Reference Model and Simulation Framework for Semantic P2P Networks and its Application to Fault-tolerant Semantic P2P Networks

By

Abdul-Rahman Mawlood-Yunis

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

Carleton University
Ottawa, Ontario

Your file Votre référence
ISBN: 978-0-494-70532-2
Our file Notre référence
ISBN: 978-0-494-70532-2

To my loving wife Arkhawan, my lovely daughter Sara, my precious son Sipan, and my newly born angel Zylan; this dissertation is a gift with love for you all.
Abdul-Rahman

Acknowledgments

During years of PhD study, I experienced a lot of up and down periods; however, I never felt that I was alone. I always believed that my supervisors would be there when I needed them the most. For that, I would like to thank both my co-supervisors, and especially Nicola Santoro for his continued personal support.

Special thanks to my co-supervisor Micheal Weiss for his thorough review of my dissertation and publications. His generous comments, suggestions and availability for vastly improved my research work is deeply appreciated.

Many thanks to my proposal committee members, professor Amiya Nayak, professor Doron Nussbaum, professor Dorina C. Petriu and professor Babak Esfandiari for their valuable suggestions and consistent help. Many thanks and appreciation to my external examiner professor Qusay H. Mahmoud from University of Guelph for accepting to review and examine my thesis.

I would like also to thank my friend Dr. Abdulghany Mohamed from Sprott School of Business, Carleton University, for all his valuable time, suggestions, discussion, editing, etc. My thanks to Dr. Peter Taillon from School of Computer Science, Carleton University for his editing support on many occasions. Many thank to numerous people from the School of Computer Science for their incredible services. These include administration staff, Linda Pfeiffer, Sharmila Namasivayampillai, Claire Ryan, and Edina Sandler and technical staff, Gerardo Reynaga and Andrew Miles.

Finally, my apology to my wife and my kids for being preoccupied with my research work and not spending enough time with them. I hope that I would be a better Dad and compensate for what we missed because of my work.

Abstract

Current research directions in semantic peer-to-peer (SP2P) networks are evolving to combine two complementary technologies: peer-to-peer (P2P) networks and formally-structured information (ontology). SP2P systems represent the next step in the evolution of P2P networks because SP2P systems incorporate several additional features not present in P2P networks. Ontologies are advantageous in P2P networks because they provide the possibility for improving search and content retrieval. The current SP2P research efforts have generated many and diverse realizations and architectures. This diversity in implementation and architecture in turn has led to an ambiguity and incompatibility in defining domain abstracts and concepts. Progress in this area is hampered by a lack of commonality between these approaches, which makes their comparison and translation into practical implementations difficult. In this study, we describe a reference model for SP2P systems in an effort to model the emerging decentralized computing paradigm in a generic and high level abstraction. An SP2P simulation framework based on the reference model has been built. The framework is generic which enables instantiating different existing SP2P systems. A particular system can be considered an instance of the framework. This facilitates examining the effect of architecture changes, i.e., introducing different component implementations, on the behavior of the existing system. Further, the framework enables testing newly developed solutions to existing problems in SP2P systems. Simulation models are derived from the framework for studying the lack of fault-tolerance in the current SP2P systems. Two novel solutions are developed for reliability problem in the SP2P systems, and the effect of the developed solution on existing SP2P system behaviors are studied.

ad	Answer determination	Cp	Concept property
ae	Answer evaluation	Cr	Concept relationship
aj	Autonomous joining	GQA	Generous Query Answer
ap	Answer precision		Algorithm
ar	Answer recall	Int	Intermittent fault
as	Answer selection	$IRSP2P$	Irreducible SP2P system
av	Answer arrival	$JXTA$	P2P networking protocol
ch	Cycle handling	K	Coverage
cq	Compose queries	MC	Message complexity
d	Network degree	$MVQA$	Majority Voting Query
dm	Data model		Answer Algorithm
f	Peer profile	Nu	Neutral
fs	Query forwarding strategy	N	Network size
g	Number of groups	$O(L)$	Order of L,
i	Data instance		L is number of network links
id	Peer identification	OWL	Ontology Web Language
l	Query language	$P2P$	Peer-to-Peer network
lm	link (connection) management	Pn	Punish
lp	Length of mapping path	$\hat{P}$	Less sever punish than Pn
m	Mapping	Pr	Permanent fault
mc	Mapping correctness	Re	Reward
md	Meta-data	$\hat{Re}$	Partially reward
me	Mapping expressiveness	RDF	Resource Description
mi	Mapping implementation		Framework
mm	Mapping maintenance	$RSP2P$	Reducible SP2P system
mw	Mapping owner	S	Set
n	Peer's neighbors	$SP2P$	Semantic P2P systems
o_i	Partial ontology	$SP2P : sim$	Semantic peer-to-peer
p	Peer		simulation framework
pai	Partial answer integration	$SPRQL$	Query language for RDF
pq	Place queries	SQL	Structured Query Language
pv	Peer Discovery	Tr	Transient fault
q	Query	T_s	Entire system operation time
qa	Query Answerer	$TRQA$	Time Redundancy
qd	Query determination		Query Answerer Algorithm
qf	Query Formulator	TTL	Time-To-Live
r	Resource	$WordNet$	Open source online thesaurus
rt	Router	XML	Extensible Markup Language
sc	Select concepts	$Xquery$	XML Query language
sim	Similarity function	ϑ	Ontology
sn	Semantic Neighborhood	$\sqsupset$	Hyponymyous
t	Time	$\sqsubset$	Hypernym
tp	Routing termination policy	$\equiv$	Synonymous
w	Peer weight (Level of trust)	$\perp$	Not related
z_i	Index i	$*$	Related

Table 1: List of symbols and abbreviations

A peer	Represents an active object or an entity in the network
Autonomous joining	A process in which a peer autonomously select which other peers it is going to connect with
Mapping	The semantic relationship between concepts from independent information sources (ontologies)
Network degree	The constrain on the number of relations a peer could make
Peer discovery	Process or protocol in which peers discover their acquaintance(peers with similar profile)
Peer neighbors	The reference peer possesses to other peers in the network
Peer profile	The description of peer's domain knowledge, the information content offered by a peer, description of peer's schema, or expertise and services a peer provide comprise peer's profile
Semantic neighborhood	Connected peers with compatible information resources comprise semantic neighborhood
Similarity function	A function to measure the strength of the relation, the semantic affinity, between any two profiles
Synonymous relation ($\equiv$)	The relation c1 $\equiv$ c2 means that the two concepts are synonyms. In other words, c1 and c2 are different concepts with similar or identical meanings and are interchangeable.
Hyponymyous relation ($\sqsupset$)	The relation c1 $\sqsupset$ c2, means that the c1 have a hyponymyous relationship with c2, i.e. c2 is more generic or broad than c1
Hypernym relation ($\sqsubset$)	The relation c1$\sqsubset$ c2 means c1 is hypernym of c2. That is, c1 is more generic or broad than c2
Related relation ($*$)	Any other relations between concepts other than those described above can be captured by the $*$ relation.
Not related relation ($\perp$)	The relation $\perp$ means that two concepts have no semantic relation with each other.
Δ_1	Waiting time before detecting transient fault
Δ_k	Waiting time after detecting transient fault, k $\geq$2
u	% of network peers updating their ontology

Table 2: Definitions of some of the SP2P model concepts and the Simulation parameters

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem definition

Peer-to-Peer (P2P) is a scalable and effective paradigm for building distributed systems. P2P systems are inherently scalable since bottlenecks caused by central servers are substantially avoided. P2P systems also enable effective sharing of resources through direct exchanges between participants in the system. As a result, P2P systems become very popular and efficient way for sharing content or resources among distributed peers. The decentralization of control, the autonomy and the dynamicity of peers, and the effective sharing of resources are among the features which makes P2P networking attractive for large-scale distributed systems and applications. However, data and resource descriptions held by peers in a P2P networks lack explicit semantic.

Current attempts to solve problems pertaining to the lack of data semantic have focused on explicating the meaning of the information content, i.e. semantics augmentation. The backbone for semantics augmentation is ontology, which is concerned with defining a common conceptualization of the domain of interest plus a commitment of the involved parties to the conceptualization [43]. 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 [44].

In order to harness the power of P2P networks, research directions in P2P computing are evolving to combine two complementary technologies: P2P networks and ontologies. From this combining emerges Semantic Peer-to-Peer systems (SP2P). SP2P systems represent the next step in the evolution of P2P networks as they incorporate several additional features that are not present in P2P networks. These include formally-structured information (ontology), local mapping, and semantic based routing.

In SP2P systems resources are stored in numerous peers to be queried for [83]. Query execution process in SP2P networks is comprised of several steps [3, 48, 49]. Peers join a network after finding the first peer with compatible knowledge representation. That is, peers establish mappings to semantically related peers. Subsequently, peers submit queries to their neighboring peers using concepts in their own personalized local ontologies. Upon receiving a query, each peer starts processing the query locally. If the concepts used to formulate the query are compatible with concepts in its local ontology, it sends back query results to the querying peer (query initiator). Otherwise, it routes the query to other peers for which they have a direct mapping, after invoking the mapping component. Query forwarding will continue, until either: i. the query reaches the query initiator[1], ii. the query exceeds a specified number of query forwards ("hops") or iii. the time to live for the query message expires. The querying peer collects all answers returned and evaluates them for the correctness. The query initiator updates its confidence level in the directly connected neighbours in the capabilities to provide correct query answers. The query initiator can also inform the neighbouring peers about the query result. Thus, the entire query forwarding paths will be informed of the result of a un/successful query through propagating query result along these paths.

