Peer-to-Peer Netw Appl
DOI 10.1007/s12083-009-0066-2

From P2P to reliable semantic P2P systems

Abdul-Rahman Mawlood-Yunis · Michael Weiss · Nicola Santoro

Received: 4 October 2008 / Accepted: 23 December 2009

Abstract Current research to harness the power of P2P networks involves building reliable Semantic Peer-to-Peer (SP2P) systems. SP2P systems combine two complementary technologies: P2P networking and ontologies. There are several types of SP2P systems with applications to knowledge management systems, databases, the Semantic Web, emergent semantics, web services, and information systems. *Correct semantic mapping* is fundamental for success of SP2P systems where semantic mapping refers to semantic relationship between concepts from different ontologies. Current research on SP2P systems has emphasized semantics at the cost of dealing with the traditional issues of P2P networks of reliability and scalability. As a result of their lack of resilience to temporary mapping faults, SP2P systems can suffer from disconnection failures. Disconnection failures arise when SP2P systems that use adaptive query routing methods treat temporary mapping faults as permanent mapping faults. This paper identifies the disconnection failure problem due to *temporary semantic mapping faults* and proposes an algorithm to resolve it. To identify the problem, we will use a simulation model of SP2P systems. The Fault-Tolerant Adaptive Query Routing (FTAQR) algorithm proposed to resolve the problem is an adaptation of the generous tit-for-tat method originally developed in evolutionary game theory. The paper demonstrates that the reliability of an SP2P system increases by using the algorithm.

Keywords Semantic P2P networks · Reliability · Overlay networks

1 Introduction

Peer-to-Peer computing is important for scale distributed systems and applications that require effective management of large-scale, distributed, and heterogeneous data. The importance of P2P computing is due to its characteristics: the decentralization of control, the autonomy and the dynamicity of peers, and the effective and transparent sharing of resources.

Data and resource descriptions held by peers in a P2P system lack explicit semantic. That is, data and resource descriptions are represented heterogeneously along different aspects. For example, data could be in XML[1] files, relational tables, text files, or RDF[2] documents. Even when the same type of representation format is used for storing information, the structure and semantics of concepts used in the modeling may vary among different peers. An example of semantic differences would be using different vocabularies to

A.-R. Mawlood-Yunis (✉) · N. Santoro
School of Computer Science, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6, Canada
e-mail: armyunis@scs.carleton.ca,
armyunis@connect.carleton.ca

N. Santoro
e-mail: santoro@scs.carleton.ca

M. Weiss
Department of Systems and Computer Engineering,
Carleton University, 1125 Colonel By Drive,
Ottawa, Ontario, K1S 5B6, Canada
e-mail: weiss@sce.carleton.ca

[1] http://www.w3.org/XML/.
[2] http://www.w3.org/RDF/.

Published online: 20 January 2010

refer to the same physical or conceptual object by different information representations (one's "zip code" is somebody else's "area code"), or using the same vocabulary to refer to different conceptual or physical real life objects in different representations: a "terminal" for one is a computer monitor, but a "station" for somebody else.

To effectively harness the power of P2P systems, heterogeneous information representation problem need to be overcame. Indeed, solving this problem becomes imperative to the success of information search and retrieval applications as well as for the success of the organizations that rely on them.

Current attempts to solve the problems pertaining to heterogeneity of data have focused on explicating the meaning of the information content, i.e., semantics augmentation. The backbone for exploring semantic based solutions to the information heterogeneity is Ontology, which is about defining a common conceptualization of the domain of interest plus a commitment of the involved parties to the conceptualization [16]. Using Ontology for modeling information resources or resource descriptions, concepts are defined in terms of their properties and relations to other concepts; concept definitions provided elsewhere on the Web or foreign peer repositories are reused using metadata; and new facts are inferred using the existing ones [15].

Despite some usefulness and existence of a number of common ontologies [14], the prominent difficulties with this type of work include problems of: adaptation (as common ontology undermines peers' autonomy), maintenance (as ontology domain concepts change or evolve over time), and, scalability and expressiveness (determining future growth of ontology and deciding on the appropriate level of detail of ontology description [38]). Further, in a dynamic, open and distributed environment such as P2P networks a common ontology solution is less feasible because having all peers commit to a common meaning is impracticable.

To overcome limitations associated with using common ontologies, Contextualization, or local ontologies, has been suggested by some authors [5, 6, 13] as an alternative strategy for modeling information sources. Following this paradigm, individual peers annotate their information sources with semantics in their own ontologies. These semantics are provider-specific, and reflect the information provider's knowledge of the application domain, experience, or culture. This implies a shift from large and centralized to small and distributed ontologies. Hence, the emergence of Semantic Peer-to-Peer (SP2P) systems. SP2P systems combine two complementary technologies: P2P networking and local ontologies. There are several types of SP2P systems with applications to knowledge management systems, databases, the Semantic Web, emergent semantics, web services, and information systems (see Table 1).

SP2P systems eliminate problems associated with the use of common ontologies (e.g., maintenance, scalability and adaptation problems, and the need for peers' commitment to common ontologies). Indeed, by shifting to SP2P system the door has opened for solving the problems pertaining to heterogeneity of data.

However, in this paradigm one needs to provide explicit *semantic mappings* (translations) among own local ontology and semantically related foreign ontologies to enable sharing of resources. Semantic mapping refers to defining semantic relationship between concepts from independent information sources (ontologies). Detailed description of SP2P components, including mapping, can be found in [30].

Current research on SP2P systems has emphasized semantics at the cost of dealing with the traditional issues of P2P networks of reliability and scalability. As a result of their lack of resilience to *temporary mapping faults*, SP2P systems can suffer from *disconnection failures*. A disconnection failure arises when an SP2P system that use adaptive query routing methods treats temporary mapping faults as permanent mapping faults.

To reduce the impact of disconnection failures, we need to make the query result evaluation function of an SP2P system fault-tolerant. The query result evaluation function must be able to tolerate temporary mapping faults, otherwise there is a risk that the network connectivity of the system will deteriorate, i.e. we need to improve the state of neighborhood connectedness of SP2P networks in the presence of semantic diversity and ontology changes.

This paper has two objectives: (1) to identify the disconnection failure problem, and (2) to propose an algorithm to resolve it. To identify the problem, we will use a simulation model of SP2P systems. We propose the Fault-Tolerant Adaptive Query Routing (FTAQR)

Table 1 Types of SP2P systems

SP2P Types	Systems
P2P Knowledge Management	KEx [5]
P2P Databases	coDB [12], Piazza [18], PeerDB [36], Hyperion [25]
P2P Semantic Web	Bibster [19], Somewhere [39], P2PSW [40]
P2P Emergent Semantics	Chatty Web [1], DisES [11]
P2P Information Systems	P2PSLN [17], OBSERVER [33], Edutella [37], P2PISM [42]
P2P Web Services	ESTEEM [4]

algorithm to resolve the problem. The algorithm is an adaptation of the generous tit-for-tat method originally developed in evolutionary game theory [2]. The paper demonstrates that the reliability of an SP2P system increases by using the algorithm.

To address the first objective, we built a generic simulation model of an SP2P system. The simulation is generic in the sense that it covers the core components and features common to existing SP2P systems such as [1, 5, 17, 18]. To increase the reliability of SP2P system we propose the Fault-Tolerant Adaptive Query Routing (FTAQR) algorithm. By evaluating query answers with the local ontologies, peers gradually decrease the confidences to their neighbors rather than directly dropping them if they replied incorrect answers.

The FTAQR algorithm is shown to be effective in eliminating the effect of temporary mapping faults, while detecting permanent faults. Further details of causes of mapping faults will be provided in Section 4. The simulation model is used to evaluate how using the FTAQR algorithm affects the reliability of an SP2P system.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the features of SP2P systems used in the development of the FTAQR algorithm. Section 4 identifies sources of semantic mapping faults related to ontology modification. Section 5 presents the FTAQR algorithm and its design. Section 6 presents the simulation model, while Section 7 shows the results of assessing the use of the FTAQR algorithm. Section 8 concludes the paper and identifies directions for future work. SP2P systems combine two complementary technologies: P2P networking and ontologies [15, 16].

2 Related work

The are two related literature streams: (1) SP2P systems, and (2) the maintenance of mappings in SP2P systems.

2.1 Review of SP2P systems

In this subsection we review six pioneering SP2P systems: Chatty Web [1], Bibster [19], KEx [5], OBSERVER [33], Piazza [18], and Edutella [35].

Chatty Web [1] describes a method for building a *common ontology* from local interactions between peers. A global semantic agreement is reached through the bottom-up construction of a common ontology. Each peer has its own ontology, which may be different from the ontologies of other peers. Using the XQuery language, a query imposed on one peer would be translated to a semantically equivalent query and imposed on different peers. Over the course of its lifetime, each peer will be able to find other related peers. As part of normal operation of the system — query translation and forwarding — the peers will be able to identify their semantically related peers. The lowest possible common knowledge among all peers of the network constitutes a shared conceptualization of the domain of discourse.

Chatty Web accounts for semantic mapping faults. It uses a simple probability function to determine the effect of mapping faults on the emergent semantics. However, it assumes that faults are equally distributed and independent from one another. Despite its novelty in explaining the effect of semantic mapping faults on the SP2P systems, we believe that the assumptions made in the design of Chatty Web are too broad; more importantly, not all faults are permanent.

Bibster [19] is one of the earliest P2P systems for sharing bibliography files among researchers. Peers in Bibster use a common shared ontology to model local files, query content and the expertise of peers, where a peer's expertise refer to its knowledge. The shared common ontology is the ACM topic hierarchy represented using the SWRC (Semantic Web for Research Communities) ontology. When a bibliographic entry is made available to the Bibster system, it will be automatically aligned with the SWRC ontology.

Peers advertise their expertise in the network and discover other peers whose knowledge may help them answer queries. Peer discovery is based on semantic matching between semantic content of the queries and the expertise models of the peers. Knowledge about the expertise of other peers forms a semantic network. Matching between query content and the expertise model is to rank peers; the ranking forms the basis for intelligent query routing in Bibster.

Using a common ontology in a P2P network jeopardizes the independence of peers, a core attribute of P2P networks. We believe that an appropriate approach to semantic knowledge sharing should not violate peer independence, and allow them to create their own local semantics. Local semantic mapping among peers with heterogeneous information representations is a more suitable approach for open information systems such as system based on P2P networks.

