Introducing Autonomic Behaviour in Semantic Web Agents

Valentina Tamma, Ian Blacoe, Ben Lithgow-Smith, and Michael Wooldridge

Department of Computer Science, University of Liverpool,
Liverpool L69 3BX, United Kingdom

Abstract. This paper presents SERSE – SEmantic Routing SystEm– a distributed multi-agent system composed of specialised agents that provides robust and efficient gathering and aggregation of digital content from diverse resources. The agents composing SERSE use ontological descriptions to search and retrieve semantically annotated knowledge sources, by maintaining a *semantic index* of the instances of the annotation ontology. The efficient retrieval is made it possible through the semantic routing mechanism, that permits to identify the agent indexing the resources requested by a user query without having to maintain a central index, and by reducing the number of messages broadcasted to the system. The system is also capable of exhibiting autonomic behaviour. Autonomic behaviour is characterised by self configuration and self healing capabilities, aimed at permitting the system to manage the failure of one of its agents and ensure continuous functioning.

1 Introduction

The Semantic Web primarily aims to share knowledge from distributed, dynamic, and heterogeneous sources, whose content is expressed in a machine-readable format by means of languages such as RDF [1] and OWL, in a similar way to that in which information is shared on the World Wide Web. Agents play an integral role in this vision; they use these machine-readable representations to gather and aggregate knowledge, as well as to reason in order to manage inconsistencies, and to infer new facts. Together with their ability to process Semantic Web content, agents contribute features, such as distribution, autonomy, and social ability, that make them particularly suited to manage large, heterogenous, and distributed knowledge bases. In recent years, many tools have been developed for managing traditional knowledge sources, but such approaches usually imply a centralised, and static environment where the ultimate control is centralised. This type of approach does not promise to scale well to the Semantic Web, which is an open, dynamic, and often chaotic environment.

Distributed, decentralised systems are thought to be a better alternative for scalability [2]; their architecture is characterised by system components each with equal roles and the capability to exchange knowledge and services directly with each other. Peer-to-peer technology (P2P) such as Edutella [2] or Morpheus [3] is a possible answer to this quest for decentralisation. P2P systems are networks of peers with equal roles and capabilities, and recently peer-based management systems have been proposed, which exploit P2P technology for sharing and retrieving huge amounts of data [4]. However,

Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 653–667, 2005.

most approaches are oriented at file sharing, rather than at the management of semantically enriched content as provided by the Semantic Web. The agent paradigm seems to offer equally good prospects for the management of semantically annotated content: on the one hand, agents are intrinsically distributed, and platforms for agent oriented programming offer standardised communication protocols and management mechanisms (for instance, Jade [5]). On the other hand agents can provide "smart", service-based support for autonomous semantic web tools, and well-automated discovery mechanisms for advertising and locating resources within an open framework, established trust and reputation frameworks, and proactive support for fact maintenance [6]. One way in which the adoption of the agent-oriented paradigm can be beneficial to semantic web applications is by making them exhibit *autonomic behaviour*. Autonomic computing is an emerging branch of software engineering promoting the design and implementation of self-managing systems, many of which consist of several interacting, autonomous components that in turn comprise large numbers of interacting, autonomous, self-governing components at the next level down [7]. This type of behaviour is intended to make it easier to manage the complexity and scalability of complex distributed systems, such as those to manage Semantic Web content.

In this work we concentrate on the *robust* and *efficient* gathering and aggregation of digital content from diverse resources. We developed a multi-agent system composed of specialised agents that is able to search and retrieve semantically annotated knowledge sources. In addition to searching for digital content, the semantic information used to annotate resources is used to explore the addition of autonomic features to the system, in order to equip it with self-management and self-healing capabilities, aimed at permitting the system to manage the failure of one of its agents and ensure continuous functioning. In this paper we introduce the system SERSE (SEmantic Routing SystEm) and its main functionalities. This paper extends our previous work in this area [8,9] by introducing the autonomic behaviour features exhibited by SERSE and by presenting details of its multi-platform implementation. In the remainder of this paper we describe the system's conceptual architecture and the information flow between the system components. We examine the two main functionalities offered by the system, namely query management and autonomic behaviour, and we present a set of experiments aimed at evaluating the performance for each of these functionalities.[1]

2 SERSE

SERSE's primary goal is to enable the semantic retrieval and aggregation of the digital content of web resources. SERSE is designed as a multi-agent system composed of specialised agents capable of functioning in a scalable, self-managing, open, and dynamic fashion. The system requires resources to be semantically annotated according to one or more ontologies expressed in OWL, and at present is not capable of discovering annotated resources autonomously. For this purpose SERSE relies on the Annotation System component of Esperonto, that informs it of newly acquired content providing references

[1] SERSE was developed as part of the now concluded Esperonto projectIST-2001-34373 whose aim was to provide a set of tools for performing the transition from the traditional web to the semantic web [8].