Ontologies are advantageous in SP2P networks because they provide the possibility for improving search and content retrieval. While most successful P2P networks are used for exchanging music and streaming files, e.g., BitTorrent[2], eMule[3], and KaZaA[4], SP2P networks will open up new possibilities beyond file-sharing and streaming. SP2P networks can enable richer and more useful descriptions of peers, services,

[1]The query must stop here, otherwise an infinite forwarding loop would be possible.
[2]http://www.bittorrent.com/
[3]http://www.emule-project.net/
[4]http://www.kazaa.com/

and shared artifacts. This will facilitate new ways of sharing knowledge, data management, and collaborative working within academic communities, research labs, universities, various emergency service departments, hospitals, and pharmacies [3, 57].

Current research efforts on SP2P systems have generated many diverse realizations and architectures. This diversity has, in turn, led to ambiguity and incompatibility in defining domain abstracts and concepts and, as such, has hampered progress in this area. For instance, system comparisons as well as their translation into practical implementation have been hindered. This diversity of SP2P implementations results from the variety of backgrounds (e.g., knowledge and database management, information retrieval, and P2P) of the different researchers and the still nascent state of the field. It is in such a context that this study endeavours to build a reference model for SP2P systems in an effort to model the emerging decentralized computing paradigm in a generic and high level abstraction.

Various definitions for the reference model and its specification are available in the literature, see [35, 36] for some of these definitions and requirements. The SP2P reference model developed in this study meets the essential requirements of the reference modeling, i.e., the model represents SP2P's class domain - a conceptual framework for understanding significant constructs of the SP2P systems and the relationship between them. It captures the characteristics common of many SP2P systems and provides normalized description of key concepts of these systems.

In order to obtain key constructs of the reference model, the primary focus was the identification of prominent or distinctive *features* of existing SP2P systems. The features are user-visible aspects or characteristics of prominent SP2P systems and other related work. They define both common aspects of the SP2P systems as well as the differences between them. The applied method is known as a Feature-Oriented Domain Analysis(FODA)[56].

The model leads to an establishment of common terminologies for the domain. Which, enables a better understanding and communication among members of the community – both important for the advancement of the current SP2P systems. The model also provide guidelines for comparison among individual systems. Individual systems could be compared with each other in terms of their compliance with the generic model, and their implementation of the generic features.

An SP2P simulation framework (SP2P:sim) will be derived from the reference model. The SP2P:sim framework captures the essential characteristics of existing SP2P systems such that a particular system (e.g. Chatty Web [3], Piazza [49], P2PSLN [48], KEx [16]) can be considered an instance of the reference model.

SP2P simulation models based on the SP2P:sim framework will be built. The models allow for the simulation of key parameters of different types of existing architectures. Hence, they can be used for studying various aspects of the current SP2P systems. For example, examining different adaptive query routing approaches, mapping methods, answer evaluation strategies, semantic neighbourhood building. In this study, the simulation will be used to study the lack of fault-tolerance capability in SP2P systems and its implications on the network connectivity of these systems. The lack of fault-tolerance capability of the SP2P systems is selected for detailed study because research has underlined the significance of fault occurrence, specifically semantic mapping faults, and the need to address them [24, 29, 33, 66, 73, 117, 119].

In the context of this study a *fault* is an incorrect semantic mapping, or the failure to map between concepts from different ontologies. A fault occurs when i. a concept in one ontology is mapped onto a semantically unrelated concept in a different ontology, or ii. a concept in one ontology cannot be mapped onto an existing semantically related concept in another ontology. The formal definition of fault is provided in Chapter 5.

The effect of semantic mapping faults could be minimized by distinguishing permanent from non-permanent semantic mapping faults, i.e. one need not treat transient faults as permanent ones. Distinguishing between different types of fault is important because non-permanent faults – both transient and intermittent(see Chapter 5 for definition of these terms) – do occur in open environments such as P2P networks. For example, ontology changes or evolution, static mapping, unavailability of data sources, and peer misbehavior are among the sources that (individually or jointly) generate non-permanent faults. The prevalence of such sources leads one to believe that transient fault occurrences are common. Moreover, it is important to emphasize that the occurrence and/or frequency of occurrence of such faults are not the only concern: one needs to also consider the impact of such faults as a single failure could be catastrophic.

Two novel solutions, Generous Query answerer Algorithm (GQA) and Time Redundancy Query answer Algorithm (TRQA), are developed for the lack of fault-tolerant problem in the SP2P systems. The effect of the solution on three SP2P systems: Chatty Web, Piazza and P2PSLN systems are simulated along with studying the effect of the solutions on the systems that are only possible to be built using our simulation framework.

In summary, the diversity of existing SP2P systems and their lack of compatibility are hindering progress in this area especially because it makes system comparison and translation into practical implementations difficult. A reference model and a reference model based simulation framework allow for the derivation of a variety of SP2P system. This enables a comparative study of a number of aspects of these systems along with studying the effect of introducing architecture changes on the existing SP2P systems. Hence, the significance and the need for an SP2P reference model and an SP2P simulation framework. The approach is validated by deriving simulation models from the simulation framework and using the derived models for exploring a specific problem, namely lack of fault-tolerance in SP2P systems. Two novel solutions are developed for reliability problem in the SP2P systems, and the effect of the developed solution on existing SP2P system behaviours are studied.

1.2 Objectives of the research

This research aims to build a reference model for SP2P systems. An SP2P simulation framework (SP2P:sim) derived from the reference model captures the essential characteristics of large numbers of existing SP2P systems. Simulation models developed based on the SP2P:sim framework allow for the simulation of key parameters of different types of existing systems. The simulation models will be used for studying the impact of our developed fault-tolerant algorithms on the existing SP2P systems.

More precise description of the objectives of this study is as follow:

Objective 1 *Build a generic SP2P reference model and a simulation framework based on this model to reproduce features of existing SP2P systems*

Objective 2 *Use generic SP2P reference model and simulation framework to examine the impact of architectural changes to existing SP2P systems*

Objective 3 *Derive simulation models from the SP2P:sim framework to test developed solutions for the lack of fault-tolerance capability on the reliability improvement of different existing SP2P systems.*

1.3 Research contributions

The contributions of this study are the following:

(i) Developing a reference model for the SP2P systems. The model leads to the establishment of common terminologies for the domain. In doing so, this leads to a better understanding and communication among members of the research community which are important for the advancement of the current SP2P systems. The model also provide guidelines for comparison among individual systems. Individual systems can be compared with each other in terms of their compliance with the generic model and their implementation of the generic features.

(ii) Building an SP2P simulation framework (SP2P:sim) based on the reference model. The SP2P:sim captures the essential characteristics of existing SP2P systems. It can be used for studying various aspects of the SP2P systems, for example, examining different adaptive query routing approaches, mapping methods, answer evaluation strategies, and semantic neighbourhood building.

(iii) Developing a preliminary solution for the lack of fault-tolerance problem in the SP2P systems. The solutions includes: 1) an analysis of the casual relationship between fault-types and fault-causes, and the description of the situation and circumstances under which different types of faults, transient, intermittent, and permanent, could arise; 2) developing two fault-tolerant query answer algorithms, namely Generous Query Answer algorithm (GQA) and Time Redundancy Query Answer (TRQA) algorithm. The algorithms detect permanent mapping faults and tolerate the transient ones.

(iv) Deriving simulation models from the SP2P:sim framework for studying different

aspects of existing SP2P systems. SP2P simulation results, obtained from running the simulation with developed algorithms, provide SP2P system designers with useful information about SP2P systems' behaviors, and the relationship between systems' behaviors and the underlying system parameters.

(v) Opening up new avenues of research on permanent and non-permanent semantic mapping faults in SP2P systems and in other related research areas including, emergent semantics [3, 2, 60, 102], semantic negotiation [39], and semantic web services [108].

This study also resulted in a number of publications, and they are:

Journal papers

Parts of chapter 3 will appear in Springer Journal on Data Semantics:

[1] A.-R. Mawlood-Yunis, M. Weiss, and N. Santoro. Reference Model for Semantic Peer-to-Peer Networks. In *Springer Journal on Data Semantics, JODS XV* (Accepted in April, 2010).

Parts of section 5.5.2 are published in:

[2] A.-R. Mawlood-Yunis, M. Weiss, and N. Santoro. From P2P to reliable semantic P2P systems. In *Peer-to-Peer Networking and Applications Journal*, Springer, 2010 (published online, January 2010).

Book chapter

Parts of chapter 2, and sections 5.3, 5.4, and 5.5.3 are published in:

[3] A.-R. Mawlood-Yunis, M. Weiss, and N. Santoro. Fault-Tolerant Emergent Semantics in P2P Networks. In *Cardoso, J., and Lytras, M. (eds.), Semantic Web Engineering in the Knowledge Society*, pages 161-187, IGI Global, 2008.

Conference and workshop publications

Parts of chapter 3 is published in:

[4] A.-R. Mawlood-Yunis, M. Weiss, and N. Santoro. A Reference Model for Semantic Peer-to-Peer Networks. In *Proceedings of 4th International MCETECH Conference on e-Technologies, LNBIP*, pages 313-394, 2009.

Parts of chapter 4 and section 5.5.2 are published in:

[5] A.-R. Mawlood-Yunis. Reliable Peer-to-Peer Semantic Knowledge Sharing System. In *proceedings of 3rd International Workshop on Reliability in Decentralized Distributed Systems (RDDS)*, pages 894-903, 2008.

Parts of sections 5.3.2 and 5.5.3 are published in:

[6] A.-R. Mawlood-Yunis. Fault-tolerant Semantic Mappings Among Heterogeneous and Distributed Local Ontologies. In *Proceedings of 2nd International workshop on Ontologies and Information Systems for the Semantic Web (ONISW)*, pages 31-38, 2008.

Parts of chapter 2 and sections 5.3 and 5.3.2 are published in:

[7] A.-R. Mawlood-Yunis, M. Weiss and N. Santoro. Fault Classification in P2P Semantic Mapping. In *Proceedings of Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa)* at *Intl. Conf. on Artificial Intelligence (IJCAI)*, 2007.