KEx [5] is an architecture for semantic knowledge management in a P2P network. KEx's core principle is that the heterogeneity of the knowledge represen-

tation should not be seen as an obstacle to knowledge management, but rather as an opportunity for promoting innovation. KEx facilitates building a community for sharing knowledge among a group of autonomous peers. The knowledge within the community is available to all peers and searchable. Each peer can either request information or provide information to other peers.

A document and a context repository is associated with each peer. The document repository is a place where structured document are saved. The context repository is a place where the semantics of concepts will be clarified when handling query requests and responses. Document repositories can store different types of knowledge, including references to experts, and links to other peers and external resources. The approach taken by KEx to knowledge management is an attempt to replace a centralized knowledge base with a set of distributed autonomous repositories.

OBSERVER [33] uses an approach where semantic relationships between ontologies are predefined. It tries to reduce the problem of having to know the semantics and structure of all information systems globally to one of defining synonym relations between the concepts used in different ontologies. Thus, the problem of searching a global information system is reduced to searching multiple ontologies. The synonym relationship between ontology concepts is determined by human experts and saved on an inter-ontology relation server. This server is used by the query processor component to locate semantically related information.

In addition to the maintenance and scalability problems associated with one or more inter-ontology relation server, the semantic relationship between concepts are declared manually. This is a serious disadvantage in comparison to systems that determine the relation between concepts from different ontologies dynamically. Further, while OBSERVER determines semantic information loss during query forwarding and translation [34], it does not resolve the problem.

Piazza [18] comprises an infrastructure and mapping language for semantic mapping and data management in a P2P environment. The system takes into account both domain and document structure. In Piazza, queries posed to one peer can be reformulated and posed to semantically related peers. The transitive closure of the translations among peers is used to answer queries. Piazza's contribution to data management is an important development toward moving to distributed semantic data management as opposed to the current practice of data integration and meditation. However, peers in Piazza system are static entities. They are not expected to leave the network. Thus, Piazza lacks the ad hoc property of open P2P networks. It is also difficult for new peers to join a Piazza network because of its rigid structure.

Mappings among Piazza peers' are carefully designed and created manually. In Piazza there is no explicit consideration for mapping faults. Piazza's investment in careful mapping design and creation could be seen as an implicit way for fault tolerance using an avoidance approach.

Edutella [35] enables sharing of learning resources among distributed and independent educational resource providers. Edutella uses the JXTA [21] protocol for its P2P network infrastructure, and RDF [23] to describe resources. Queries are routed using the JXTA group construct, that is, peers send queries to other peers in the same group. This is a very primitive form of semantic query routing. Further, query routing in Edutella is not adaptive. Once peer groups have formed, they continue to be used for broadcasting queries in the network. Control or ownership by peers over local resources has only been considered recently [22]. Edutella could benefit from the adaptability and fault-tolerance properties of the FTAQR algorithm described in this paper.

2.2 Maintenance of mappings in SP2P systems

Löser et al. [27] suggest that the information evaluation strategy used by a system is one of the criteria for distinguishing adaptive query routing approaches in semantic overlay networks. While we concur with the authors, we also present a new algorithm for query result evaluation.

Acknowledging that the topology of a SP2P network can change during query propagation, Zaihrayeu [42] highlights three different scenarios which have the potential for generating faults (transient faults). The author tries to transform the identified problems through a set of assumptions, another example of the avoidance approach. Our approach is resilient to the temporary unavailability problem highlighted by Zaihrayeu. More generally, our approach is resilient to the update problems associated with P2P networks and can detect the permanent faults. Peers with temporary faults remain connected, but permanently faulty peers will be abandoned.

McCann, et al. [29] describe the MAVERIC (Mapping Verification) system which continuously monitors information sources to automatically detect "broken" mappings. In their system, information sources are

probed periodically, and query answers are compared to a priori known answers. When newly retrieved query answers differ from the predicted answers, an alert about a potentially broken map is sent to the system administrator. this paper is relevant to our study but differs in approach. Instead of continuously monitoring information sources, we suggest that changes should only be detected when querying information sources, that is, changes that are current and relevant to our queries.

We concur with Colazzo and Stariani [10] in that corrupted mappings have drastic consequences on the query results. However, our solution to the problem is different from theirs. Our algorithm differentiates between permanent and transient mapping faults. We tolerate transient mapping faults and detect permanent ones. Our solution is to leave the resolution of permanent faults to system administrators. This is done in order not to reduce the algorithm's general applicability. In contrast to Colazzo and Stariani's approach, our solution is independent of the data representation and query language, and thus more broadly applicable.

3 From P2P to SP2P systems

Our close study of existing SP2P systems [1, 5, 11, 17, 19, 25, 33, 42] and related work on semantic P2P systems [7, 8, 20, 24, 28] allows us to identify points of differences to distinguish P2P from SP2P systems: (1) use of formally-structured information, (2) local mapping, (3) autonomous peer resource management, and (4) semantic-based routing. This list is not meant to be exhaustive.

Data or information in SP2P systems is *structured* and formal. The purpose of formally-structured data is to enrich data semantics and support inferences, which, in turn, improve search performance and search result quality.

Local mapping is used to translated between ontologies when forwarding queries between the peers. Peers can have different data schemas or knowledge representations.

Autonomous peer resource management focuses on how peers control resources without giving up their autonomy. In contrast to conventional P2P networks, resources in SP2P are neither replicated nor assigned to other peers in the network for the purpose of using them in the processing of queries. This is because the focus of SP2P systems is on applications where the replication of resources is not permitted [18, 25]. However, in semantic-based P2P file sharing systems this constraint may be relaxed.

Query routing in SP2P systems is different from non-semantic P2P systems. In SP2P, semantic-based peer selection relates peers with similar domain knowledge, and these relations are used in the query routing process. SP2P systems use semantic relations to forward queries rather than flooding, random walk or other non-semantic approaches.

4 Mapping corruption due to ontology modification

Improving the reliability of SP2P systems by tolerating semantic mapping faults is a core concern of this paper. Hence, describing scenarios which raise mapping faults in SP2P systems is important for understanding and designing semantic mapping fault-tolerance algorithms. In this section, we describe different situations in which different types of semantic mapping faults (corruption) occur as a result of ontology modification, when an old ontology is replaced with a new one. We use the term ontology modification for both ontology versioning and evolution, as both introduce modifications to the existing ontology. In the scenarios described, the emphasis is on two aspects: (1) the *extent* (or level) of ontology modification, and (2) modification update messages. We start by describing the forms of ontology modification.

4.1 Forms of ontology modification

Ontology modification occurs in various forms, including concept and datatype modification [26]. Some forms of ontology modification are based on concept modification:

1. *Adding new concepts* to existing ontologies. For example, adding a newly discovered class or type of drugs, proteins or diseases to existing relevant ontologies.
2. *Deleting concepts* from existing ontologies. When concepts are outdated, or no longer used, concepts may be deleted from the ontology structure.
3. A change in the *meaning* (or conceptualization) of existing concepts. The change can take the form of removing or adding concept relations or concept properties. An example of adding a new property is attaching a new hydrogen fuel type to the concept car. An example of removing a property is removing the disc drive property from the personal computer (PC) concept.

4.2 Mapping fault scenarios

A short description of the kinds of faults that can results from ontology modifications are listed below:

1. In circumstances where ontology modification is not a complete substitution to previous ontologies, it is possible for related peers or application to continue working. However, there are possibilities for *intermittent faults*. Intermittent faults occur when there are situations where related peers are unable to interpret the meaning of concepts in modified ontologies.
2. The *level* of ontology modification and whether or not the modified concepts will be *used* in the mapping process will determine the mapping result. The higher the level of modification and the repeated use of the modified concepts could give rise to inability of related applications or peers to work with the modified ontology results in a *permanent fault*.

The process of ontology modification can result in one of the following two situations:

1. *Unavailability* for short periods of time, if the access to the ontology is blocked while the modification is performed or,
2. A *race* situation between information source and information users, if the ontology user is informed about the change *before* or *after* the modification is made. Each of the two described situations will result in the *transient type of fault*.

For a detailed list and analysis of fault causes, as well as the relation between fault causes and fault types, we encourage the reader to consult our detailed account in [32].

5 Fault-tolerant adaptive query result algorithm

Query result evaluation strategy is an important aspect of adaptive query routing in SP2P systems. That is, for the SP2P systems to be reliable, they need to employ correct result evaluation function. Incorrect evaluation function prevent semantically related peer from teaming-up together. In this section we will describe Fault-Tolerant Adaptive Query Result (FTAQR) algorithm steps and procedures for solving the SP2P disconnection failure problem. The algorithm is simple in concept, easy to implement and highly effective. Several design decision have been made during algorithm development. These include:

A. *Use of local knowledge*. Peers have only knowledge of their immediate neighbors. A peer's knowledge is about its belief in the reliability or ability of their neighboring peers on providing correct answers to their queries. Reliability is defined in the range of $[0, 1]$, where 1 means that neighboring peers are able to return correct answers to a query and 0 means they are not. Peers disconnect from each other when the reliability reaches ≤ 0.
B. *Normalization is not applied*. Sending a query on an outgoing link could result in several query answers. The number of query answers depends on the number of cycles in the network starting from the querying peer. All answers are treated equally. That is, no extra weight is given to any particular answer or querying path.
C. *Use of average values*. The average value of query answers is used to evaluate the reliability of outgoing links.

These decisions are made to make the algorithm simple to be understood. Future revision of these decisions is possible. The algorithm is made up of three essential functions: (1) *initialization*, (2) *result evaluation*, and (3) an *update* function, and proceeds along the following steps:

1. At network startup, peers start *connections*. Connected peers set their trust value in each other to 1, and system parameters for query result evaluation are initialized.
2. The query result evaluation function verifies the $(\equiv, \supset, \subset, *, \perp)$ relations between concepts in the query answer and concepts in the querying peer's local ontology, that is, whether the semantic relationship between query concepts and a peer's local concepts are *exact same*, *related*, and *totally not related*. The result verification could also be interpreted as checks on whether the query response satisfies all, some or non of query constraints as well. For example, when a peer P sends a query with four concepts: $Q(A, B, C, D)$, and receives a response to with a similar number of concepts: $R(\bar{A}, \bar{B}, \bar{C}, \bar{D})$, if P has defined the relation between concepts (A, B, C, D) and $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ to be exactly the same $(\equiv)$, then the relationship between query and query response concepts is 100%.
3. Based on query result evaluation relation in step 2, peers update their confidence in the reliability of their outgoing links.