to both the resources and their RDF annotations. The description of the Annotation System is outside the scope of this paper. However, for the purpose of describing SERSE, it is sufficient to say that annotations are semi-structured representations of information referencing instances (of one or more concepts in the annotation ontology) that appear in the content of web resources. [2]

The core of the system is represented by a network of specialised agents providing *indexing* and *routing* functionalites, that permit them to efficiently retrieve resources based on the semantics of their content. Each agent is *specialised* with respect to a concept, meaning that it can access the resources whose annotations contain instances of that concept, and it is only aware of those agents specialised with concepts that are *similar* or *related* to its own. Therefore, the agent network is organised into *semantic neighbourhoods* that mirror the structure of the ontology (in terms of the hierarchical and specific relationships defined in the ontology).

Neighbourhoods are partially overlapping, and this permits the routing mechanism to find the answer to a query in a limited number of hops, without having to browse the whole ontology and without having to flood the network with a large number of messages. Semantic neighbourhoods are automatically determined when the system receives a notification of new ontological content – received as new concepts are used to annotate resources. The neighbourhoods are not static but they dynamically change as the system is required to handle further notification of new ontological content, or if the ontology is modified (and a new version of the ontology is used in the annotation). In this way, we have multiple overlapping neighbourhoods, each centred on one concept, and agents have knowledge only of the agents composing their neighbourhood.

Indexing ontological content consists of creating structures that link resources, identified through their URLs, to RDF statements describing instances of the concepts in the ontologies. The routing functionality permits SERSE to route queries to the agents that are capable of retrieving the resources annotated with the concepts they are specialised on. SERSE handles queries expressed in RDQL [10] (an RDF query language developed by HP as part of the Jena toolkit) [11] on any combination of concepts and concepts properties (including object properties). Complex queries are decomposed into simple ones, each regarding a single concept. Each simple query is sent to one of the agents in the network of routers, and the agent consults its index to determine whether it can answer the query. If the agent cannot answer the query, then it routes the request to the agent in its neighbourhood that handles the concept *closest* to the one in the query. We evaluate similarity between concepts according to the approach proposed by [12]. However, we modified the algorithm so that it exhibits a greedy but less precise behaviour, implemented through heuristics, and that provides a higher number of potential matches. Ehrig and Staab's approach is aimed at ontology mapping, a process that can be taken off line and requires high precision in order to establish the correct mappings. Semantic routing is different in nature: the evaluation of similarity should be sufficiently precise to determine a new agent to whom the query can be routed, not necessarily the *best* agent. In addition, semantic routing is a dynamic process executed on line, and therefore it requires fast computation in order to minimise the time spent by the user waiting for an answer. We discuss in more detail the indexing and routing in Section 4,

[2] We are currently working at making SERSE a standalone system.

where these functionalities are related to the component of SERSE's architecture that provides them.

In addition to the main indexing and routing facilities, the system is also intended to be self-governing; it uses autonomic computing techniques to preserve index knowledge and to adjust the index connections when one or more indices within the system are unavailable. Autonomic behaviour is also used to maintain the system operative in case of failure of one agent or one platform. Section 5 describes the mechanisms used to implement autonomic behaviour in SERSE.

3 Conceptual Architecture

SERSE's conceptual architecture is composed of six types of specialised agents providing different functionalities. The heart of the architecture is composed by the network of `Router agents`, providing indexing and routing capabilities. These agents are complemented by a number of other specialised agents providing ancillary services, that implement system management functions. Figure 1 shows the different roles played by agents in SERSE and the message flow in the system.

SERSE is built within JADE – a FIPA compliant agent deployment environment [5]. The system is designed to be distributed over a number of JADE platforms, on different host machines, with each platform containing a part of the indexing system and its own interface agent set. This enables the system to operate even when reduced to one platform, and to dynamically reconfigure the index network in response to temporary or permanent outages of agents and platforms in the system. It also uses the JENA semantic web toolkit to handle RDFS, OWL, and RDQL. SERSE is able to use ontological

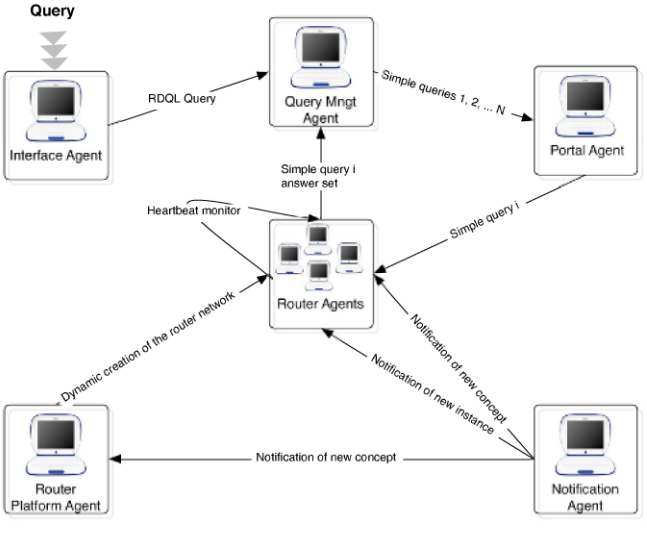

Fig. 1. SERSE conceptual architecture

definitions expressed in either RDFS or OWL (Lite and DL), using the full range of expressions available. The different roles that agents play in SERSE are described below, and Figure 2 shows the interactions between the different types of agents on a single platform and on multiple platforms.