Parts of chapter 1 are published in:

[8] A.-R. Mawlood-Yunis, M. Weiss and N. Santoro. Issues for Robust Consensus Building in P2P Networks. In *Proceedings of International Workshop on Ontology Content and Evaluation in Enterprise (OnToContent)*, pages 1020-1028, 2006.

1.4 Research methodology

In this section, we describe the research methodology followed in this study. The study involved three key tasks: 1. reference model development, 2. simulation framework building, and 3. simulation model creation for validating the applied approach. The methodology steps are depicted in Figure 1.1, each box represents an activity in the methodology and the arrows show the sequence of the activities.

Reference model development

The study started with a thorough review of relevant works on SP2P networking, semantic mapping, semantic mapping faults, and SP2P simulations and modeling. The literature review revealed that there are different SP2P systems and types (e.g.

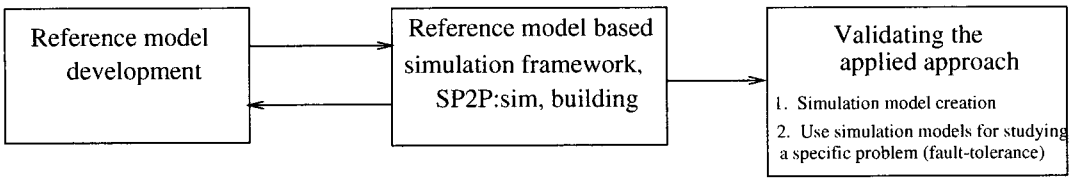

Figure 1.1: Research methodology activities

Piazza, Chatty Web, KEx, Somewhere, hyperion, peerDB, coDb, Esteem). These systems are incompatible with each other, employ different architectures, and were developed by professionals from different backgrounds (for details see Chapter 2). The review provided us with necessary information to start developing a reference model for SP2P systems. We obtained key constructs of the reference model from comparative analysis of existing SP2P systems. We studied the features of existing SP2P systems and identified the commonalities among them to create a construct for the reference model. Similar approach is applied in [61] for building a reference model for service oriented architecture and discussed in [56, 111] for building reference model for product line software and Mashup respectively.

Simulation framework building

An SP2P simulation framework SP2P:sim based on the reference model was built; for each individual construct from the reference model, a class or multiple classes were created, and related classes were grouped together to form packages. The simulation led to the revision of the reference model to include new design aspects discovered during reference model implementation. The creation of the reference model and the simulation framework were carried out in several iterations. In each iteration, new features were added to both the reference model and the simulation framework. The directed arrows in Figure 1.1 represent the model and framework development iterations.

The simulation framework allows for the simulation of constituent key parameters(e.g. query routing approaches, mapping methods, query answerer strategies) of SP2P system. As such, it allows for the configuration of a variety of SP2P architectures which could potentially be used for a comparative study of a number of aspects of these architectures including fault-tolerance, data modeling, and query routing. In

this study we focused on the issue of fault-tolerance.

Validating the applied approach

The reference model and the SP2P:sim framework were validated by the use of simulation models that were created for specific problems, namely fault-tolerances. Based on the thorough review of works on fault-tolerance, semantic mapping, and mapping fault analysis of the relationship between fault causes and fault types a fault classification taxonomy was developed. Proper fault-tolerance methods were identified, and used to develop a set of three fault-tolerance algorithms for SP2P systems, viz.: Generous Query Answer (GQA) algorithm, Time Redundancy Query Answers (TRQA) algorithm, and Majority Voting Query Answer (MVQA) algorithm[5]. These algorithms were used in the simulation model to test how they impacted specific SP2P systems. Aspects related to the simulation running and how a particular SP2P system can be configured using the simulation components, and issues relevant to network creation and resource distribution are described in chapter 4.

1.5 Organization of the dissertation

This dissertation is organized as follows: Chapter 2 surveys the work most relevant to this study. Chapter 3 describes a reference model for SP2P systems, Chapter 4 describes an SP2P simulation framework, SP2P:sim, In Chapter 5 the lack of fault-tolerance capability of SP2P systems is selected for detailed study. The chapter is divided into four sections: Section 5.2 describes building SP2P simulations using SP2P:sim framework, and Section 5.3 defines fault and fault types, and lists fault causes as well as classifies them along the temporal dimension. Section 5.4 demonstrates SP2P's execution process and the negative impacts of a lack of fault-tolerance in SP2P systems. Section 5.5 describes two solutions, Generous Query Answerer Algorithm, subsection 5.5.2 and Time Redundant Query Answerer Algorithm, subsection 5.5.3, for the lack of fault-tolerant problem in the SP2P systems. Finally, in Chapter 6 the dissertation is concluded and future research work is identified.

[5]while we have implemented and tested MVQA algorithm, the experimental results are not included in this study for the lack of time. Hence, no further discussion on MVQA is provided

Chapter 2

Related Work

2.1 Introduction

The incorporation of ontologies in P2P networks has been previously reported in the scientific literature in various research precedents such as: creation of a semantic overlay network, where semantic networks are created on existing P2P networks, semantic-based query routing, and adaptive query routing. SP2P systems which employ ontologies include several types: P2P Knowledge Management systems, P2P Databases, P2P Semantic Web, P2P Emergent Semantics, P2P Web Service, and P2P Information Systems. Table 2.1 lists these system types and some system instances.

In this chapter three relevant literature streams are reviewed: i. semantic reconciliation approaches, ii. reference models and iii. the maintenance of mappings in SP2P

SP2P Types	**System Instances**
P2P Knowledge management	KEx[16]
P2P Database	coDB [37], Piazza [49], PeerDB [82], Hyperion [57]
P2P Semantic Web	BiBSter [46], Somewhere [98], P2PSW[104]
P2P Emergent Semantics	Chatty Web [3], DisES [34]
P2P Information System	P2PSLN [48], Observer [76], P2PISM [117]
P2P Web Services	ESTEEM[13]

Table 2.1: SP2P system types and instances

systems. Furthermore, the differences between P2P and SP2P are highlighted.

2.2 Semantic reconciliation approaches

We observed from the literature review that approaches to the semantic reconciliation among distributed, autonomous and heterogeneous information sources are somewhat different from each other. The existing works could be roughly classified into four different inter-related classes:

- Local mapping and query translation,
- Collaboratively built ontologies and consensus reaching,
- Pattern extraction
- Tagging and social networks.

The names are related to the way each approach tries to reconcile the semantic differences among different information source representations. In the following subsections, a description for each approach along with the review of some prominent contributions in each class is provided.

2.2.1 Local mapping and query translation

In this part we provide a short description of several different systems. These systems are: Chatty Web [3], KEx [17], H-Match [20], Bibster [46], Piazza [49], OBSERVER [76], Edutella [83], and SomeWhere [98]. The underlying networks for these approaches are mostly P2P networks and a common theme among selected approaches is the use of local mappings and query forwarding to achieve some from of knowledge sharing and cooperation. In other words, peers have their own local data representations (ontologies) and local mappings between local information presentations are used for query re-writing and query answer formation. Bibster is the exception as it relies on existing global ontology.

Chatty Web [3] describes a method for building a *common ontology* or semantic global agreement from local interaction between peers. Each peer has its own ontology

which might be different from 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. During their lifetime, peers will be able to find other related peers. That is, through the normal operation of the system, query translation and forwarding, peers will be able to identify their semantically related peers. The lowest common knowledge among all the peers of the network constitutes a shared conceptualization of the domain under the discourse.

The Chatty Web account for the semantic mapping fault, uses a simple probability function with the assumption that faults are equally distributed and independent from each other. Despite the fact that the admission for the semantic mapping fault by Chatty Web authors is an important step, we believe that their assumptions are too lax and more importantly, not all faults are permanent.

OBSERVER [76] is an approach for information query processing by defining a semantic relationship among pre-existing ontologies. The approach tries to reduce the problem of knowing the semantics and the structure of all information systems around the globe to defining synonymous relationships between concepts used in various ontologies of the same or related domains. Thus, the problem of searching whole global information systems is reduced to searching multiple ontologies.

In OBSERVER, the synonymous relationship between ontology concepts from different ontologies is determined by human experts and saved on an inter-ontology relation server. The inter-ontology relation is used by a query processor component for locating semantically related information. Observer is an SP2P system, because it uses mapping and does not require common conceptualization of the domain, i.e., global ontology.

In addition to the maintenance and scalability problem associated with the centralized inter-ontology relation component of the system, the semantic relationship between concepts in OBSERVER is declared manually. This is a serious disadvantage in comparison to systems which determine on the fly the relationship between concepts from different ontologies.

OBSERVER is concerned with semantic losses during translation among ontologies; it accepts that semantic losses could result in information retrieval loss during query processing. OBSERVER is satisfied with calculating semantic losses, and does not

ratify the problem [77].

Piazza [49] developed an infrastructure and mapping language for semantic mapping and data management in P2P environments. The system takes into account both the domain and the document structure. In Piazza, queries posed on one peer can be reformulated and be posed on semantically related peers. The transitive closures of the translations among peers are used to answer queries.

Piazza's contribution to data management is an important development toward moving to P2P distributed semantic data management as opposed to the current practice of data integration and mediation based systems. However, peers in the Piazza system are static entities. That is, they are not expected to leave the network. Thus, Piazza lacks the ad hoc property of P2P open networks. It is also difficult for new peers to join the Piazza's P2P network because of its rigid structure.

Mappings among Piazza peers' are carefully designed and created manually. In the Piazza system 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 method.

H-MATCH [20] is a dynamic ontology matching algorithm. The algorithm is used in distributed open networks. Each node in the network provides partial ontology of the domain that has been built independently from others. The algorithm is used to enable knowledge sharing and ontology evolution.