We have identified five different update policies: Complete, Firm, Partial, Benevolent, and Failure Guarded. The policies differ in two aspects: whether

or not peers consider partial answers to be faulty or correct, and how match peers tolerable toward faults. These policies could be described as follows:

The *Complete* policy implies that only exact answers are considered by the querying peers. The peers that are returning partial answers will be treated as if they were returning incorrect answers. Peers implementing this policy are not tolerable to faults, thus, whenever neighboring peers return incorrect or partial answers, their confidence values will be reduced by 1.

The *Firm* policy is similar to *complete* policy in accepting only exact answers. However, peers employing the *firm* policy are more tolerable toward partial answers. Peers will reduce the confidence in their outgoing links by values < 1 whenever they receive partial answers. The value that is used to deduct the confidence in the connected neighbor depends on the type of query answer, i.e. $\{\subset, \supset, *\}$.

The *Partial* policy is neutral (neither reward nor punish) toward neighboring peers when they return partial answers, but it reacts like the *Complete* and the *Firm* policies toward the incorrect answers, i.e., it reduces the confidence value of outgoing neighbors by -1.

The *Benevolent* policy is a fault-tolerant policy. It accepts partial results and follows the generosity tit-for-tat approach in dealing with incorrect answers. For a neighboring peer to be disconnected from a querying peer, it has to return five incorrect answers in sequence. That is, querying peers are generous in dealing with faults and tolerate up to four faults in sequence. after that is follows the tit-for-tat policy. The number of the faults that the policy could handle is system variable, i.e. the value of the variable is set by the system users according to their needs and application purpose.

The *Failure Guarded* policy is similar to the *Benevolent* policy in dealing with partial results and tolerating faults. Further, *Failure Guarded* approach provide peers with an additional capability which prevent them to reach to a state of total isolation and prevent network from disconnection. This is reached by changing the result evaluation function: peers check their outgoing connection degree prior to cut ties with their neighbors. That is, when peers have only one outgoing neighbor, they will not update their confidence value in their neighbor even if they receive incorrect query answers from them. The continually connection policy is not achieved freely. Peers employing this strategy have to accept high rate of faults for the sake of the connectivity.

The numerical values that are used for updating confidence value in outgoing links for all five policy are presented in Table 2. These values are system parameters, their values are set by system administrator in such away that application needs could be met in the best possible way.

Table 2 The numerical values used by the studied update policies

Update policies	Values
Complete	[1, −1, −1, −1, −1]
Firm	[1, −0.1, −0.1, −0.05, −0.2]
Partial	[1, 0, 0, 0, −1]
Benevolent	[0.2, 0.15, 0.1, 0.05, −0.2]
Failure Guarded	[0.2, 0.15, 0.1, 0.05, −0.2]

6 SP2P system simulation

In this section, we describe the SP2P simulation system that has been built to identify the disconnection failure problem in the SP2P systems as well as to demonstrate our solution. The simulation is an event based simulation, and its control loop execution represents a single event. The simulation is written in Java language and uses several open source packages. These include Repast,[3] Protege,[4] Jena,[5] and JavaFreeChart.[6] The simulation is been built on top of Repast, a multi-agent simulation framework. The simulation uses the Repast's event scheduling, network components, and visual display.

The simulation comes in two different implementations. The first one is a Gui based implementation where the user can see the simulation progress while it is running, and the second one is a batch based implementation. In the latter version, the simulation executes all the scheduled tasks and reports the result into files specified during simulation configuration. Different from the first implementation, all the figures and statistical results in the report are produced using a JFreeChart open source package version 1.0.12, not Repast's visual display. The batch implementation should be used when the effect of several simulation parameters need to be determined and compared on the same chart. The simulation code and the simulation's class hierarchy document are available upon request.

The simulation is made of seven key constructs. The simulation constructs are Peers, Resources, Query Formulator, Semantic Neighborhood, Mapping, Router, and Query Answerer. In the following we will briefly describe each of these simulation constructs.

[3] http://repast.sourceforge.net/repast_3/index.html.

[4] http://protege.stanford.edu/plugins/owl/api/.

[5] http://jena.sourceforge.net/.

[6] http://www.jfree.org/jfreechart/.

6.1 Peers

Peers are unique entities in the simulation. Each peer manages an ontology. Any number of peers could be simulated. There is no limitation on the number of peers that can be simulated other than the computational capacity of the machine(s) that is running the simulation.

6.2 Resources

The simulation runs with the synthetic/actual data. For the simulation to function it requires a number of ontologies which is equivalent to the number of peers in the network, each peer owns a resource. Since number of peers in the network is undetermined ahead, the network size is simulation parameter, we come up with three different methods to render new ontologies from existing ones. The methods are different from each other based on whether one or more ontologies exist for rendering new ones, and whether concepts selected for creation of new ontologies are selected randomly or systematically. For our experiments, four different ontologies were initially created for electronic device domain. For each experiment, then an enough numbers of ontologies were rendered from these initial ontologies, and each newly rendered ontology is assigned to a peer. The four initial ontologies were created by students from School of Computer Science at Carleton University, they are stored at http://www.scs.carleton.ca/~armyunis/Owl_File.

6.3 Query formulator

In the SP2P system simulation, queries can have up to k concepts. The k value for a peer P is constrained by the total number of concepts exist in the peer's local ontology Z. that is, $k \leq Z$. Query concepts are chosen from local ontologies, and any peer can initiate a query. Peers that initiate queries as well as query concepts are selected randomly. At each simulation run a new query is created and a new initiator is selected. Chances for the query concepts and query initiators to be different than prior concepts and initiators are high. The probability of a peer to be a query initiator at each simulation run is $1/N$, where N is the number of peers of the simulation. The probability of the query to be exactly the same query as the previous one is $(1/D)^{\|f\|}$, where D is the size of local ontology each peer possesses, and $\|f\|$ is number of concepts that comprise the query content. The probability of the same peer to pose the same query in two different system runs is then the multiplication of both above terms, i.e., $(1/N)(1/D)^{\|f\|}$.

Algorithm 1 shows pseudo code for the described steps. The *resouceConcepts* in Algorithm 1 is an array. To facilitate query construction, peers' ontologies concepts are extracted into arrays. When a query is created, query concepts are selected from querier peer's local array. Updating peers' ontologies change ontologies and their corresponding arrays. Thus, resouceConcepts changes in each simulation run.

6.4 Semantic neighborhood

To connect peers with similar domain knowledge, the simulation employs the discovery method similar to the one employed by the JXTA [21]. On network startup, peers exchange their profile, subsets of peers' local ontologies concepts, and a tree comparison algorithm is used to determine the similarity relation between the profiles. When a new peer joins the network, it will broadcast its profile, and peers with similar domain knowledge representation will reply to the advertisement and connection establishes. In addition to the similarity function (*sim*) which has to be greater than a predefined threshold (e.g. sim $\geq \delta$), peer connections are also restricted by the number of connections, d that each peer can have. The value of d is user defined variable, and get set on simulation startup. Algorithm 2 shows pseudo code of the described steps.

6.5 Router

The adaptive query routing strategy is employed in the simulation. Peers use the semantic neighborhood initially created for search and content retrieval to send queries, and update their neighbors following the update policy applied by the Query Answerer component. Query cycling is prevented using query path information, and queries are stopped from forwarding when peers have no more semantic link to traverse. We do not apply any forwarding policy, e.g. TTL policy,

Algorithm 1 Query formation steps

```
Set queryConcepts ← {}
Query q ← null
int i ← queryConceptsSize
while (i < MaxnumberOfConceptsInQuery
      and i < localResouceSize)
      add (queryConcepts, resouceConcepts[i])
      i ← i+1
end while
q ← new Query (queryConcepts)
```

Algorithm 2 Semantic neighborhood formation function

```
/* compare, a tree comparison function call */
Set sourceAndTarget ← new HashSet()
for(int i ← 0 To i < numberOfNodes)
   for (int j ← i To j < numberOfNodes)
        /* SchemaCollection contains all schemas */
        compare (sourceAndTarget, SchemaCollection[i],
        j ← j+1
   end for
   i ← i+1
end for
/* set of function calls to rank neighboring peers and create a
   semantic neighborhood */
compare (Collection sourceAndTarget, list peerProfile)
connectionMap (sourceAndTarget)
sortConnectionMap (connections, rows, columns)
sortByStrength (sortedMapById)
semanticNeighborhood (getGroups())
```

to stop query forwarding because in the current setting of the simulation it would not have made match difference. Algorithm 3 is pseudo code for the route function executed by the Router.

Algorithm 3 Route function

```
SEND(Query query, AnEdge edge, Vector UsedLinks,
        Peer initiator)
Set <Peer> neighbors ← getOutNodes ()
AnEdge anEdge ←null
Peer peer ← null
if (neighbors ≠ {} and QueryContent ≠ {})
   while (MoreNeighborsLeft)
         peer ← getNeighbor ()  // Next peer
         Set <AnEdge>edges ← getEdgesTo(peer)
         anEdge= edges [0] //next edge
         if (!(UsedLinks.contains(anEdge)))
           UsedLinks.add(anEdge)
           peer.SEND(query, edge, UsedLinks, initiator)
         end if
   end while
   setCycle()
else
   getCycle().add(initiator)
   answerHandler(initiator, query, edge)
end if
```

6.6 Mapping

In the current version of the simulation the mapping component is not developed. This is because simulating mapping faults were enough for the purpose of the experiments. However, the simulation allows for the seamless integration of the existing mapping algorithms or thesaurus [9, 41] to perform query and query result translations. Mapping (or query translation) can be carried out either by peers sending queries or peers receiving queries. System designers need to decide whether the sender or the receiver perform these translations or mappings. We assume queries as well as query results, which are merely service descriptions, to be of limited size. Similar assumptions are made by others, for example Chatty Web and Edutella [1, 37] authors. Under these circumstances, query translations and query result processing require too little computational power to cause any undesirable results. This claim has been validated through running SP2P simulation with a new set of tests where number of peers vary from 50 to 150.[7]

6.7 Query answerer

Query Answerer QA component executes two main functionalities: 1. update the network, and 2. apply the fault-tolerant algorithm. The update functionality comprise of two procedures: A) removing faulty links, and B) changing the strength of the links as they participate in query answers. In the simulation, peers send their answers directly to the querying peer, and QA component of the querying peer process the result in two steps: i. automatically determine if the answer is satisfiable, and then 2. apply the fault-tolerant policy. The definition of the satisfiability depends on the applied fault-tolerant policy, and five different fault-tolerant policy have been proposed in Section 5.