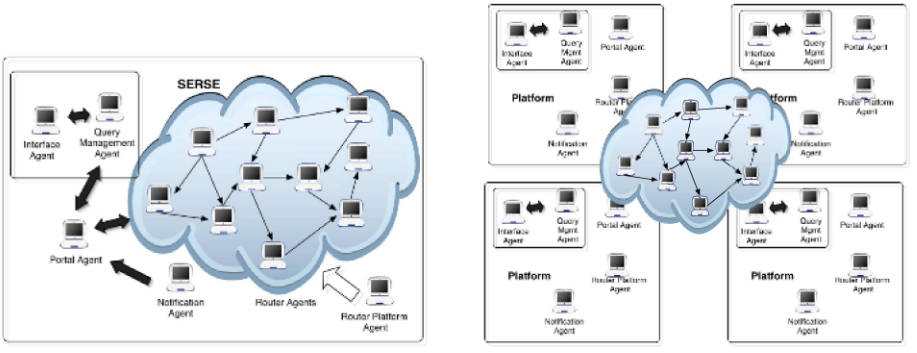

Fig. 2. SERSE architecture on a single platform and distributed over multiple platforms

– `Router Agents`: Router agents provide the core functionalities of the system: indexing, routing and self-management. In order to provide these functionalities agents maintain two types of indices, a *content index* and a *routing index*. The content index stores the URI identifying the RDF statements referring to instances of some resources, together with the URLs used to identify them. The routing index stores the communications address and concept handled by each of the router agents that are semantic neighbours. Routing indexes contain entries for three types of neighbour links:
  - *actual*: neighbour concepts which are handled by existing agents;
  - *ontology*: neighbouring concepts (according to the ontology) for which no agent yet exists; and
  - *implied*: concepts outside the neighbourhood that are presumed to exist, and to be linked to existing concepts. These links can be implied from the absence of one or more ontology neighbours.

  Implied and ontology neighbours are used to provide some of the self management functionalities described in Section 5 and are used to query the routers even if the generation of the network is not complete, and, more in general, in all cases when a path between two agents should have been established, because they concern concepts in the ontology that are related, but the link has not been created, yet. By means of this mechanism, each agent is responsible for a sub-set of the total system knowledge and has only localised knowledge of its semantic neighbours.

  `Router agents` are also equipped with self management capabilities that allow them to actively respond to changes in the state of their neighbourhood. In order to ascertain the actual status of their neighbourhood, Router agents employ two

types of messages: they both monitor the result of their outgoing routing messages (to verify that they do not return an error), and they periodically send *heartbeat messages* [7] that "ping" their neighbour. In addition, Router agents periodically save the state of their content and routing indexes enabling the knowledge to be recovered following any failure of the agent. Router agents are distributed over multiple platforms, while the other agents described below are replicated for each of the platforms.

- `Router Platform Agents`: They enable the distribution over multiple platforms and provide management services, such as the creation of a new `Router agent`, for each agent platform on which the network of routers is distributed. The `Router Platform Agent` is also responsible for triggering the dynamic creation and adjustment of the network of routers upon receipt of the notifications of new content, as described in Section 5
- `Notification Agents`: They are the interface between each platform and the Annotation System of Esperonto, and receive notifications regarding the annotation of new resources, or the addition of new concepts in the ontology. They decompose notifications regarding multiple concepts and re-send these atomic notifications into the `Router Agents` network as Agent Communication Language messages.
- `Interface Agents`: They provide a connection between each agent platform and the software components operating outside the platform, such as the web-based query interface, by creating a socket interface and passing query and response objects across it.
- `Query Management Agents (QMA)`: They decompose complex queries, that involve multiple concepts linked by logical connectives, into atomic queries. The atomic queries are then sent into the `Router Agent` network; when the QMA receives the responses to each query, these are aggregated by re-applying the logical connectives, thus producing a set of web resources that match the constraints expressed in the complex query. During the process duplicate instances are identified and removed.
- `Portal Agents`: They act as a gateway into the `Router Agent` network, through which all atomic notifications and queries are passed. Each platform in the system has a `Portal Agent`, that maintains a list of *significant points* within the router system, and send messages into the network by initially routing them to the most appropriate of these points.