To perform matching, H-MATCH uses the meaning of concept names and property names as well as concept context. The matching processes use the WordNet thesaurus to determine the relationship between concepts. H-MATCH provides three different levels of matching: shallow matching, intermediate matching and deep matching. The differences in these matching levels are found in the details they consider during mapping.

KEx [17, 16] is an architecture for semantic searches in a P2P network based on the principles that the heterogeneity of knowledge representation should not be seen as an obstacle to knowledge management; rather, it should be seen as an opportunity for promoting innovation.

KEx facilitates building a community of sharing among autonomous peers. The

knowledge within the community will be available for peers and the search for interesting information will be possible. Each peer could either request information or provide information to other peers or group of peers.

A document repository and a context repository are associated with each peer. A document repository is place where the structured document are saved and the context repository where the semantics of concepts will be clarified during query requests and responses. Different kinds of knowledge could be saved in document repositories including references to experts in some domain, links to other peers and to external resources.

KEx's approach to knowledge management is a pioneering attempt to replace centralized knowledge bases with distributed autonomous ontologies. We will consider configuring KEx using our simulation model and study the effect of fault-tolerance on its answer retrieval improvement.

Bibster [46] is a P2P system for sharing bibliography files among researchers. Peers in Bibster use common shared ontology to model local files, query content and peers' expertise, where peer expertise refer to peer knowledge.

ACM topic hierarchy and SWRC ontology are used as a common ontology. That occurs when a bibliographic entry is made available to the BibSiter system; it will be automatically aligned with the ACM and SWRC ontologies.

Peers advertise their expertise in the network and discover other peers with knowledge that may be relevant to answer user queries. Discovery is based on the semantic matching between semantic content of the query and the expertise model of the peer.

The knowledge of other peers' expertise forms the semantic network. In other words, matching between query content and the expertise model is used for ranking peers and that ranking forms the basis for intelligent query routing.

Using common ontology in P2P networks jeopardizes peers independency, the core attribute of P2P networks, by presenting their local data and semantics. We believe that an appropriate approach to semantic knowledge sharing should not violate peer independency in creating their own local semantics. We believe that local semantic mapping among peers with heterogeneous information representation is a more suitable approach for open information systems, such as those system based on P2P

networks.

SomeWhere [97, 98, 5] envisions the Semantic Web as a large P2P data management system. In such a system, peers will have small class taxonomy to annotate their personal documents with semantics and make use of local semantic mappings for the exchange of information with each other.

Creating a large Web of people (peers), (i.e., building a new Semantic Web system comparable in scope and scale to the current Web without centralized components) constitutes the main objective of the SomeWhere project.

Despite SomeWhere's engagement with local ontologies, semantic mappings and the P2P system architecture makes it attractive for research. SomeWhere's main preoccupation with creating a Semantic Web and with no attention to information exchange, makes it irrelevant to our research problem.

Edutella [81] enables sharing of learning resources among distributed and independent educational resource providers. Edutella uses the JXTA[1] protocol for its P2P network infrastructure, and RDF[2] 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. Furthermore, query routing in Edutella is not adaptive. Once peer groups have formed, they continue to be used for broadcasting queries in the network and control or ownership by peers over local resources has only been considered recently[3].

2.2.1.1 Some features of the SP2P systems

A number of aspects from the SP2P systems are summarized in Table 2.2. The working environment for selected approaches are mainly P2P networks and the aspects are *purpose, mapping approach, future interaction and fault-tolerance capability.* The meaning of these aspects are as follows:

(a) by **Purpose** we refer to the exact goal of the approach or the system.

(b) Mapping strategy refers to the system attitude to word mapping. For the

[1]https://jxta.dev.java.net/
[2]http://www.w3.org/TR/rdf-sparql-query/
[3]http://www.edutella.org/edutella.shtml

systems surveyed, the following mapping modes were identified: i. Re-use of existing mapping methods. Systems following this mapping strategy, consider mapping to be a separate activity and existing mapping methods could be plugged into their approach. ii. Development of scattered mappings. Systems following this strategy, developed their own semantic mapping method and used it in mapping between peers. iii. Development of the centralized mapping. In this approach, systems have a separate component where the relationship between concepts from various ontologies can be declared manually.

(c) Future interaction deals with how different approaches use the collected information from the interaction with other peers for future interactions. Three categories can be identified in this regard: i. Making use of the interaction history for *intelligent future query forwarding.* ii. Cache answers for better performance. iii. Naive semantic query forwarding where neither performance nor intelligent feature routing are considered.

(d) Fault-tolerance considers peers' reaction to the interactions that result in incorrect answers in various approaches.

2.2.2 Collaboratively building ontologies and consensus reaching

An ontology engineering methodology has been suggested in [109] to enable knowledge management in a distributed environment with heterogeneous information sources. The methodology embraces building ontologies collaboratively and reaching consensus on the semantics of concepts and domain conceptualization.

The procedure starts by building general core ontology; individual users then extend the core ontology and adapt it to their local context. Later, the core ontology users provide feedback on the core ontology quality, i.e., what should and should not be part of the core ontology, to a centralized authority. By having users express their observations and suggestions to a centralized authority for analysis, a shared common ontology will emerge from the feedback and the core ontology after several iterations.

To track argument exchanges by peers participating in the ontology building, and to help new users of the system to understand prior design decisions, authors of the

	Purpose	**Environment**	**Mapping**	**Future interaction**	**Fault-tolerance**
Chatty Web	Building global consensus	Orthogonal	local and preexist	Routing table built	Only Permanent Faults considered
OBSERVER	query processing in global information system	Network of related ontologies	Centralized terminological relationship	Not handled	Not applicable
Piazza	Query answering using mapping	Network of sites or peers (P2P)	local and preexist	Query answers cached	Not handled
H-MATCH	Knowledge sharing and evolution	P2P	Built-in	Routing table built	Not handled
KEx	Knowledge discovery and exchange	P2P	Built-in	Query answers cached & Routing table built	Not handled
SomeWhere	Semantic search	P2P	Built-in	Not considered	Not Considered
Bibster	File sharing	P2P	Not used	Peers ranked	Not handled

Table 2.2: Some aspects of the selected systems and methods

methodology complemented their prior work by designing an ontology for argument exchange [109].

Building ontologies using the above methodology stresses the centralized role of ontology in enabling interaction between distributed and heterogeneous information sources. That is, the focus of the methodology is on the shared property of the ontology.

Using a common ontology to overcome the semantic incompatibility problem among heterogeneous information sources is not realistic for an open and dynamic environment such as P2P systems. This is because P2P systems are comprised of a high number of peers, and it is difficult to force the peers to use a similar data representation. Other shortcomings of the methodology are: it requires continuous human involvement, and it ignores the fact that there are several partial ontologies for almost all domains which they have to reuse.

To enable scalability, the authors of [27, 28] suggest introducing meaning negotiation and continuous alignment to the described ontology engineering methodology. The success of this extended work has yet to be seen.

2.2.3 Pattern extraction or structure similarity

In this section we present two relevant and similar contributions. Namely, Distributed Emergence System (DistES) [34] and constructing consensus ontologies for the semantic web [105].

The DistES protocol is based on an evolutionary algorithm for discovering and merging knowledge in P2P environments. Each peer owns local data which is represented in a hierarchical structure. Peers extend their knowledge by querying other peers, selecting the best results among the query answers and merging the selected results with local data.

The process of selecting foreign concepts and forging concept relations for integration with local data is based on their occurrences extend in the query answers. Concepts and concept relations (See Chapter 5 for information on ontology concepts and concept relationships) with high occurrence, appearing in multiple query answers, will be selected for merging with local data, and those with fewer occurrences are ignored.

The emerging ontology manifests the general consensus among peers that participated in the interaction.

Similarly, [105] uses the occurrence rate of concepts and concept relations among multiple, small and related ontologies used for web annotation, to construct a merged ontology on the fly. The newly constructed ontology will be presented to the user for refining the search. The occurrence rate strategy, called the pattern reinforcement methodology by [105], is also used for removing inconsistency in the conceptualization (i.e., contradicting statements) that might occur during the creation process of the new merged ontology.

2.2.4 Tagging and social networks

The launch of the social book marking Web site "del.icio.us"[4], the photo sharing service "Flickr"[5] and others, opened-up a new way for categorizing Web information sources, i.e. building ontologies collaboratively by mass of Web users.

A network of English words comprised of *numerous tags* used by independent users for labeling *same online document* forms the basis for ontology creation using this strategy. Similarly, using the *same tag* by independent users to refer to *numerous different resources* is the basis for creating online communities around using common resources, i.e., share common interest [78].

Currently, a great deal of discussion and interest have been devoted to social networking and collaborative ontology building in academia. Several approaches following this strategy have been surveyed in [103].

The fact that Web users choose to label web documents, and that different user groups have contradictory interests, make the success of this approach unlikely. We believe that while it is possible for the social tagging approach to succeed in building ontologies for some general domain, taking total advantage of the values' ontologies add to the masses of existing information in various domains – explicating the meaning of the information, hence, enabling machine processing of information and improving information search and information integration – is a more involved procedure than allowing application user to label resources.

[4]http://del.icio.us/
[5]http://www.flickr.com

2.3 Reference models

To the best of our knowledge, there are only few works that directly address the problem of building reference models for P2P networks. Some of these works are described here. In [1], a reference model for unstructured P2P networks have been presented. In addition to identifying core components of P2P networks, [1] discusses the network's essential design decisions. It also provides a brief comparison of some relevant P2P networks. Similarly, a reference model for structured P2P networks has been provided in [26]. From a high-level abstraction view, we consider the reference model which will be described in this study to be an extension/adaptation of the mentioned P2P reference models to a new environment. In this new environment, semantic aspects play essential roles in modeling and building P2P network. That is, In addition to the components described in [1, 26], components such as semantic mapping, semantic neighborhood, query formulator, semantically enhanced resource description/representation are SP2P's specific model components. Other related works are [64, 101]. In [101] authors show only preliminary steps toward modeling semantic overlay networks. The efforts in [64], on the other hand, is spent more on discussing different query routing strategies rather than generic model. There are also some related works in a closely related domain, i.e. grid domain, for example [91]. These works were helpful for understanding system layers and describing components from high level perspective.