Sending query responses directly to the querying peer leads to one think of some interesting and reasonable concerns. For example, if the responses sets are large, the receiver may become unresponsive for a while. However, since query responses are merely services descriptions of limited size, i.e. metadata of the actual resources, this scenario is most unlikely. Furthermore, individual peers should only send as many queries as they can manage without causing any reduction to their ability to operate properly. Algorithm 4 represents answer handling steps and procedures.

[7]The virtual memory size of the used machine has restricted us from using larger number of the peers.

Algorithm 4 Answer handling function

```
int count ← 0
int nocq ← query.getQueryOrigin().size()
Set <String>temp ← query.getQueryContent()
if (temp.size() > 0) {
  for (int i =0 To i < getQueryContentSize)
      if ((initiator.getResouce()).contains
        (temp.elementAt(i)))
        count ← count + 1
        i ← i + 1
      end if
  end for
end if

/* ChangeInStrength value represents a fault-tolerance
   policy in Table 2 */
if (count = nocq)
  { Update(anEdge, ChangeInStrength[0]) }
else if (count = (nocq -1))
       { Update(anEdge, ChangeInStrength[1]) }
else if (count = (nocq -2) )
       { Update(anEdge, ChangeInStrength[2]) }
else if (count = (nocq -3))
       { Update(anEdge, ChangeInStrength[3]) }
else { Update(anEdge, ChangeInStrength[4]) }

setEdgeColorStrength (anEdge)
removeUnusedLinks (initiator, anEdge)

Update(AnEdge anEdge, double nstrength ) {
     double newStrength ← 0
     newStrength ← anEdge.getStrength() + nstrength
     anEdge.setStrength(newStrength)
}
```

7 Reliability assessment tests

In this section, we describe fault simulation and the experimental results which are carried out in order to demonstrate the effectiveness of FTAQR and its various policies. The SP2P system reliability is measured through network connectivity, network component count. The effectiveness of a fault-tolerant policy is measured using the number of queries executed which is also number of the simulation ticks or runs (nrs), against network components. A highly connected network, low number of network components, is reported as a reliable network, and a network with high number of components, disconnected and fragmented network, is perceived as unreliable network.

The results are for two types of experimental settings: the SP2P system with fault-tolerance (benevolent and failure guarded test results); and the systems without fault-tolerant capabilities (complete, firm, and partial test results).

Faults are generated using the knowledge about the peers' different ontology sizes, query formulation and routing strategy. When a query is created with a set of peer local concepts and that query is passed through other peers that have only a subset of query concept constituents, the end query result will contain only the minimum common denominator of concepts. The difference between the number of concepts in the answer and the number of concepts in the query symbolizes the fault. This is the same strategy applied by Chatty Web [1] to stop forwarding queries.

7.1 Complete semantic relation maintenance

When only exact answers are considered by the querying peers, peers that are returning partial answers will be treated as if they were returning incorrect answers. Peers implementing this policy are not tolerable toward faults, thus, whenever neighboring peers return incorrect or partial answers, their confidence values will be reduced by 1.

Figures 1, 2, and 3 show that the network deterioration trends are almost identical for cases when the

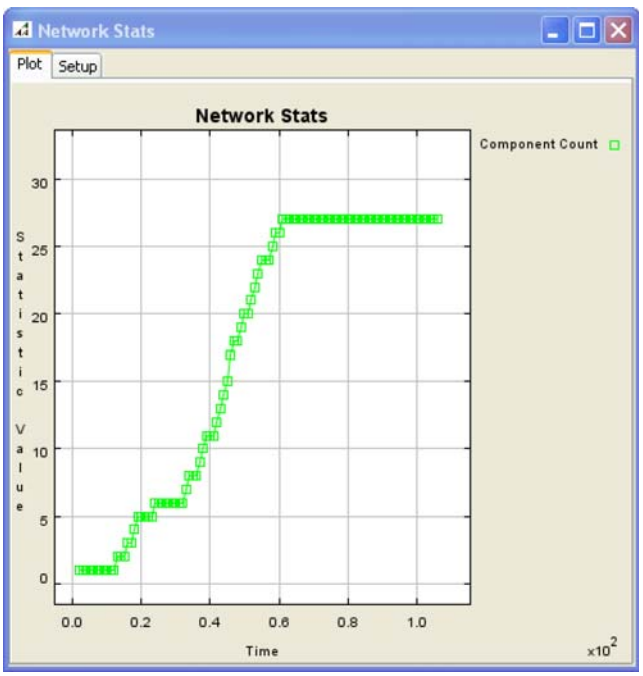

Fig. 1 Network deterioration under a complete policy when answer is partially asserted (⊂)

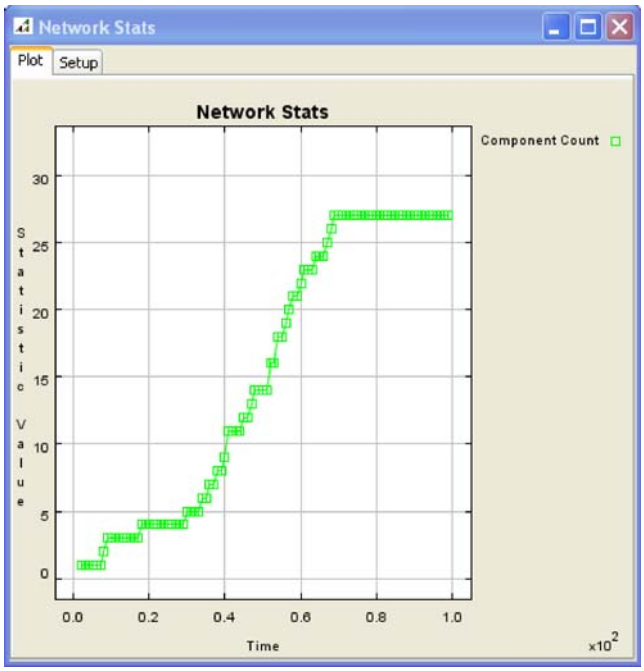

Fig. 2 Network deterioration under a complete policy when answer is related to the request ($*$)

semantic relation between query concepts and peer's local concepts are $\subset, *$, and $\perp$. It will take close to $0.65 * 10^2$ queries for the network to reach a total disconnection state, and it is almost same for all three cases.

Figure 4 on the other hand, shows that when there is no fault in the system, peers with complete semantic agreement, i.e. peers have identical ontologies, remain connected. Network deterioration stops after peers dropping their links to less compatible neighbors. Figure 5 depicts the network stability after excluding partially compatible peers.

7.2 Firm semantic relation maintenance

Figures 6, 7, and 8 show that the trend of the network deterioration is similar to the complete semantic maintenance case (Subsection 7.1). One difference between this situation and the previous one is that, the number of queries executed for the firm case is larger ($> 2 * 10^2$ nsr) than the previous case. This is due to the firm maintenance policy being more tolerable toward partial answers than the Complete maintenance policy. Peers will reduce the confidence in their outgoing links by values < 1 whenever they receive partial answers.

Another difference between this case and the previous one is the noticeable difference in the number of executed queries among the three type of answers $\{\subset, *, \perp\}$ before the networks reach to the failure state. The numbers are $2.75 * 10^2$, $1.8 * 10^2$, and $0.8 * 10^2$ respectively. This implies that having a network where

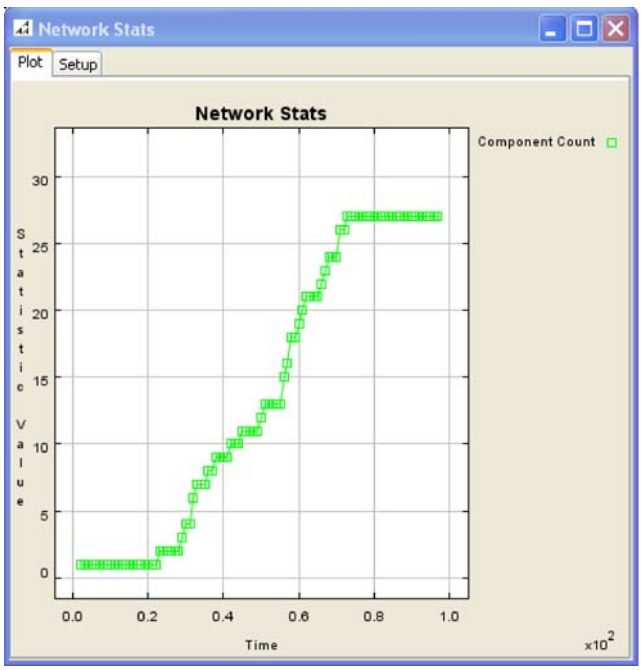

Fig. 3 Network deterioration under a complete policy when answer is completely faulty ($\perp$)

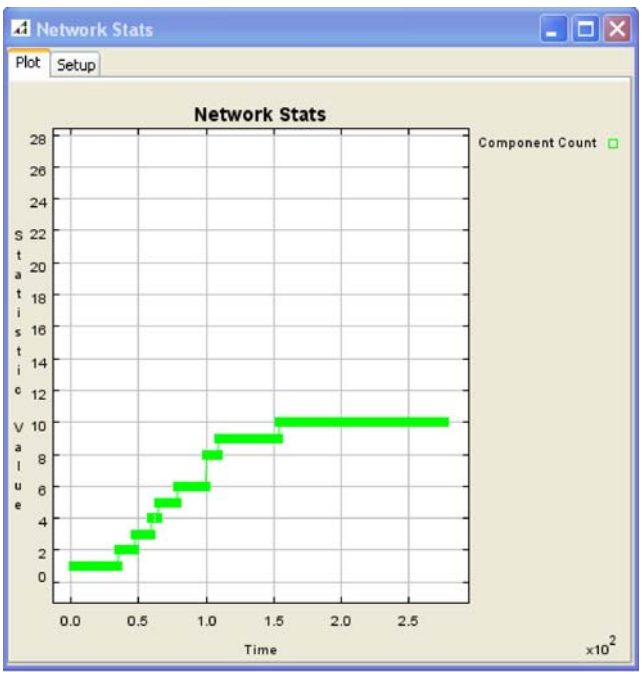

Fig. 4 Network in stable state under a complete policy after excluding incompatible peers