Finally, the other main component of SERSE is the web-based query interface. This enables the construction of queries using concepts from multiple ontologies, logical connectives between the concepts, and specification of the values of concept properties. Responses to queries are displayed as lists of web resources, identified by URLs, that match query constraints together with the URIs of the instances that annotate them. In addition, query replies also contain a list of the concepts that are neighbours of each the responding agents. This enables follow-up queries in which the original query is modified by changing property values of concepts, exchanging one concept for a similar one, broadening or narrowing a query by substituting ontological ancestors or descendents of a concept, etc.

4 Query Management

As mentioned in Section 2, SERSE handles queries specified in RDQL on any combination of concepts and concept properties (including object properties). Queries are sent from the local `Interface Agent` to the local `QMA`, where they are decomposed into atomic queries. Query decomposition is achieved by syntactically parsing the query and identifying blocks that form atomic queries, but preserve the semantics of the original query.

The `QMA` sends each atomic query to the local `Portal Agent`, which forwards each of them to the most competent `Router Agent` known to the local `Portal Agent`. In the current implementation of SERSE, these agents are those which have knowledge of the root nodes of each of the ontologies that have been notified to the system. The purpose of this initial semantic routing is to enter the router network in the general semantic area of the queried concept improving the efficiency of the routing process. Although routing first to the root node agents might potentially be perceived as a bottleneck, these agents are effectively those that are likely to have the smallest workload from handling queries. In fact, in the domain ontologies used by SERSE, as well as in most domain ontologies, the majority of the instances are direct instances of very specific concepts (leaf nodes), whilst root nodes have few (if any) instances. Therefore, the additional routing effort of these agents is compensated by answering fewer queries. In addition, any set of significant entry points could become a bottleneck, and alternatives are constrained by the processing necessary to identify the best entry point, and message workloads.

Once an atomic query is received by the appropriate `Router Agent`, it extracts the query constraints expressed in RDQL, then it consults its content index to check if it stores the URI of instances of the query concept. Any instances that match the query contribute to the answer set, which consists of a list of resources that are described by matching instances, and is returned directly to the `QMA` that sent out the query. Included in the query reply is information about the concepts handled by the replying `Router Agent` and the agents address which is then used in follow-up queries. This then enables users to semantically browse from one concept to other closely related concepts, using knowledge about these relationships held by the `Router Agent` and revealed by the original query.

If no instance is referenced in the content index, the query is routed to the semantic neighbour with the most similar expertise This semantic routing mechanism is designed to move messages in a series of hops across the network of `Router Agents`, until the message is addressed to the `Router Agent` indexing instances of the concept in the message.

5 Autonomic Behaviour

SERSE has been designed to autonomously react to a number of events that can affect its processing. These include the notification of new ontology, but also exceptional events such as the controlled shut down of an agent. The aim is to have a system that can work in an open environment, such as the Semantic Web, and that is scalable, robust, and

requires limited human intervention for its functioning. For this reason, SERSE has been designed as a multi-agent system in which agents can join and leave the system without having to take (part of) the system off-line, or without degrading the performance of the system.

Autonomic behaviour in SERSE supervises two main functionalities: dynamic management of the network of router agents, and failure management.

The management of the router agents consists mainly of the of the operations to create the network of routers from scratch once the system is notified by the `Notification Agent` that a new ontology is available. Failure management consist of the functionalities that enable the system to continue to operate despite the temporary or permanent loss of agents or whole platforms from an existing index network. Autonomic behaviour is achieved by a number of different mechanisms:

- *Creation requests messages*: When the `Notification Agent` in one of the platforms receives a notification of new annotation ontology, it determines autonomously the root concept(s) and generates a creation request message for each of these concepts, to be sent to the `Router Platform Agent`, that in turns, creates a router agent for each root concept.
- *Router network population*: The population of the network of routers is triggered by the notification of new content messages received by SERSE. If the message notifies instances of a concept for which a router agent has not yet been created, the `Router Platform Agent` creates a new `Router agent`, and each of the neighbouring router agents affected by this event update their neighbourhood indices, with the pointers to the new actual neighbours. In this situation, ontology and implied links are created, in order to fill gaps between the existing routers and the newly created one.
- *Heartbeat monitor*: `Router Agents` monitor the success of messages sent to neighbours, and record this in their routing index. When messages are unsuccessful the neighbour is first set to a warning level, and if failure continues for a short time the entry is marked as unavailable. The neighbour will be considered available again if a message is received from it within a time period, but otherwise will eventually be removed from the neighbourhood.
- *Index backup and backup recovery*: `Router Agents` periodically save their knowledge to an XML backup file, which enables the recovery of knowledge following the failure of the `Router Agent` or platform. The knowledge stored in the file consists of the contents of both the content index and routing index. Recovery from failure of a platform is addressed by having the `Router Platform Agent` on start-up (following a manual platform re-start) check for saved state files, and, if any are found, re-creating `Router Agents` using the stored knowledge. Recovery from the failure of individual `Router Agents` is addressed by them contacting the local `Router Platform Agent` when they shut-down, and the `Router Platform Agent` will then use the saved state to re-create the `Router Agent`.
- *`Router Agent` shutdown procedure*: When `Router Agents` are subject to a controlled shut-down of their platform, they immediately save their knowledge to file, and then contact each of their neighbours to inform them of the shut-down.