2.4 Semantic mapping maintenance

Peers in SP2P systems are independent or autonomous with regard with who they interact with and on how they respond to service requests. These peers are scattered around the world with their own information content. For example, information stored in repositories is located at different government departments, research labs, universities, interest groups, enterprises. Peers also have total control over their local information resources: They can change, update, remove or restrict access to their information.

Information or data held by peers in a SP2P system is represented heterogeneously

along different aspects. For example, data or information can be in XML[6] files, relational tables, text files, or RDF documents[7]. Even when the same type of representation format is used for storing information, information modeling, structure, and semantic concepts used in the modeling may vary among different peers. An example of semantic differences would be using different vocabularies to refer to the same physical or conceptual object by different information representations: one's "zip code" is somebody else's "area code". Another example is using the same vocabulary to refer to different conceptual or physical real life objects in different representations: a "terminal" for one person may refer to a computer monitor, but may signify a "station" for somebody else.

SP2P systems are prone to information incompatibility. This issue arises due to the fact that peers in a P2P environment are autonomous. Which, in turn, permits the existence of heterogeneous information resources or data. For example, concepts developed independently by different academic researchers, different research labs, various emergency service departments, hospitals and pharmacies, just to mention a few, are a sure source of knowledge incompatibility among these independent parties (peers). The current success of the P2P network and Semantic Web initiatives have indeed underscored the lack of semantic compatibility among heterogeneous information representations [3].

Heterogeneous distributed information systems need to overcome the semantic interoperability problem in order to communicate usefully. Indeed, solving this problem becomes imperative for the success of information search and retrieval applications as well as for the success of organizations that rely on them. Attempts to solve the problems pertaining to heterogeneity of information representation 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 on the part of the involved parties to this conceptualization [44, 43].

In using ontology for modeling information sources, concepts are defined in terms

[6]http://www.w3.org/XML/
[7]http://www.w3.org/RDF/

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 [75, 44].

Despite some usefulness and existence of a number of common ontologies [42], the prominent difficulties with this type of work include problems of: adaptation (common ontology undermines peers' autonomy), maintenance (ontology domain concepts change or evolve over time), and scalability and expressiveness (determining future growth of the ontology and deciding on the appropriate level of detail of the ontological description [97]). Furthermore, in a dynamic, open and distributed environment such as P2P networks, a common ontology solution is less feasible because requiring all peers commit to a common meaning is impracticable.

To overcome limitations associated with using common ontologies, Contextualization, or local ontologies, has been suggested [16, 19, 40] 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 provider's knowledge of the application domain, experience, or culture. This implies a shift from large and centralized ontologies to small and distributed ontologies.

Contextualization eliminates problems associated with the use of common ontologies, e.g., maintenance, scalability and adaptation problems, and the need for peers' commitment to common ontologies. It also helps the formulation of queries that can be understood by other peers. Shifting from large and centralized ontologies to smaller and probably simpler distributed local ontologies, contextualization opens the door for solving semantic incompatibility problem.

Correct semantic mapping is fundamental in information exchange among distributed and heterogeneous parties. Various research groups have considered the issue of incorrect semantic mapping, i.e. semantic mapping faults [3, 2, 24, 29, 33, 53, 77, 79, 94]. Earlier studies that were concerned with the effects of semantic mapping faults include: i. those that have looked into the existence and classification of different types of faults [53, 79, 94]; and ii. those which were concerned with distributed semantic knowledge exchange [3, 77]. In the latter group, correctness is scrutinized mainly through semantic mappings and query forwarding. While semantic mapping is concerned with the quality of the mapping, semantic query forwarding is concerned

with the estimation of information loss that occurs due to query concept dropping, the replacement of a query concept by a more general or a more specific concept in the query translation chain.

In pointing to future research direction, Ehrig, emphasizes that "*errors are an integral part of ontology alignment*", in the sense that "*despite biggest efforts in reducing errors in alignment of ontologies, mismatches always have to occur*" in both centralized and distributed ontologies [33]. In the same vein, Cudre-Mauroux, Aberer and Feher caution that one should not assume or take for granted that mappings created by autonomous parties in peer data management systems are always correct [24](see also [3]). Moreover, Doan and Halevy argue that "*In dynamic environments, sources often undergo changes in their schema and data. Hence, it is important to evolve the discovered semantic mappings. A related problem is to detect changes at autonomous data sources (e.g., those on the Internet), verify if the mappings are still correct, and repair them if necessary*". Indeed, Doan and Halevy point out that, "*Despite the importance of this problem it has received relatively little attention*" [29].

While these studies have recognized the importance of maintaining correct semantic mappings in the presence of data and schema changes, they do not distinguish between permanent and non-permanent mapping faults. Non-permanent fault(s) have significant implications for system robustness, efficiency and inclusiveness. A robust SP2P system requires a consideration of non-permanent faults because a lack of such capability could – in the event of transient or intermittent faults – potentially lead to a disruption in services and loss of resources that peers provide. Moreover, when SP2P systems use learning from past interactions to make future query routing decisions (as opposed to flooding techniques), the distinction between permanent and non-permanent mapping faults becomes a critical issue. Unless query routing adaptation is based on correct assumptions about permanent mapping of faults there is a potential for excluding semantically related peers from future routing decision(s). The potential for preclusion is particularly relevant for real time applications as well as mission-critical applications such as national security and business-to-business applications. For related examples, in a security situation discarding a viable source of information simply due to transient faults could negatively impact the level of accuracy of the collected information; or in business, preventing a valuable business partner from participating in business transactions because of transient faults could jeopardize potential financial gains in business-to-business applications.

The distinction between permanent and non-permanent semantic mapping faults is also important when building ontologies using the Emergent Semantics approach. Emergent behaviour is a well-known phenomenon in biology, physics and (distributed) computing. For example, several optimization and network routing techniques have been inspired by the way the behaviour of an ant colony, as a whole, emerges from local interactions between individual ants.[8] Similarly, local cooperation between robots in multi-robot systems for search and rescue operations has been modeled after the formation of flocks of birds [10].

Inspired by emergent behaviour, the approach of Emergent Semantics has been proposed as a solution to the semantic interoperability problem among autonomous, heterogeneous information sources with local ontologies. Emergent semantics refers to the bottom-up construction of interoperable systems, in which semantically related peers are discovered and linked together during the normal operation of the system, as part of regular search and query forwarding operations. In this approach, individual information source providers supply semantic mappings (so-called *semantic bridges*) between their own local and semantically-related foreign information sources [3, 2, 60, 102]. Emergent Semantics (consensus formation) in a P2P network is the lowest common knowledge among all peers' contextual ontologies in the network.

The distinction between permanent and non-permanent semantic mapping faults during emergent semantics process can result in the building of domain ontologies and the generation of emerging shared semantics that are more complete and agreeable than those that are built without the consideration of temporal semantic mapping faults.

More recently some researchers have started looking into the issues that one could characterize as pertaining to transient mapping faults. For instance, as in [119], peers that participate frequently in answering queries but their answers are not correct are double checked for schema changes. Thus, a recurrent incorrect query answer from semantically related peers is used as a detection mechanism for the need to check ontology mappings. Once schema changes have been detected, user intervention is requested to repair mappings in order for peers to continue collaborating. The described situation – detection and repair of mapping faults over a period of time – is a manifestation of what is called transient mapping fault occurrence. It is unfortunate

[8] http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html

that while the effect of such transient fault could be drastic on system functionality, the authors of [119] do not take any action other than highlighting the need for manual repair. It is not clear when (e.g. after how many faults) it will take place.

Zaihrayeu [117] acknowledges the fact that P2P networks could change during query propagation, for example a peer may become temporarily unavailable or a new peer with relevant information sources joins the network. Zaihrayeu highlights three different scenarios which have the potential for generating transient faults. The scenarios are made up of three parts: query submission, network change, and query arrival time. That is, whether a submitted query will arrive before/after a related peer has been dropped or before/after a new relevant peer has joined the network. To cope with the dynamic property of P2P networks, Zaihrayeu suggests implementing a query propagation algorithm with assumptions that allow treating a dynamic P2P network as a static one. The suggested assumptions are:

"*i. if a related acquaintance query (a query from peers with relevant profile) is added by some peer after the user query was submitted, but before the corresponding net query reached the peer, then the net query is processed as if the acquaintance query existed before the user query was submitted;*

ii. if a related acquaintance query is deleted by some peer after the user query was submitted, but before the corresponding net query reached the peer, then the net query is processed as if the acquaintance query did not exist before the user query was submitted; and

iii. if a related acquaintance query is added by some peer, and, by this time, the peer has already processed a net query, whose rewriting would include the newly added acquaintance query, then the acquaintance query is not considered for this net query. In other words, the effect is the same as if the acquaintance query were not added."

The above described situations suggests that posing the same query at different times one might get different answers; in other words, query answers are time-dependent. Hence, in cases where wrong query answers are obtained, one should not assume that the answers are going to be permanently faulty. Thus, there is a need for query propagation algorithms that are transient-fault tolerant.

In an effort to manage dynamic versioning and evolution of distributed ontologies, Ma et al. [66] urgently argue for need for an ontology evolution representation approach

with time constraints. A key issue in the approach is the prevention of inconsistency arising among distributed contextualized ontologies[9].