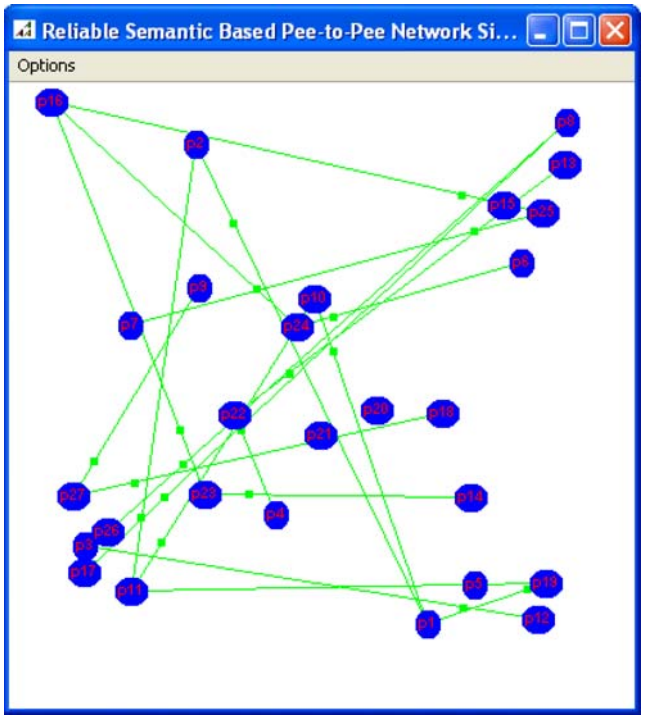

Fig. 5 Compatible peers stay connected under a complete policy

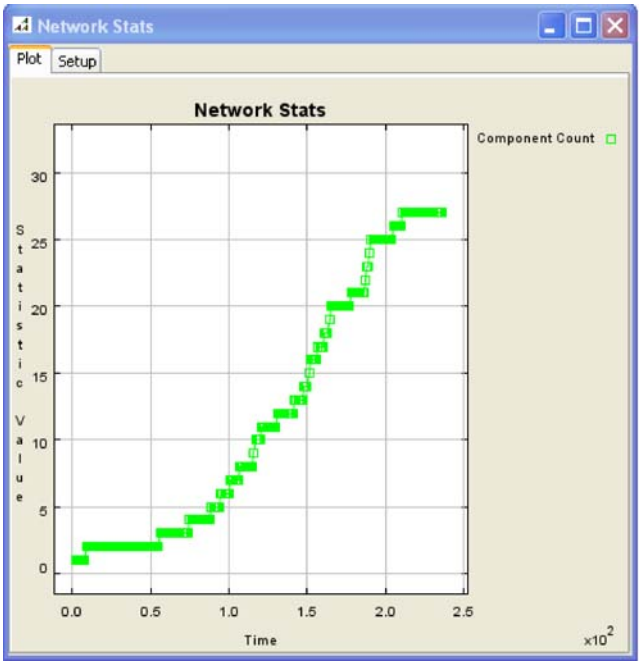

Fig. 7 Network deterioration under a firm policy when answer is related to the request (∗)

peers are able to return partial answers ⊂, the number of executed queries will be greater than the number of executed queries for a network in which peers return only related answers ∗. The number of executed queries for the latter case will be greater than the network in which peers return only incorrect answers, ⊥ answer. As a matter of fact, the last case is the same as the Complete answer maintenance, because the query answer contains no partial result of any type.

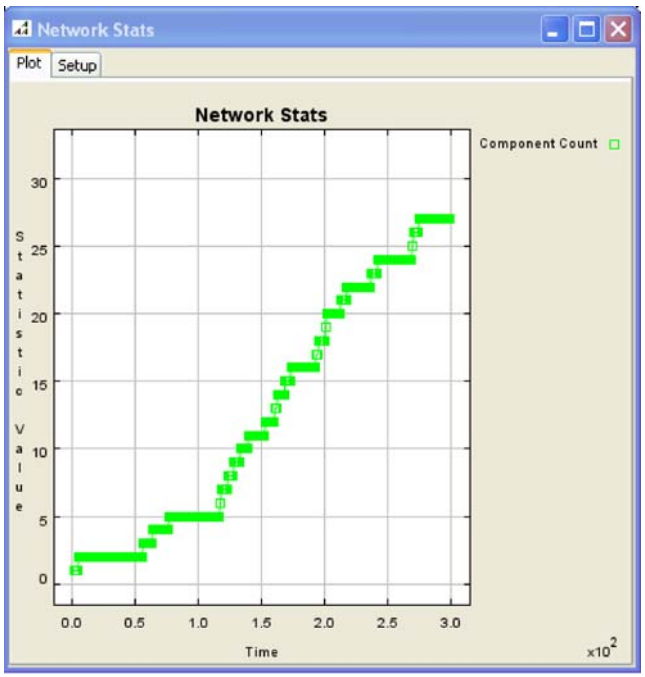

Fig. 6 Network deterioration under a firm policy when answer is partially asserted (⊂)

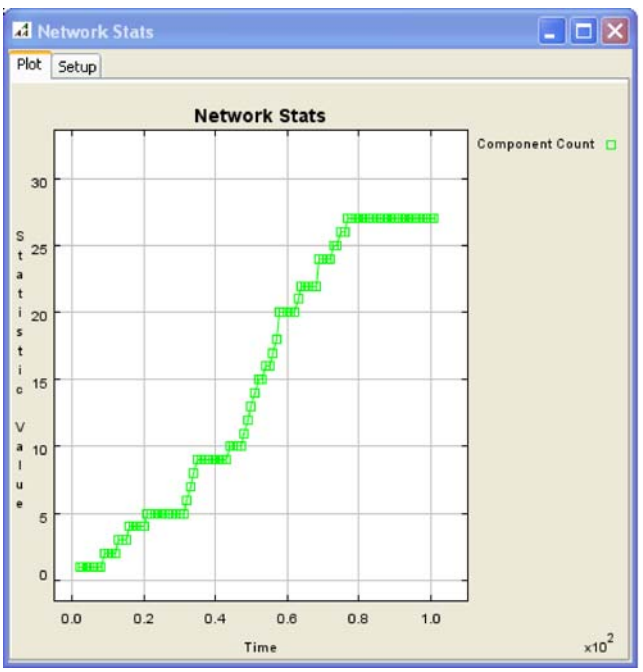

Fig. 8 Network deterioration under a firm policy when answer is completely faulty (⊥)

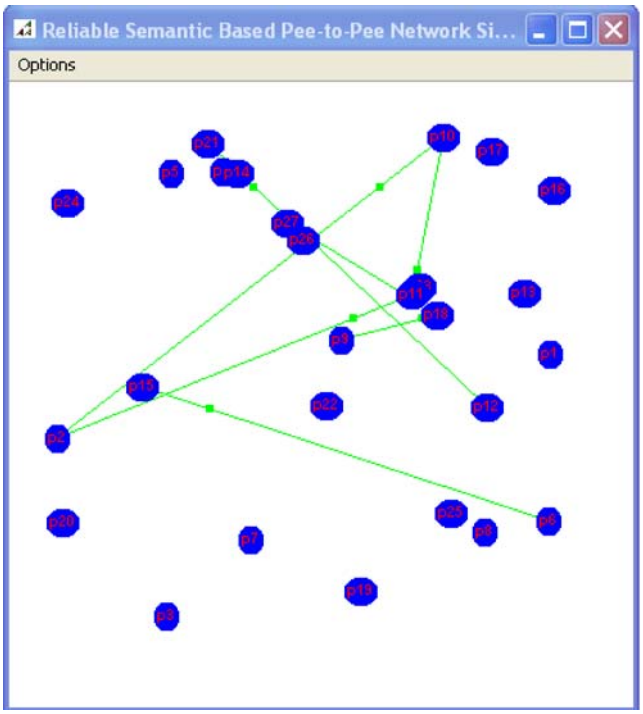

Fig. 9 Compatible peers stay connected under a firm policy

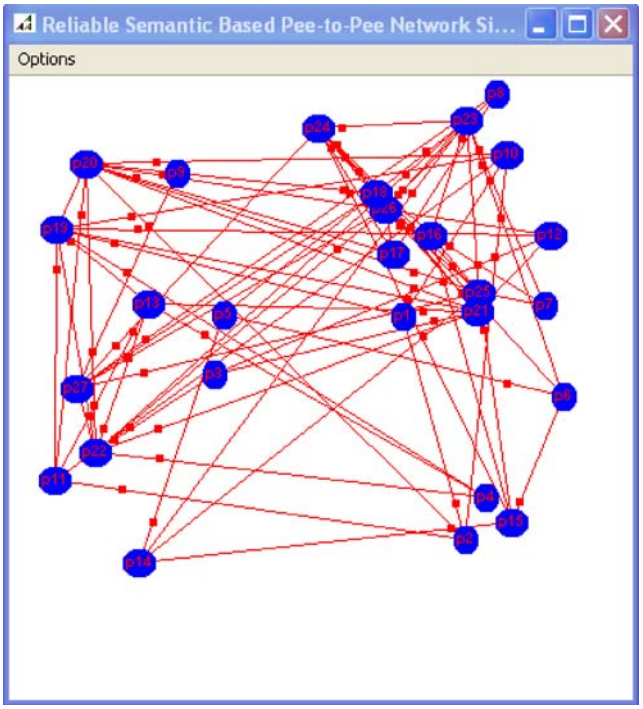

Fig. 11 Initial SP2P network under a partial policy

Figure 9 on the other hand, shows when there is no fault in the system, peers with complete semantic agreement remain connected, and network deterioration stops after peers drop links to their less compatible neighbors. Figure 10 depicts the network stability after excluding partially compatible peers.