This enables the neighbours to reactively adapt their neighbourhood connections to reflect the loss of neighbour. Recovery from shut-down, like that for failure, is initially a manual process but once started the `Router Platform Agent` will detect the saved-states and restore the `Router Agents`.

6 Experimental Evaluation

We conducted a number of experiments aimed to analyse the performance of the two main functionalities provided by the system: query management, and autonomic behaviour. In our experiments, we used two ontologies developed as part of the use-cases of Esperonto, the Fund Finder and the Cultural Tour ontologies for which we had also the annotated documents storing the instances of the concepts. The Fund Finder is expressed in OWL-Lite, and it is composed of around 50 concepts (12 of which are root concepts), and of 118 instances. The Cultural Tour ontology is an RDFS ontology composed of 60 concepts, and has more that 61000 instances.

In order to test the performance of the query management process we measured, for each ontology, the round-trip reply time for a set of twenty fixed queries, listed in increasing order of complexity. Figure 5 and Figure 6 illustrate the last query we posed for each of the ontologies, in order to show the level of complexity of the queries used in the experiments. The queries were posed to SERSE in sequence, and for each query we performed 1000 repetitions, in order to guarantee the reliability of the results. Figure 3 shows the response time, averaged over the repetitions, for each of the ontologies. We have compared these results with those obtained by qurying the static RDF model in Jena, the response times averaged over 100 repetitions for each query are depicted in Figure 4.

With respect to the autonomic behaviour exhibited by SERSE, we measured, for each of the two ontologies, the query response time in two different scenarios. Scenario 1 aims to test how well SERSE copes with the notifications of new content. This was achieved by creating new `Router agents` along the route of a query, by means of in-

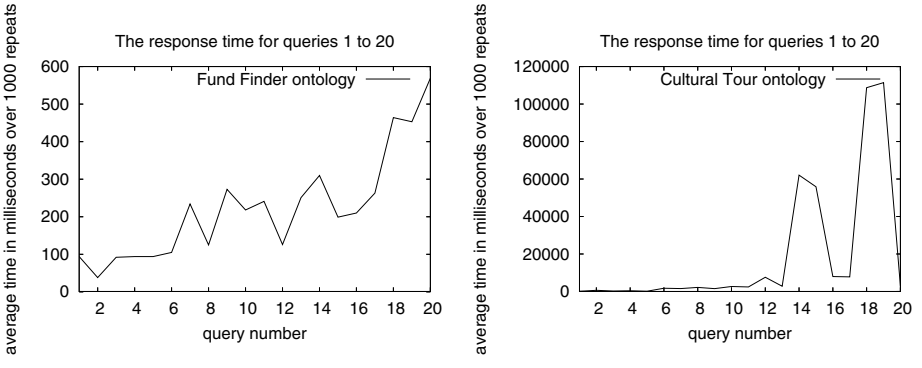

Fig. 3. SERSE response times in relation to queries about the Fund Finder and Cultural Tour ontology