To prevent inconsistency from happening, due to a change or evolution in ontology, Ma et al. suggest that a change in one ontology should trigger or be followed by changes in other related ontologies. Furthermore, given a specific sequence of change operations performed on these distributed environments, Ma et al. transform the inconsistency prevention problem into the problem of finding [66]:

"*whether the sequence is timing consistent with respect to all the time constraints of the distributed environment and, the need to know whether there exists a sequence of ontology change operations that satisfies all the time constraints of the whole distributed environment*".

While we concur with Ma et al [66]. that inconsistency among connected ontologies due to evolution and versioning has to be prevented, we do not believe that requiring sources which are undergoing changes to inform their connected ontologies (or risk isolation) will help to solve the problem of transient mapping faults. This is because the target ontology will take time to respond to the update notification and as such during the intervening period (i.e., between notification and update) ontologies will remain inconsistent and so transient mapping faults will arise.

McCann et al. [73] argue that , "*In dynamic environments, such as the Web, sources frequently change their query interfaces, data formats, or presentation styles. Such changes often invalidate semantic mappings, causing system failure*". Thus, McCann et al. build a MAVERIC (Mapping Verification) system, which continuously monitors sources "*for detecting broken mappings*" automatically. In this system, information sources are probed periodically and query answers are compared to the prior known answers. Once newly retrieved query answers differ from the predicted/existing ones, an alert signal about a potential broken map is sent to the system administrator. This work is relevant, but significantly different, than the fault-tolerance algorithms which will be presented in section 5.5. First, while McCann et al. focus on faults due to information source changes, we consider a broader set of fault sources including not only changes in information sources but also on source unavailability, static mapping, etc (see section 5.3.3). Second, rather than monitoring information sources continuously, we suggest detecting changes only when we query information sources.

[9]also referred to by partial ontologies

That is, we are interested only in detecting changes that are current and relevant to our queries. Furthermore, McCann et al. have not fully considered the implications of frequent and numerous changes of information sources.

In summary, previous studies have underlined the significance of faults occurrence during semantic knowledge exchange among peers, and the need to address them. To effectively deal with semantic mapping faults, we argue that SP2P systems need to be semantic mapping fault-tolerant. The lack of fault-tolerance in these systems, when used for real time or critical applications, could potentially increase financial losses and security risks as well as fail to use available resources. To introduce fault-tolerance in SP2P systems it is crucial that faults be differentiated between permanent and non-permanent ones, especially given that for various reasons a certain percentage of faults are transient. If faults are not differentiated, non-permanent faults will be treated as permanent with serious consequences including lack of inclusiveness, efficiency and robustness. In this study the reference model and the SP2P:sim framework were validated by the use of a simulation models that were created for examining and resolving the lack of fault tolerance problem in SP2P systems.

2.5 Differences between P2P and SP2P systems

SP2P is the latest development in the P2P networking advancement. SP2P systems incorporate several additional characteristics not present in P2P networks. We reviewed existing SP2P systems [3, 16, 34, 46, 48, 57, 76, 104, 117] and other research on semantic P2P systems [21, 20, 47, 54, 62], and came to the conclusion that there are several characteristics that distinguish P2P systems from SP2P systems. These include: 1) formally-structured information, 2) local mapping, 3) autonomous peer resource management, and 4) semantic based routing.

Data or information managed by peers in SP2P systems is *structured* and formal (e.g., meta-data about learning objects in Edutella [81] and domain ontologies in observer [76]). The purpose of formally-structured data is to enrich data semantics and support inferences which in turn improve search performance and the quality of retrieved information.

Local mapping in SP2P systems is used as a translational capability to forward queries

between the peers under the conditions when the peers possess different data schema or data representations.

Autonomous peer resource management pertains to peers' control over own resources. That is, in contrast to conventional P2P networks, resources in SP2P are neither replicated nor assigned to other peers in the network in order to be used by network peers for processing queries. This is because the focus of SP2P systems are mainly applications where replication of resource is not permissible. Examples of such application include a collaboration between different health departments, research labs, and universities where replication of resource is not permissible [31, 49, 81, 57]. However, in semantically-enhanced P2P file sharing systems this feature can be relaxed.

Query routing in SP2P systems is different than non-semantic P2P systems. This is mainly due to the fact that SP2P systems are unstructured P2P networks. That is, SP2P systems are different than structured P2P networks such as Chord [25] or Pastry [99] and other distributed hash table based systems. In SP2P, semantic based peer selection relates peers with similar domain knowledge, and these relation links are used for query routing process. That is, in SP2P systems the less relevant peers are ignored during network building and queries are sent only to the most relevant peers. For more information about query routing in SP2P systems see sections 3.2.5 and 4.5.6.

We see the above described system aspects to be prominent characteristics that differentiate SP2P systems from the conventional P2P networks like BitTorrent[10], eMule[11], and KaZaA[12].

[10]http://www.bittorrent.com/
[11]http://www.emule-project.net/
[12]http://www.kazaa.com/

Chapter 3

Reference model for SP2P networks

The current SP2P research efforts have generated many and diverse realizations and architectures. This diversity in implementation and architecture has led to an ambiguity and incompatibility in defining domain abstracts and concepts. This problem is mainly due to the involvement of a variety of researchers from different backgrounds (e.g., knowledge management, databases, information retrieval, P2P) into a still recent and evolving area.

In this chapter, we describe a reference model for SP2P systems in an effort to model the emerging decentralized computing paradigm in a generic and high level abstraction. The model represents SP2P class domain - a conceptual framework for understanding significant constructs of the SP2P systems and the relationship between them. It captures the characteristics common of many SP2P systems and provides normalized description of key concepts of these systems.

The potential contribution of the reference model to the advancement of the current SP2P systems spans various areas. These include: i. an establishment of common terminologies for the domain, this leads to better understanding and communication among members of the community. ii. Providing guidelines for comparison among individual systems, individual systems could be compared with each other in terms of their compliance with the generic model and their implementation of the generic features.

3.1 Model development method

In this section, the model development method that was used for identifying the model components, features and properties is described. The study was started with a thorough review of relevant works on SP2P networking, semantic mapping, and SP2P simulations and modeling. The literature review revealed two important facts.

First, the review disclosed that there are different methods to build the reference model. For example, Volker and Katarina [111] explain that there are two main methods for building reference models: i. when a large number of systems for the domain under consideration are available, the reference model is developed by extracting common components of existing systems. ii. For the domains which are new and not much systems are available, the reference model is created by leveraging reference models of the domains that are close to the ones under consideration.

Second, the review revealed there are different SP2P systems and types (e.g. Piazza, Chatty Web, KEx, Somewhere, Hyperion, PeerDB, coDb, Esteem, Observer, Edutella). These systems are incompatible with each other, employ different architectures and were developed by professionals from different backgrounds.

In this study, we mainly followed the first approach – extract common components of existing systems and related works – to build the reference model for SP2P systems. That is, the SP2P reference model captures the common characteristics of many SP2P systems and other related works and provides normalized description of key concepts of these systems and related works. This is so because a large number of SP2P systems and types are available. More specifically, in order to obtain key constructs of the reference model, the primary focus was the identification of prominent or distinctive features of existing SP2P systems. The features are user-visible aspects (characteristics) of prominent SP2P systems and other related work. They define both common aspects of the SP2P systems as well as the differences between them. The applied method is known as a Feature-Oriented Domain Analysis(FODA)[56]. We also studied reference models for domains close to the SP2P systems. These include the reference model for Grid computing [91] and web services [61]. The reference models created in these studies represent class domains and they were obtained by following the first method of reference model creation that we have also followed in this study to create the SP2P reference model.

Several SP2P systems and prominent research works were chosen to extract the distinctive features of SP2P systems. Four of these systems (Chatty Web [3], KEx [16], P2PSLN [48] and Piazza [49]) and their corresponding component, component properties and component relationships are described below. We focus on these systems for their *contribution, novelty and types* (Chatty Web, Emergent semantics; KEx, Knowledge management; P2PSLN, Information retrieval; Piazza, Database). The detailed study of the selected SP2P instances demonstrate two important aspects of the described SP2P reference model. First, they illustrate our analysis and examination approach of the existing SP2P systems and second, the distinctive features of these systems contribute to the described reference model. The detailed information about component methods, properties, relationships, and class models of four SP2P systems are as follows:

3.1.1 KEx system

The component, component properties, and relationships of the KEx system are as follows:

- Peers

  In KEx, a peer manages a set of resources and own a unique *id*. A peer also has reference to other peers with similar knowledge and a schema description to be used for identifying peers with similar knowledge.

- Resources

  Each node stores context, data and metadata. The *context* is the perspective that peers have on the domain knowledge. Metadata refers to links to other resources and mappings to context stored at other peers. KEx uses XML-Schema specification to represent contexts. Special hierarchal notation called CTXML [18] is used for the context representation.

- Query maker

  KEx has a special component called *query maker*. The query maker is used to select concepts from local repository and to compose queries which are simple search messages. That is, queries are composed without the use of query specialized language. Query content is made of one or more local concepts and

context, called *focus* in KEx. The concepts describe the resource that the user needs to retrieve. The context explicates query content semantic; it is the entire path(s) from the concept(s) to the root of the concept hierarchy. For a detailed example on how concepts are selected, queries are composed and posed on the neighboring peers, the readers are encouraged to see [16].

- Peer federation and knowledge links

  In KEx, semantic neighbourhood takes two forms: peer federation and knowledge links. First, peer federation refers to peers with semantic compatible resources that agree to act as one entity, i.e., answer query requests from other peers, group together and form a federation. Peers form federations with their acquaintances (peers with similar schema). Second, *knowledge links* refer to peers that are able to discover the semantic relationship between their own resources and other peers' resources during query answering and mapping. Peers store the discovered information and use it to build a network of knowledge links. A peer p incorporates into its semantic knowledge network another peer $\bar{p}$, if concepts of peer $\bar{p}$ correspond, completely or partially (see [18]), to concepts of peer p.