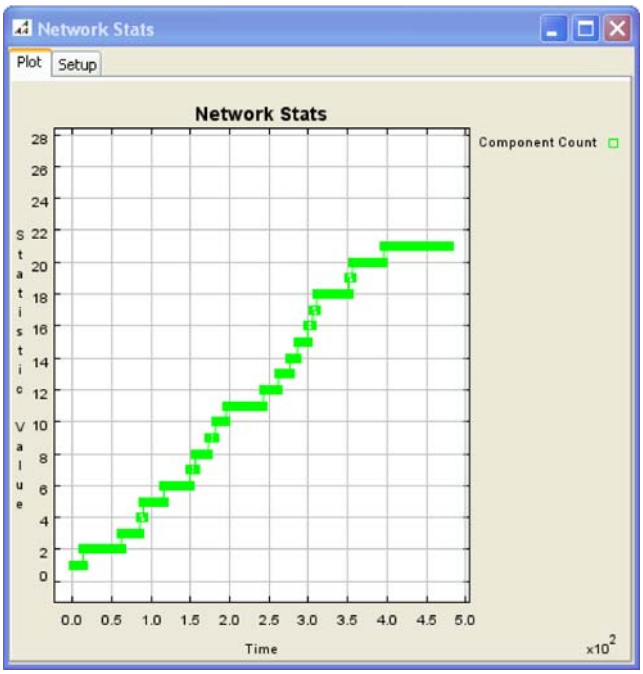

Fig. 10 Network in stable state under a firm policy after excluding incompatible peers

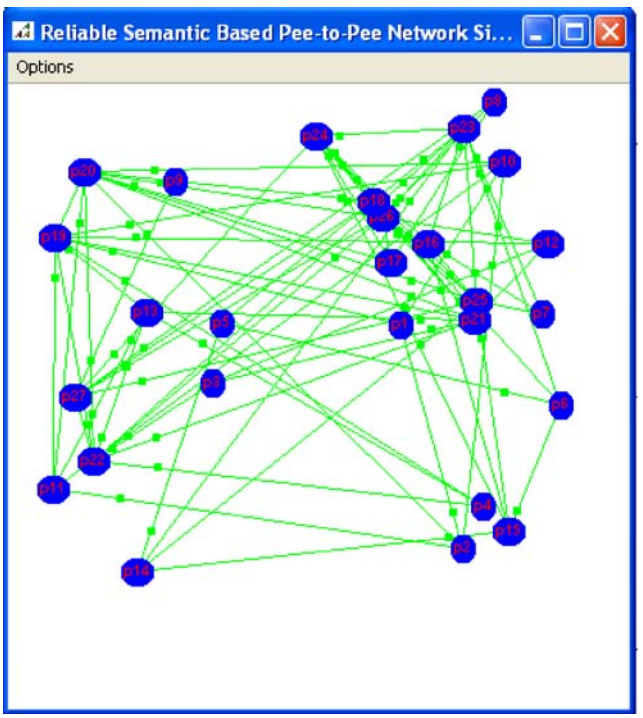

Fig. 12 Peer connections unchanged from initial setting under a partial policy

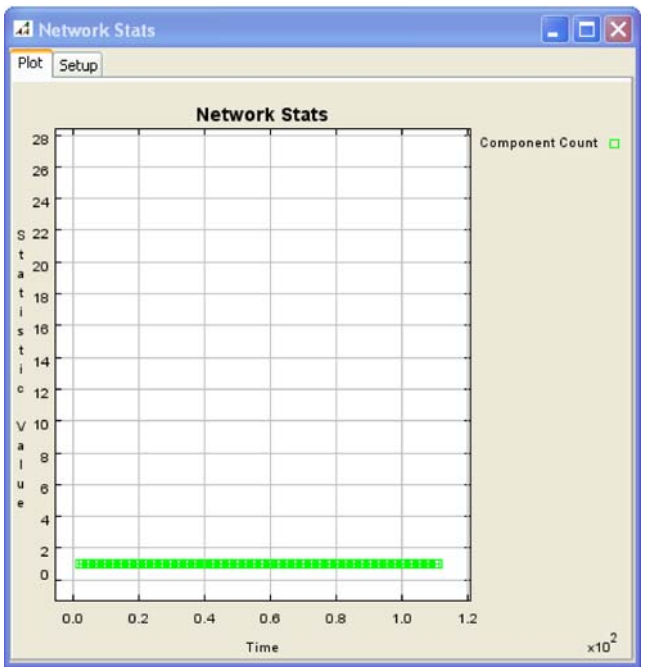

Fig. 13 Network in a stable state under a partial policy

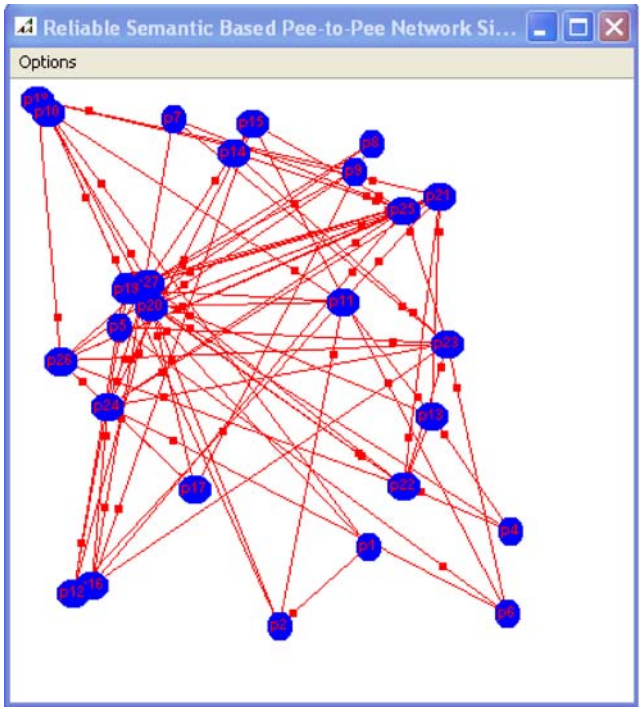

Fig. 15 Initial SP2P network under a Benevolent policy

7.3 Partial semantic relation maintenance

Figures 11 and 12 show that when SP2P peers return only partial answers, but not faulty answers, the network stays connected, and Fig. 13 indicates that there is no sign of the network deterioration. Furthermore, the number of executed queries for a situation when the combination of faulty and partial results are returned is higher, $(4 * 10^2)$, than the previous two cases. This

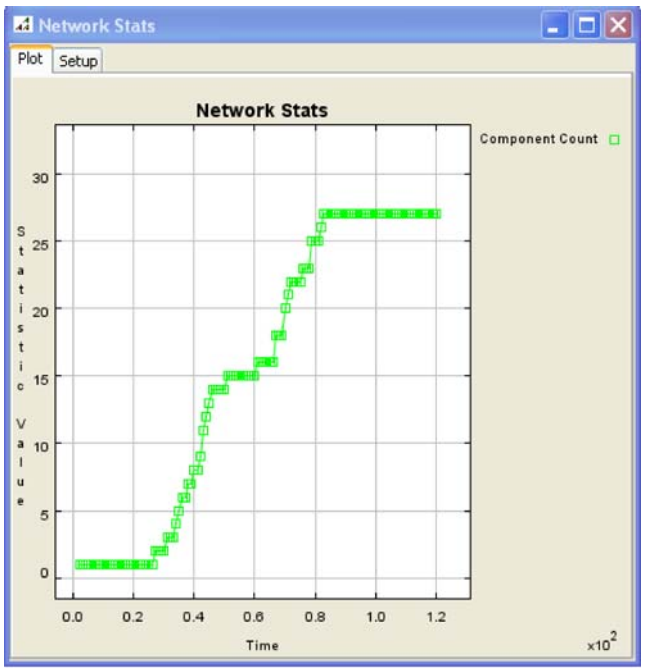

Fig. 14 Network deterioration under a partial policy when answer is faulty

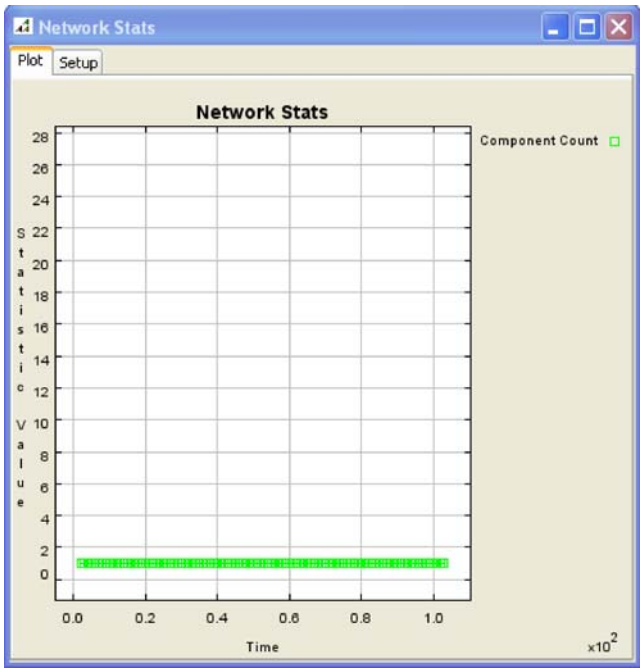

Fig. 16 Peer connections unchanged from initial setting under a Benevolent policy

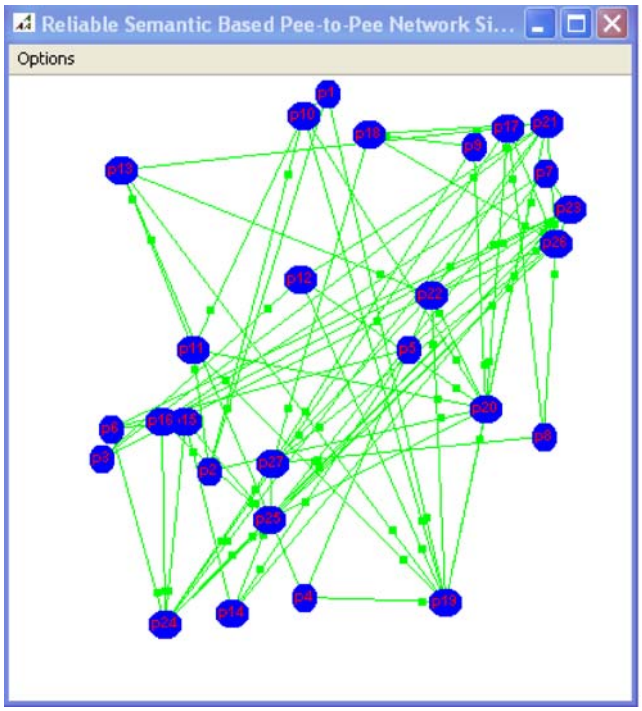

Fig. 17 Network in stable state under a Benevolent policy

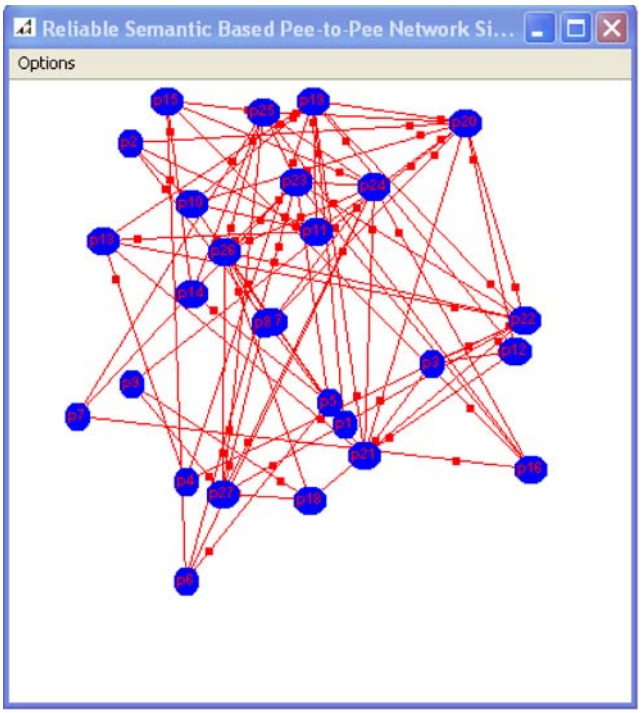

Fig. 19 Initial SP2P network under a failure guarded policy

is because the partial query answers have no negative impact on the confidence peers have in their outgoing neighbors.