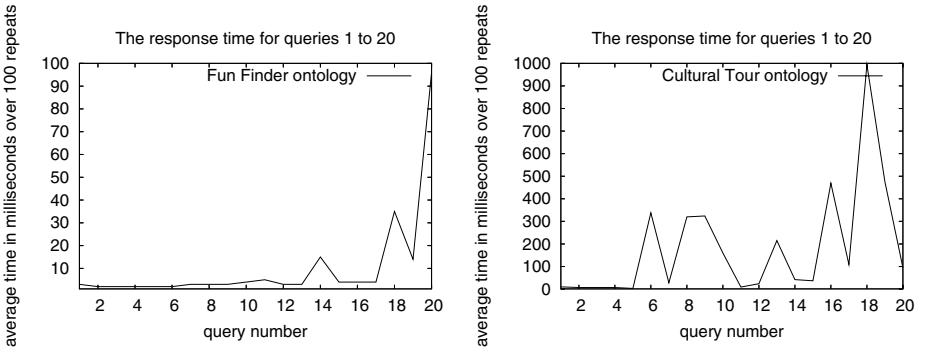

Fig. 4. Jena response times in relation to the same queries for each of the two ontologies

```
SELECT ?x, ?z WHERE
(?x, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/Fund_Finder.owl#Discount>)
(?x, <http://www.blacoe.uk/Fund_Finder.owl#Aims>, ?y)
(?y, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/Fund_Finder.owl#Objective>)
(?y, <http://www.blacoe.uk/Fund_Finder.owl#objectiveName>, "Company_Creation")
(?x, <http://www.blacoe.uk/Fund_Finder.owl#negotiated_by>, ?u)
(?u, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/Fund_Finder.owl#Negotiator_Body>)
(?u, <http://www.blacoe.uk/Fund_Finder.owl#actsForBody>, ?t)
(?t, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/Fund_Finder.owl#State_Funding_Body>)
(?z, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/Fund_Finder.owl#subvention>)
(?z, <http://www.blacoe.uk/Fund_Finder.owl#Deadline>, "30-juny-2005")
(?z, <http://www.blacoe.uk/Fund_Finder.owl#Aims>, ?w)
(?w, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/Fund_Finder.owl#Objective>)
(?w, <http://www.blacoe.uk/Fund_Finder.owl#objectiveName>, "Quality")
(?z, <http://www.blacoe.uk/Fund_Finder.owl#hasRelatedRegulation>, ?v)
(?v, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/Fund_Finder.owl#Diari_Oficial_de_la_Generalitat_de_Catalunya>)
(?v, <http://www.blacoe.uk/Fund_Finder.owl#date>, "26/04/1996")
```

Fig. 5. Query number 20 for the Fund Finder ontology

```
SELECT ?x, ?z WHERE
(?x, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/tesauro#RelacionExistenciaPersona>)
(?x, <http://www.blacoe.uk/tesauro#referencia>, "500001146")
(?x, <http://www.blacoe.uk/tesauro#entidad_existente>, ?y)
(?y, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/tesauro#Persona>)
(?y, <http://www.blacoe.uk/tesauro#fuente>,
       "Nadia Sokolova [Barcelona (CapCom) : Espaa?, ? - Barcelona (CapCom) : Espaa?, ?]")
(?y, <http://www.blacoe.uk/tesauro#autor_anotacion>, "prototipo")
(?z, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/tesauro#RelacionCreacion>)
(?z, <http://www.blacoe.uk/tesauro#estado>, "provisional")
(?z, <http://www.blacoe.uk/tesauro#creacion_relacionada>, ?w)
(?w, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
       <http://www.blacoe.uk/tesauro#ObraLiteraria>)
(?w, <http://www.blacoe.uk/tesauro#tipo_obra_literaria>, "articulo")
(?w, <http://www.blacoe.uk/tesauro#referencia>, "EL PASEO DE ROSALES ")
```

Fig. 6. Query number 20 for the Cultural Tour ontology

troducing messages notifying the acquisition of new content – that is, of new resources containing instances of some concept that was not instantiated before. The experiment was designed to implement the following procedure:

1. Remove all notifications concerning resources containing instances of a concept, for instance `Organisation Applicant` in the Fund Finder ontology;
2. Add a new notification for the concept `SME`, subsumed by `Organisation Applicant`;
3. Build SERSE: this consists of starting the `Router Platform agent` for the platforms, loading the ontology model and the notifications, and the dynamic generation of the network of routers from the notifications;
4. Run query no. 1, an atomic query with subject SME;
5. Notify one resource with instances of `Organisation Application`;
6. Run query no. 2, an atomic query with subject SME;
7. Notify one resource with instances of `Company`;
8. Run query no. 3, an atomic query with subject SME;

Figure 7 illustrates the relations existing between the concepts in the ontology that are used in the notifications and queries of Scenario 1. Scenario 2 aims to test how the system responds to an increase in the workload due to introducing agents in the semantic neighbourhood, and hence to the increase in the number of semantic similarity (and relatedness) calculations that needs to be performed during the semantic routing process. The process followed to set up the experiment mirrors the process followed in Scenario 1, but it uses different parts of the ontologies, and receives notifications related to five concepts.

Figure 8 shows the response times for the queries posed to the system in both scenarios. The experimental data concerning the round trip response time to different queries shows that the query management process implemented in SERSE takes a longer time to answer the queries when compared with Jena. This result is quite predictable because SERSE adds the overhead of the messages exchanged in order to enable the semantic routing and the system's self management . However, SERSE is still quite efficient, keeping the response time generally under the second. In that respect the results obtained are very promising. However, there some anomalies with queries number 14, 15, 18, and 19 in the Cultural Tour ontology. We have identified a number of reasons that contribute to these anomalies:

Fig. 7. The Fund Finder concepts used in the experiments of Scenario 1

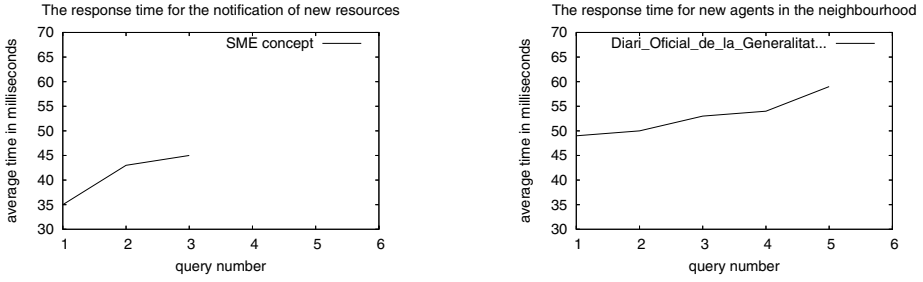

Fig. 8. Response times in relation to the queries in Scenario 1 and Scenario 2

1. Number of instances returned by each atomic query: For each query we match large sets of instances by URI, and then we match them with the corresponding resources by URL.
2. The time that RDQL takes to process the RDF model: This time varies considerably, as it can be seen by the values in Figure 3, and it is proportional to the number of statements in the RDF model.
3. Large sets of instances and resources returned: the resulting query result messages are quite large and the transmission time increases.
4. Time necessary to check for duplicates when large number of resources are returned as results of complex queries.
5. Number of semantic calculations performed: that is the length of the routing path and the number of neighbours for each of the agents in the path. The effect of the increase in the number of calculations is, however, negligible, as confirmed by the experiments for Scenario 1 and Scenario 2.

With respect to the results obtained when testing the autonomic behaviour, we can see that SERSE is able to dynamically adjust its network of routers in order to cope with the notification of new content and with the addition of new agents to the neighbourhood, without degrading the performance in terms of response time. Figure 8 shows how the increase in response time remains controlled despite the introduction of new content and new agents in the neighbourhood.

7 Related Work

Autonomic computing is a new engineering paradigm that aims at building computing systems that are *self managing* [7]. Usually, self managing systems are expected to exhibit four main properties:

1. *self configuration*: the ability to configure itself according to high level goals;
2. *self optimisation*: the ability to optimise the use of resources;
3. *self healing*: the ability to *react* to the signs of a possible problem, by detecting it, and, if possible, fixing it;
4. *self protection*: the ability to defend itself from malicious attacks as well as from human error.

These characteristics remind of those defining the notion of *agency* and in [7] the authors claim that "autonomy, proactivity, and goal-directed interactivity with their environment are distinguishing characteristics of software agents [13]. Viewing autonomic elements as agents and autonomic systems as multiagent systems makes it clear that agent-oriented architectural concepts will be critically important". Hence, it is not surprising that many notions of autonomic computing are found in multi-agent systems (MAS) literature. An example is the use of an *hearbeat* message broadcasted regularly in a MAS, organised as in peers or as a network, in order to monitor the status of the other agents [14].

Self healing has been analysed in [15], where the authors present a team of broker agents, which share global knowledge about the system. This global knowledge is used to discover that a broker has been disconnected from the rest of the system and to inform the other brokers of the event. IBM has developed theABLE agent platform [16] that reduces the workload of the system administrator by supporting autonomic agents. Finally, in [17] provide a review of the various architectural issues in autonomic computing.

From the multi-agent literature perspecitve, SERSE can be classified among the cooperative information agents, such as RETSINA [18], and InfoSleuth [19]. RETSINA is a matchmaker based information system where collaborative task execution is achieved through matching service providers and requesters over the web (and more recently, over the Semantic Web). InfoSleuth explicitly deals and reconciles multiple ontologies by means of specialised ontology agents that collectively maintain a knowledge base of the different ontologies used to specify requests, and return ontology information as requested.

As mentioned in Section 1, P2P systems have been recently used to reduce the complexity of distributed knowledge management applications. A typical example of such an application is EDUTELLA [2], a hybrid P2P architecture for sharing metadata, that implements an RDF-based metadata infrastructure for JXTA [22]. However, the emphasis is more on RDF repositories of metadata rather than on the representation of semantic information in possibly heavy-weight ontologies. Some other projects use "super-peers", which start the semantic routing process in the right direction.

An aspect of peer-to-peer networks that needs to be especially analysed is *scalability*. The way in which queries are propagated in the network determines how the network itself will scale. Networks where queries are broadcasted to all peers will hardly scale, unlike those networks implementing intelligent mechanisms for broadcasting the queries only to those few selected peers that are able to answer the queries. At this end have been developed several routing protocols that manage distributed indices used to handle complex queries. Examples of such protocols are CAN [23] and Chord [24].

Other approaches emphasise the use of semantics represented in ontologies. Among these there is the SWAP project [25]. In SWAP, each node is responsible for a single ontology: ontologies might represent different views of a same domain, multiple domains with overlapping concepts, or might be obtained by partitioning an upper level ontology. Knowledge sharing is obtained through ontology mapping and alignment, however mappings are not dynamically obtained.

More recently, GridVine [28] support complex queries in RDQL, that consist of triple patterns with more than one bound variable, thus providing the possibility of asking sophisticated queries, and thus implementing scalable semantic overlay networks. However, GridVine does not deal with ontology management operations in each of its peers.