- Semantic mapping

  KEx implements a highly expressive matching algorithm. It allows for the representation of relations between concepts at different abstract levels, i.e., synonyms, hypernym, hyponym, disjoint, and compatible relation. Mapping is performed on the service provider side, queried peer. Each provider, after receiving a query applies a runtime match to the query's focus, this way interpreting the query from its perspective. The matching algorithm compares both the syntax and semantics of query concepts to concepts of local context; a concept syntax is a concept label and concept semantic is a tree path from the concept to the root of the tree. The matching algorithm is based on WordNet use to resolve semantic mapping. That is, to determine the different abstract relation between concepts.

- Query routing

  KEx uses adaptive query forwarding strategy. Peers send queries to other peers

known through discovery method as well as peers known through previous interactions, i.e., knowledge links. A knowledge link refers to extra information – peer classification information – that a peer saves in their local context about other peers. Query forwarding is controlled using several query control forwarding policies. These include time-to-live (TTL), the number of hops or the list of peers already reached. Even though KEx does not have a policy for preventing queries from rotation around cycles in the network, one can see that implementing such a policy is feasible. This could be achieved through applying one or more cycle prevention policies, similar to query control forwarding policies applied in KEx.

- Query answerer

  In KEx the service provider peers match query content concepts against their local ontologies in order to come up with proper answers and the system user determines the correctness (incorrectness) of the query answer. This is so because what is considered to be correct by a provider is not necessarily correct by the querying peer. Once the system user decides on the correctness of the answer, the related document is provided directly to the querying peer. That is, in KEx querying peers receive answers to their queries directly from queried peers, service providers.

Figure 3.1 summarizes KEx's essential classes, class properties and their association that we have derived from the KEx system description.

3.1.2 P2PSLN system

The component, component properties, and relationships of the P2PSLN system are as follows:

- Peer

  A Peer in a P2PSLN system is an active and intelligent soft device that contributes data or information to overall system resources.

- Resources

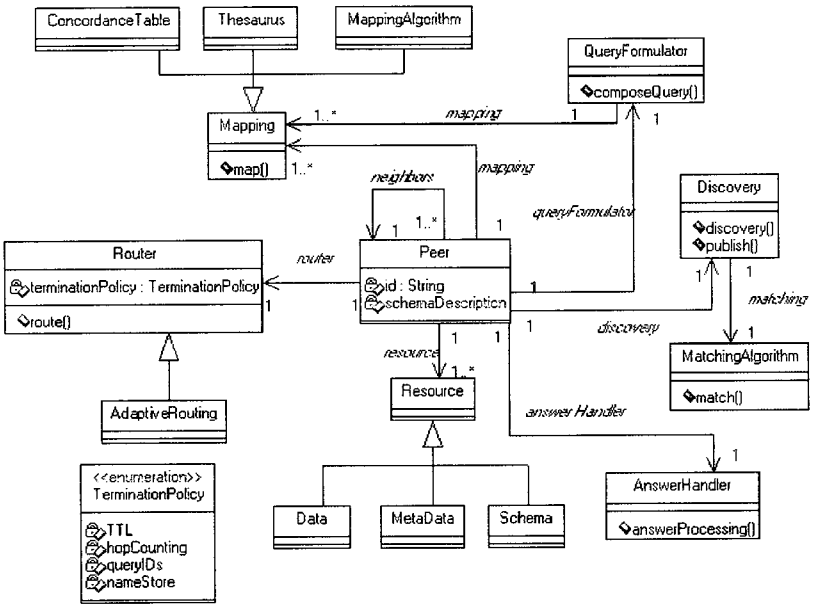

Figure 3.1: KEx classes, class properties and their association that we have derived from the description of the system

Resources in P2PSLN are XML files along with XML Schemas to describe and constrain the content of XML data files.

- Query formulator

  P2PSLN provides a graphical interface for posing queries on peers. Queries are composed using special query language that has been developed for P2PSLN. Query contents are simple text keywords.

- Semantic neighbourhood

  In P2PSLN, peers are linked to each other based on the semantic relation between their schema elements, schema structures and descriptions. That is, a peer p constructs a semantic link with another peer $\bar{p}$ if its XML schema elements, schema structure, and semantic description are related to peer $\bar{p}$. Semantic links have types and of which, there are eight. Link types represent semantic relation strength. Two peers have *equal* relation types – the highest strength relation – when their schema elements, schema structure and description are equal. The link relation type becomes *empty* when there is no semantic relation between two peers, and any other relation lies between equal and empty two link

types. In P2PSLN, peers autonomously select which other peers they are going to connect with to build their initial semantic relations. The initial relations are then followed by schema exchange and semantic mappings for furthermore connection relationship improvement.

- Semantic mapping

  P2PSLN provides three different semantic mappings, namely node, path and clique semantic mappings. The mapping considers not only peers' XML Schema, but schema structures as well. Mapping is carried out on the querying pee'r side. P2PSLN makes use of the global dictionary, WordNet, to implement mappings. The similarity degree between a set of terms, e.g. synonymy, abbreviations, that has been defined in the global dictionary indicates P2PSLN's support for a detailed and involving mappings. P2PSLN check mappings for corruption. Peers that participate frequently in answering queries but whose answers are incorrect are double checked for schema changes. If changes are detected, the querying peer updates its schema mapping, semantic link type, and the similarity degree between the two peers.

- Query routing

  P2PSLN uses adaptive query routing. That is, querying peers selects proper successors for query forwarding. Query forwarding is controlled using TTL query forward control policy. P2PSLN does not provide information on how it handles the issue of query rotations along the network cycles.

- Query answer evaluation

  In P2PSLN, query answers are evaluated for more than just whether or not they are correct. Query answer evaluation covers several aspects. The aspects are query response time, traffic overhead, precision, etc. Query results are returned directly to the querying peer by peers who hold the answers and the system user decides on the appropriateness of query answers.

The essential classes, class properties and their association that we have derived from the system description is summarized in Figure 3.2.

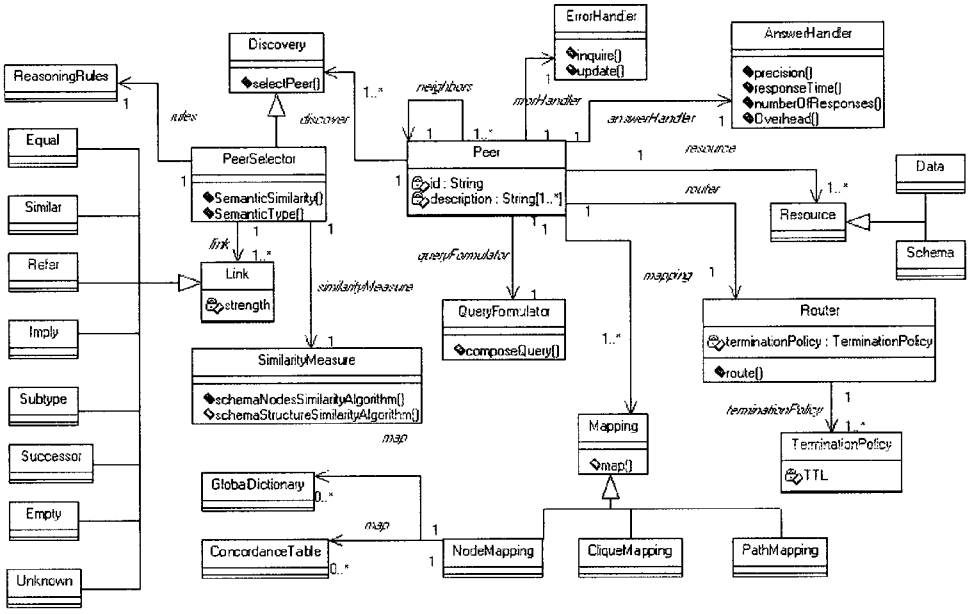

Figure 3.2: P2PSLN classes, class properties and their association that we have derived from the system description

3.1.3 Piazza system

The component, component properties, and relationships of the Piazza system are as follow:

- Peer

  Peers have unique ids, contribute resources to the overall system and select other peers to connect with. Piazza peers do not use *profile* matching to connect to other peers. This is so because Piazza uses P2P infrastructure but lacks the dynamic property of P2P networking.

- Resources

  In Piazza, the resources contributed by each peer include: 1. data instance, e.g. XML or RDF data instances, and 2. data models, e.g. XML schemas or OWL ontologies. Peers may also supply computed data, such as cached answers to queries.

- Query formulator

Piazza uses the XQuery language for composing queries. Queries in Piazza are always posed from the perspective of a given peer's schema, which defines the preferred terminology of the user. Queries are comprised of three constructs: variables, predicates, and equivalent class definitions. XQuery uses XPath expressions to bind XML nodes to the variables. A predicate specifies a condition for variable bounding; they are defined in the XQuery WHERE clause. The equality operators used by predicates in the queries specify the equivalence classes between query variables. Piazza uses OWL's owl:equivlentClass construct to define equivalency between two classes.

- Semantic neighbourhood

  In Piazza, a semantic neighbourhood of peer p comprises all nodes that are related to elements of p's schema by semantic mappings. When a new peer is added to the system, it will connect to a subset of the existing network peers that are semantically related. Peers are free to choose with whom they would like to establish semantic connections.

- Semantic Mapping

  In Piazza, mapping is carried out using queries (views). Piazza distinguishes between two different levels of mapping: data-instance mapping level and schema/ontology mapping level. At the schema level, mapping has two characteristics: 1. it is highly expressive, and 2. it has the ability to transform schema structures. Mapping can be highly expressive in that mapping may involve attributes or class correspondence, class containment, subsume, overlap, or disjoint. Support for transforming schema structure of one peer to those in a second peer is due to mapping's ability to combine multiple entries and to break a single entry into multiple fragments. At the data-instance level, mapping is based on the concordance table. The concordance table is assumed to be constructed using the existing matching algorithms and techniques. Mappings are directional and are carried out by both the query sender and the query receiver. Analyzing mapping for its correctness has been acknowledged by Piazza developers as an area that needs to be addressed in future work.