Figure 14 shows that when the peers return faulty answers, the network deteriorates quickly, and partial maintenance answers becomes close to Complete se-

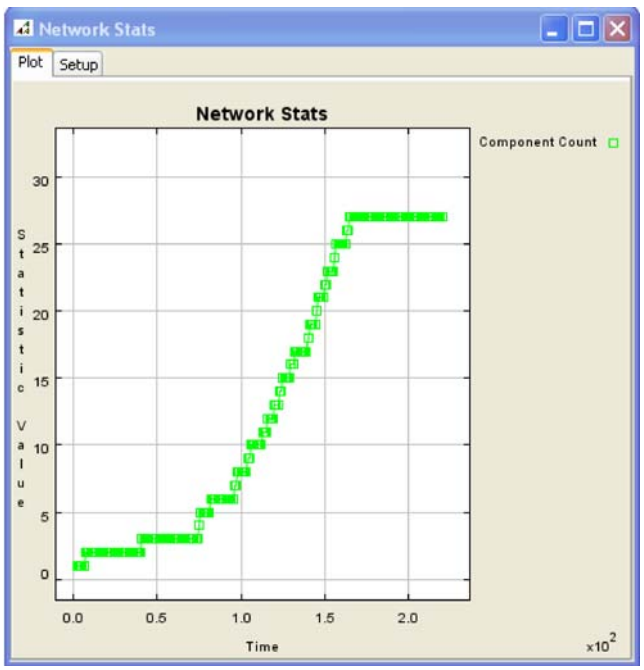

Fig. 18 Network deterioration under a Benevolent policy when answers are faulty

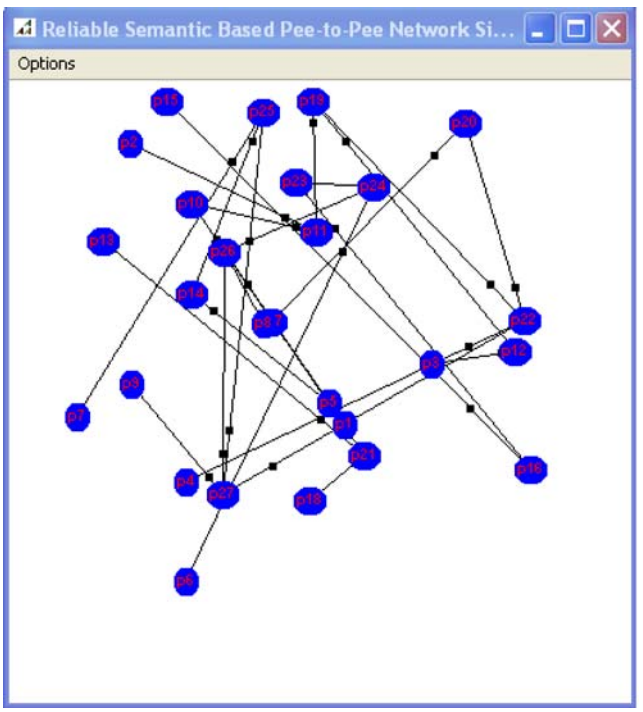

Fig. 20 Peers always stay connected under a failure guarded policy

mantic maintenance case. This because, partial relation maintenance does not differentiate between different types of faults, and treats all faults as permanent one (see [31]). Chatty system [1] follow the partial answer maintenance approach, hence the Chatty Web is not a temporary fault-tolerant system. Chatty Web could benefit from the fault-tolerant methods that will be introduced in the following.

7.4 Benevolent semantic relation maintenance

Similar to the partial answer maintenance method, Figs. 15 and 16 show that when the SP2P peers return only partial answers, but not faulty answers, the network stays connected, and Fig. 17 indicates that there is no sign of the network deterioration.

However, Fig. 18 shows that the difference between partial answer maintenance and benevolent answer maintenance arise when peers return faulty query answers. The number of executed queries of benevolent semantic maintenance is larger than partial semantic maintenance ($1.6 * 10^2$ versus $0.8 * 10^2$ nsr) due to the fact that in the latter case peers account for temporary faults.

The Piazza System [18] accounts for partial answers and supports partial result integration. However, for the benevolent answer maintenance to be fully integrated with Piazza system, Piazza needs to utilize the dynamic property of P2P networks. Currently, Peer connection establishment in Piazza is static and expensive task.

Employing the benevolent policy instead of the partial policy currently used by Chatty Web [1], the systems fault-tolerance capability will be improved substantially. That is, the system would be able to process up to $1.8 * 10^2$ different queries instead of $0.8 * 10^2$ queries before it become totally disconnect.

7.5 Failure guarded relation

Figures 19 and 20 show that when peers apply the faulty guarded strategy to prevent isolation, they will stay connected even though they might have less connections. Figure 21 shows that the network will stay in totally connected state.

7.6 Network size

Even though the main focus of this paper is on the description of a fault-tolerant algorithm and the impact of different answer handling policies on the reliability of SP2P networks, determining the effect of several components of the system on network behavior are of interest to system designers. These include studying the effect of the number of peers, number of links a peer manages, and fault rate.

In this paper, we carried out two tests to determine the effect of the number of peers on network behavior. In one test we use a non-fault-tolerant policy (Complete), in the other a fault-tolerant policy (Benevolent). The number of peers in both experiments were 50, 100, and 150, respectively. Figures 22 and 23 show the results. They indicate that except for scale, network behavior does not change with an increase in the number of peers in both cases, and the fault-tolerant policy is

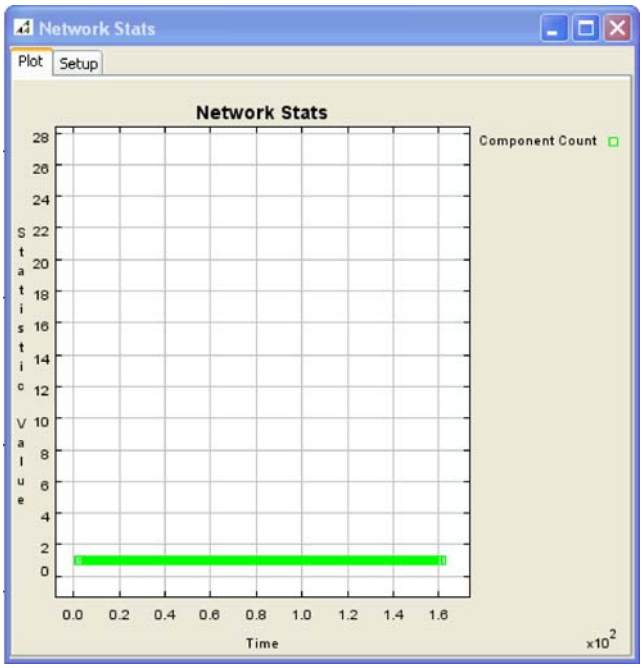

Fig. 21 Network stays connected under a failure guarded policy

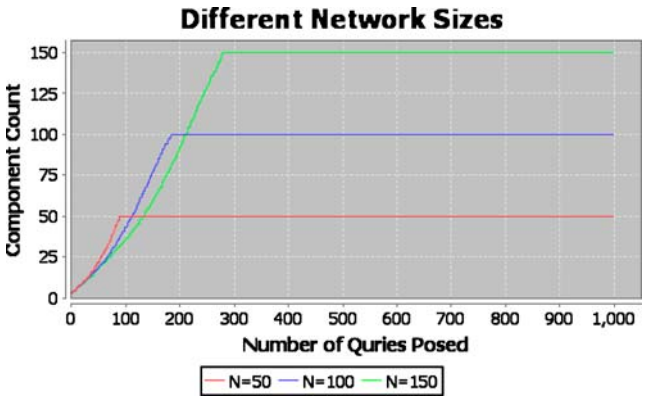

Fig. 22 Network behavior for a non-fault-tolerant policy with different networks sizes

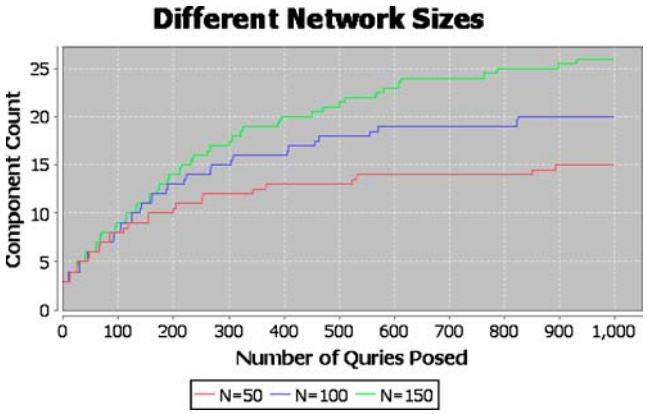

Fig. 23 Network behavior for a fault-tolerant policy with different networks sizes

effective in improving network reliability regardless of the number of peers used in the network.