8 Conclusion

In this paper we presented SERSE – SEmantic Routing SystEm– a distributed multi-agent system composed of specialised agents that provides robust and efficient gathering and aggregation of digital content from diverse resources. The agents composing SERSE use ontological descriptions to search and retrieve semantically annotated knowledge sources, by maintaining a *semantic index* of the instances of the annotation ontology. The efficient retrieval is made it possible through the semantic routing mechanism, that permits to identify the agent indexing the resources requested by a user query without having to maintain a central index, and by reducing the number of messages broadcasted to the system. The system is also capable of exhibiting autonomic behaviour. Autonomic behaviour is characterised by self-management and self-healing capabilities, aimed at permitting the system to manage the failure of one of its agents and ensure continuous functioning.

We tested the performance search and retrieval capabilities of the system, and the experimental data shows that SERSE generates generally maintains the response times under the second, showing that the overhead produced by the indexing and routing mechanisms does not impact the system performance. We also tested the autonomic behaviour, and the experimental results show how the system is able to efficiently self configure.

Acknowledgements

The authors would like to thank Terry Payne for his comments on this paper.

References

1. Decker, S., *et al.*: The semantic web: The roles of XML and RDF. IEEE Internet Computing **4** (2000) 63–74
2. Nejdl, W., *et al.*: EDUTELLA: A p2p networking infrastructure based on rdf. In: Proceedings of the WWW2002, Honolulu, Hawaii, USA (2002) 604–615
3. The Morpheus website. (`http://musiccity.com`)
4. Halevy, A., *et al.*: Schema mediation in peer data management systems. In: Proceedings of the International Conference on Data Engineering (ICDE03), Bangalore, India (2003)
5. Bellifemine, F., *et al.*: JADE a white paper. EXP In search of innovation **3** (2003)
6. Tamma, V., Payne, T.: Toward semantic web agents: Agentlink and knowledge web. AgentLink newsletter **19** (2005)
7. Kephart, J., Chess, D.: The vision of autonomic computing. Computer magazine **36** (2003) 41–51

8. Tamma, V., *et al.*: SERSE: searching for semantic web content. In: Proceedings of ECAI 2004. (2004)
9. Tamma, V., *et al.*: SERSE: searching for digital content in esperonto. In: Proceedings of EKAW 2004. (2004)
10. RDQL (`http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`)
11. The Jena website. (`http://www.hpl.hp.com/semweb/jena2.htm`)
12. Ehrig, M., Staab, S.: QOM quick ontology mapping. Number 3298 in LNCS (2004) 683–697
13. Wooldridge, M., Jennings, N.: Intelligent agents: Theory and practice. Knowledge engineering review **10** (1995) 115–152
14. Sterritt, R., Bustard, D.: Towards an autonomic computing environment. Proceedings of 14th International Workshop on Database and Expert Systems Applications, 2003
15. Kumar, S., Cohen, P.: Towards a fault-tolerant multi-agent system architecture. In: Proceedings of Agents 2000.
16. Bigus, J.P., *et al.*: ABLE: A toolkit for building multiagent autonomic systems. IBM Systems Journal **41** (2002)
17. McCann, J., Huebscher, M.: Evaluation issues in autonomic computing. International Workshop on Agents and Autonomic Computing and Grid Enabled Virtual Organizations (AAC-GEVO04), Wuhan, China (2004)
18. Sycara, K., *et al.*.: Dynamic service matchmaking among agents in open information systems. ACM SIGMOD Record. Special Issue on semantic interoperability in global information systems (1998)
19. Bayardo, Jr., R., *et al.*: InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. Volume 26,2., New York, ACM Press (1997) 195–206
20. Ehrig, M., *et al.*: The SWAP data and metadata model for semantics-based peer-to-peer systems. In: Proceedings of MATES-2003. Number 2831 in LNAI, Springer (2003)
21. Castano, S., *et al.*: Ontology-addressable contents in p2p networks. In: Proc. of WWW'03 1st SemPGRID Workshop. (2003)
22. Project JXTA. (`http://www.jxta.org`)
23. Ratnasamy, *et al.*: A scalable, content addressable network. In: Proceedings of ACM SIGCOMM. (2001)
24. Stoica, I., *et al.*: Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM. (2001)
25. SWAP: Semantic web and peer-to-peer. (`http://swap.semanticweb.org`)
26. Arumugam, M., *et al.*: Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web. In: Proceedings of WWW2002, Honolulu, Hawaii, USA (2002)
27. Lima, T., *et al.*: Digital library services supporting information integration over the web. In: Proceedings of WIIW 2001. (2001)
28. Aberer, K., *et al.*: Gridvine: Building internet-scale semantic overlay networks. Number 3298 in LNCS (2004) 107–121