ *is generated, with* M *manifesto of a SON* $\mathcal{S}$ *and* $thsh < radius$ *the affinity threshold, then all possible replies can be found in* $\mathcal{S}$*.*

Proof. Let us consider fig.6(a), where solid lines represent semantic relations according to the α affinity within a SON, and dashed lines represent the same relation between concepts in different SONs. The *radius* is 3, that is, concepts are included in a SON such that their distance from the manifesto is at most 2. If a query is generated for $M1$ with threshold 2, replies for it are all the concepts included in the SON having $M1$ as manifesto. ♣

Property 6 *If a* `SearchQuery`$(C, thsh)$ *is generated, with* C *a concept included in a SON* $\mathcal{S}$ *and* $thsh < radius$ *the affinity threshold, then all possible replies can be found in* $\mathcal{S}$*, in the parent SON, and in the children SONs of* $\mathcal{S}$*.*

Proof. Let us consider fig.6(a), and let us suppose that a query is generated for a concept C with threshold 2. If $C = M1$ then the same arguments as before apply. If c is a concept more complex than the manifesto and at the maximum allowed distance from the manifesto, such as $C3$ in the figure, then all replies to the query are found in the SON, and in the child SON with manifesto $M3$, up to and including the peers with concepts at distance 1 from $M3$. Similar arguments apply in case c is a concept less complex than the manifesto and at distance $radius - 1$ from it, to prove that responses can be found in the parent SON with manifesto $M2$. ♣

Property 7 *If a* `SearchQuery`$(C, thsh)$ *is generated, with* C *a concept included in a SON* $\mathcal{S}$ *and* $thsh \geq radius$ *the affinity threshold, then all possible replies can be found in* $\mathcal{S}$*, and in the ancestor and descendant SONs of* $\mathcal{S}$ *such that the number of hops between the root peer of* $\mathcal{S}$ *and their root peers is equal to* $\lceil thsh/r \rceil$*.*

Proof. The proof can be easily obtained by generalizing the previous case. ♣