Generally, we believe that the number of query answer responses are constrained by the number of correct answers in the network and the time to live (TTL). The latter limits the number of peers contacted during query propagation, as demonstrated by the test results. In each test, 1000 queries are executed per simulation run, and the experiments are repeated 30 times. The test results report the median values. The reader may notice that the background of Figs. 22 and 23 are different than the rest of figure backgrounds in the paper. This is because, the simulation comes in two different implementations. Figures 22 and 23 were produced using the batch based simulation, and the rest of figures are snapshots of running gui based simulation (see Section 6 for more information).

8 Conclusion and future work

In order to use the scalability and the computational power of P2P systems in new applications and domains, existing P2P infrastructures need to be improved in two different ways: (1) P2P systems need to be semantically enhanced, and (2) P2P systems need to be reliable. In this paper we have touched on both of these aspects. We have identified features that differentiate existing P2P systems from semantic P2P systems, and demonstrated the disconnection failure problem due to mapping problems in SP2P systems. An algorithm were provided to solve the disconnection problem as well. The test results showed that the proposed algorithm is effective. The simulation results demonstrated that a Failure Guarded policy prevents individual peers in SP2P systems from isolation, and guards SP2P systems from collapse. A Benevolent policy, on the other hand, prohibits peer disconnection as a result of temporary faults.

Using our simulation model, we were able to examine the impact of our algorithm on the query routing method of Chatty Web [1]. In Sections 7.3 and 7.4, we demonstrated that the reliability of Chatty Web [1] could be improved substantially by tolerating non-permanent faults. In Section 7.4, we have identified how Piazza [18] could benefit from incorporating the FTAQR algorithm. Checking on recurrent incorrect query answers from semantically relevant peers in P2PSLN [17] could be further improved by tolerating non-permanent faults using our algorithm as well.

As a future work, we are working on further improving the current algorithm. The algorithm might be improved through using a combination of majority voting techniques and fault-tolerant polices . This could help to achieve a more precise query result evaluation before any decision be made about changing peers' confidence in their outgoing links.

References

1. Aberer K, Cudre-Mauroux P, Hauswirth M (2003) Start making sense: The Chatty Web approach for global semantic agreements. J Web Sem 1(1):89–114
2. Axelrod R (1997) The complexity of cooperation. Princeton University Press, New Jersey
3. An Y, Borgida A, Mylopulos J (2008) Discovery and maintaining semantic mappings between XML schemas and ontologies. J Comput Sci Eng 2(1):44–73
4. Bianchini D, De Antonellis V, Melchiori M, Salvi D, Bianchini D (2006) Peer-to-peer semantic-based web service discovery: state of the art. Technical report, Dipartimento di Elettronica per l'Automazione Universit di
5. Bonifacio M, Bouquet P et al (2004) Peer-mediated distributed knowledge management. In: International symposium agent-mediated knowledge management, AMKM 2003, LNCS 2926: 31–47
6. Bouquet P, Giunchiglia F et al (2003) C-OWL: contextualizing ontologies. In: 2nd intl. semantic web conf. Springer, New York, pp 164–179
7. Castano S, Ferrara A, Montanelli S (2003) H-Match: an algorithm for dynamically matching ontologies in peer-based systems. In: The 1st VLDB int. workshop on semantic web and databases (SWDB), Berlin, Germany, pp 231–250
8. Castano S, Montanelli S (2006) Enforcing a semantic routing mechanism based on peer context matching,. In: Proc. of the 2nd int. ECAI workshop on contexts and ontologies: theory, practice and applications
9. Choi N, Song I, Han H (2006) A survey on ontology mapping. SIGMOD Rec 35(3):34–41

10. Colazzo D, Sartiani C (2005) Mapping maintenance in XML P2P databases. In: Bierman G, Koch C (eds) DBPL 2005, LNCS 3774, pp 74–89
11. Fergus P, Mingkhwan A, Merabti M, Hanneghan M (2003) Distributed emergent semantics in P2P networks. In: Proc. of the second IASTED international conference on information and knowledge sharing, Virgin Islands, November 2002, pp 75–82
12. Franconi E, Kuper G et al (2004) Queries and updates in the coDB peer to peer database system. In: Proceedings of VLDB04 (30th international conference on very large databases
13. Ghidini C, Giunchiglia F (2001) Local models semantics, or contextual reasoning = locality + compatibility. Artificial Intelligence Archive 127(2):221–259
14. Gomez-Perez A, Fernandez-Lopez M, Corcho O (2004) Ontological engineering. Springer, New York
15. Guarino N (1998) Formal ontology and information systems. In: Proceedings of formal ontology in information systems, Trento, Italy, 6–8 June 1998, pp 3–15
16. Gruber TR (1991) The role of common ontology in achieving sharable, reusable knowledge bases. In: Proceedings of the 2nd international conference on principles of knowledge representation and reasoning. Morgan Kaufmann Pub, San Francisco, pp 601–602
17. Hai Z, Jie L et al (2005) Query routing in a peer-to-peer semantic link network. Comput Intell 21(2):197–216
18. Halevy A, Ives Z, Mork P, Tatarinov I (2003) Piazza: mediation and integration infrastructure for semantic web data. In: Proceedings of the international world-wide web conference
19. Haase P, Broekstra J et al (2004) Bibster—a semantics-based bibliographic peer-to-peer system. In: Third intl. semantic web conf. (ISWC), Sardinia, Italy, 10–12 June 2002, pp 122–136
20. Haase P, Siebes R, van Harmelen F (2004) Peer selection in peer-to-peer networks with semantic topologies. In: Proc. of the semantics of a networked world. Semantics for grid databases. First intl. IFIP conf., ICSNW, Paris, France, 17–19 June 2004, pp 108–125
21. https://jxta.dev.java.net/
22. http://www.edutella.org/edutella.shtml
23. http://www.w3.org/RDF/
24. Joseph S (2002) Neurogrid: semantically routing queries in peer-to-peer networks. In: Proc. intl. workshop on P2P computing
25. Kementsietsidis A, Arenas M et al (2003) Managing data mappings in the hyperion project. In: The 19th intl. conf. on data engineering (ICDE), Bangalore, India, 5–8 March 2003, pp 732–734
26. Klein M, Kiryakov A et al (2002) Finding and characterizing changes in ontologies. In: 21st intl. conf. on conceptual modeling, Tampere, Finland, 7–11 October 2002, pp 79–89
27. Löser A, Staab S et al (2007) Semantic social overlay networks. IEEE J Sel Areas Commun 25(1):5–14
28. Liu L, Xu J et al (2008) Self-organization of autonomous peers with human strategies. In: ICIW 2008, Athens, Greece, 8–13 June 2008, pp 348–357
29. McCann R et al (2005) Mapping maintenance for data integration systems. In: Proceedings of the 31st international conference on VLDB, Trondheim, Norway, 30 August–2 September 2005, pp 1018–1029
30. Mawlood-Yunis A, Weiss M, Santoro N (2009) Reference model for semantic peer-to-peer networks. In: Proc. of 4th international MCETECH conference on e-technologie (Springer LNBIP), Ottawa, Canada, 4–6 May 2009, pp 319–334
31. Mawlood-Yunis A-R (2008) Reliable peer-to-peer semantic knowledge sharing system. In: Proceedings 3rd international workshop on reliability in decentralized distributed systems (RDDS), LNCS 5333, Monterrey, Mexico, 9–14 Nov 2008, p 894-0-903
32. A-Mawlood-Yunis R, Weiss M, Santoro N (2007) Fault classification in P2P semantic mapping. In: Workshop on semantic web for collaborative knowledge acquisition (SWeCKa) at intl. conf. on artificial intelligence (IJCAI)
33. Mena E, Illarramendi A et al (2000) OBSERVER: an approach for query processing in global information systems based on interpretation across pre-existing ontologies. Distributed and Parallel Databases 8(2):223–71
34. Mena E, Kashyap V et al (2000) Imprecise answers in distributed environments: estimation of information loss for multi-ontology based query processing. Int J Cooper Inform Syst 9(4):403–25
35. Nejdl W, Wolf B, Staab S et al (2002) EDUTELLA: searching and annotating resources within an RDF-based P2P network In: Proc. of semantic web workshop
36. Ng WS, Ooi BC et al (2003) PeerDB: a P2P-based system for distributed data sharing. In: Proceedings of 19th international conference on data engineering, 5–8 March 2003, Bangalore, India, pp 633–644
37. Nilsson M (2002) The Edutella P2P network—supporting democratic e-learning and communities of practice. In: McGreal R (ed) Accessible education using learning objects
38. Rousset MC (2004) Small can be beautiful in the semantic web. In: ISWC third international semantic web conference, Hiroshima, Japan, 7–11 November 2004, pp 6–16
39. Rousset M, Chatalic P et al(2006) Somewhere in the semantic web. In: Intl. workshop on principles and practice of semantic web reasoning, Budva, Montenegro, 10–11 June 2006, pp 84–99
40. Staab S, Stuckenschmidt S (2006) Semantic web and peer-to-peer. Springer Publishing, New York
41. http://wordnet.princeton.edu/
42. Zaihrayeu I (2006) Towards peer-to-peer information management systems. PhD Dissertation, International Doctorate School in Information and Communication Technologies, DIT — University of Trento

Abdul-Rahman Mawlood-Yunis PhD candidate, School of Computer Science at Carleton University. Masters in Science and Information System Science, School of Computer Science at Carleton University (2003), and Bachelor in Chemical Engineering, Technology University Baghdad (1991). His research interest include: Distributed Computing, P2P Networking, Information Systems, E-Commerce, Fault-tolerance, Mobile Agent, Software Agent and Software Engineering.

Michael Weiss an Associate Professor in the Department of Systems and Computer Engineering at Carleton University in Ottawa and a member of the Technology Innovation Management (TIM) program. His research interests include open source ecosystems, service-oriented architectures, mashups/Web 2.0, business process modeling, product architecture and design, and pattern languages. Between 2000 and 2007, he was a professor of Computer Science at Carleton University. From 1994 to 1999, he was a member of the Strategic Technology group at Mitel. Michael obtained his PhD (Dr.rer.nat.) from the University of Mannheim in 1993. He is author of over 70 peer-reviewed publications in leading journals and conferences.

Nicola Santoro PhD, is Professor of Computer Science at Carleton University. Dr. Santoro has been involved in distributed computing from the beginning of the field. He has contributed extensively on the algorithmic aspects, authoring many seminal papers. He is a founder of the main theoretical conferences in the field (PODC, DISC, SIROCCO). His current research is on distributed algorithms for mobile agents, autonomous mobile robots, and mobile sensor networks.