- Query routing

  Piazza employs adaptive query routing by posing queries only over semantic

neighbours. Given a query q posed over the schema of node p, Piazza rewrite q into a query $\bar{q}$ over the neighbors of p using local mappings. In Piazza, query forward termination is accomplished using mapping paths exploration. That is, query forwarding terminates when no useful paths remain to be traversed. Piazza does not adhere to the P2P network characteristics completely. For example, a Piazza network is cycle free. Hence, the problem of repeated query processing is a non-issue in Piazza.

- Query answerer

  Piazza relies on the system user to evaluate query answers. Thus, the system user determines correctness (incorrectness) of query answers. Piazza is concerned with two aspects of query answers: the percentage of total relevant answers that have been retrieved, i.e., *recall* rate, and the length of time that is needed for correct answers to be retrieved. In Piazza, answers are directly sent to the querying peers.

Figure 3.3 summarizes Piazza's essential classes, class properties and their association that we have derived from the system description.

3.1.4 Chatty Web system

The component, component properties and relationships of the Chatty Web system are as follows:

- Peer

  In Chatty Web each peer p has a unique id, maintaining a database according to a schema. Peers are able to identify their schema, either by explicitly storing it or by keeping a pseudo unique schema identifier which is obtained, for example, by hashing.

- Peer resources

  In Chatty Web, a resource could be a database, a website or set of files or documents a peer maintains in a P2P network. In its system data model, however, Chatty Web considers each peer to maintain a database according to

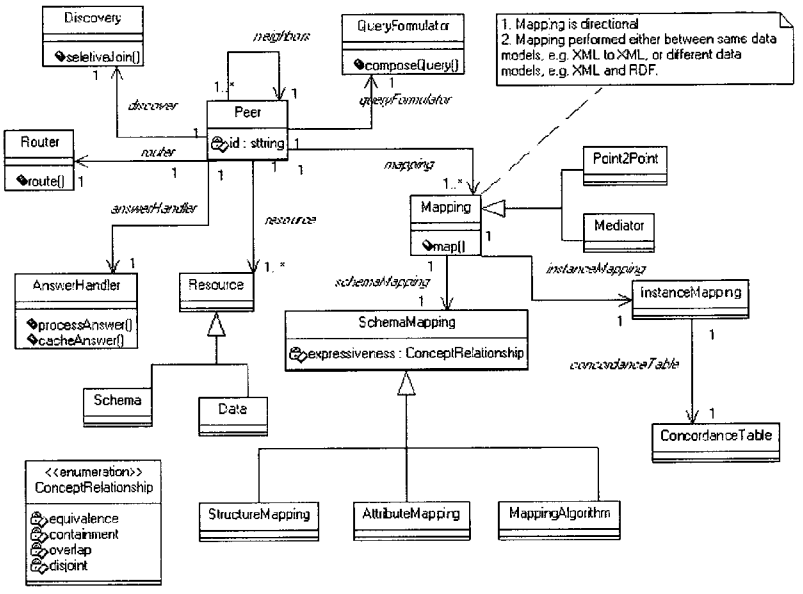

Figure 3.3: Piazza classes, class properties and their association that we have derived from the system description

its schema where the schema consists of a single relational table and the data that a peer stores consists of a set of tuples t_1, t_2, ... t_r of the same type.

- Query formulator

  Peers compose queries using their local concepts. Queries are basic relational algebra operators, e.g. selection, projection, and mapping, that is, specialized query encoding languages such as SQL or XQuery are not considered. Queries are issued to any peer through a query message. The query message holds a query identifier, query content, querying peer address and translation trace to keep track of the translations already performed.

- Semantic neighbourhood

  Chatty Web peers learn about each others' schema through sending, flooding networks with ping messages and receiving pong messages. Peers are able to learn about each others' schema because they incorporate their schema identifier into the pong message. Each peer maintains a neighbourhood n of semantically relevant peers. There are two types of peers in the neighbourhood of peer p: i. those that share the same schema with peer p and ii. those that have a

different schema. A peer p includes another peer $\bar{p}$ with a different schema into its neighbourhood n if it knows how to translate queries against its own schema to queries against the foreign schema.

- Semantic mapping

  Chatty Web regards mapping as an establishment of local agreements between peers. Chatty Web does not implement its own mapping component; rather, it relies on reusing exiting mappings. Chatty Web, however, does not put any restrictions on mappings, e.g. its expressiveness level, in order to be reused. Chatty Web envisions mapping as partial translations between schemas that is been carried out using views. That is, transformation operation provides a view of schema $SP2$ according to schema $SP1$. Hence, transformation is performed on the query sender side. Mapping incorrectness is counted for in Chatty Web and the standard maximum-likelihood technique is used for translation error rate estimation in the system.

- Query routing

  Information obtained by applying different quality mapping assessment is used to direct searches in the network; i.e., the adaptive query routing strategy is applied. Chatty Web avoids processing the same query more than once by incorporating query forwarding path information into the query structure. Query forwarding leads to query content changes. The change is the result of query transformation among peers with different schemas. Query changes are used for query forward termination. Query forwarding stops after a query becomes too different, either from a syntactic or a semantic point of view, from its original.

- Query answerer

  In Chatty Web, the querier peer evaluates query results automatically. An answer is considered to be correct if the received document conforms to the peer's local document categorization. Furthermore, Chatty Web calculates semantic relation values between query answer concepts and querying peer's local concepts to determine semantic mapping correctness along query forwarding cycle. Peers that provide answers send their answers directly to the querying peer.

Figure 3.4 shows Chatty Web's essential classes, class properties and their associations.

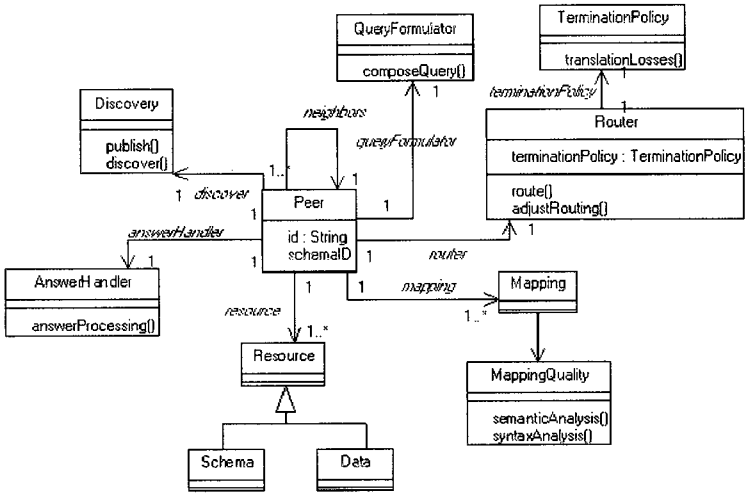

Figure 3.4: Chatty Web's essential classes, class properties and their associations

3.1.5 Edutella system

Edutella [81] enables sharing of learning resources among distributed and independent educational resource providers. Edutella uses the JXTA [1] protocol for its P2P network infrastructure and RDF[2] to describe resources. Queries are routed using the JXTA group construct. Edutella is not used for deriving the reference model constructs, Edutella's class model is described in [81]. We use Edutella for validating the proposed reference model(See Section 3.3).

3.2 SP2P reference model

There are many and diverse SP2P system realizations and architectures. This is primarily due to the involvement of a variety of researchers from different backgrounds into a still recent and evolving area. The proposed SP2P reference model meets the essential requirements of the generic architecture, i.e., it models the essential aspects of the existing systems. A particular SP2P system, e.g. Chatty Web, Piazza, can be considered an instance of the reference architecture. The model is a high level abstraction which hides implementation detail from the developers. However, it is

[1] https://jxta.dev.java.net/
[2] http://www.w3.org/RDF/

defined in a way which makes deriving concrete systems possible. Systems built based on this model should be easy to change and modify. The SP2P reference model is made of seven key constructs, *SP2P=<p, r, qf, sn, m, rt, qa>*. The constructs are Peers, *p*, Resources, *r*, Query Formulator, *qf*, Semantic Neighborhood, *sn*, Mapping, *m*, Routing, *rt*, and Query Answerer, *qa*. The seven constructs comprise the minimum components required for any SP2P system and it can be used for assessing SP2P system conformance. The collective behavior of the components Query formulator, Semantic neighbourhood and Routing together constitute the search process. Hence, the search process is not an independent and explicit component of the model. In the following, we describe each of these model constructs. That is, we define each model construct, explain how it works, and provide the justification for why the construct should be an element of the reference model.

In Table 2 brief definitions of some key model abstractions were provided for quick referencing. The model constructs are represented in UML classes and diagrams. This will enable transparent translation of model constructs into code using high level programming languages such as Java or C++.

3.2.1 Peers

A Peer *p=<id, r, f, n>* represents an active object or an entity in the network. Each peer has an identification, *id*, which uniquely identifies the peer in the network, a set of resources, *r*, that it manages, a profile, *f*, and a set of neighbors, *n*, i.e., references to other peers in the network. These elements are important and each peer in SP2P system needs to support. Peers unique identification enable various meaningful network operations including information retrieval from appropriate peers. Peers need to keep references to semantically related peer for efficient search, information exchange and cooperation. Examples of the resources that a peer could manage include sets of document files, bibliography files or learning objects that peers willing to exchange. More information about resources is provided when describing the resource construct of the model. The profile, on the other hand, is the description of peer's domain knowledge, expertise or services. For example, a subset of data model's key concepts could comprise peer's profile. Profile is needed for building semantic neighbourhood. In SP2P, peers compare their profile and connect to each other based on their profile similarity. For information on how peers use profile to connect with each other see the

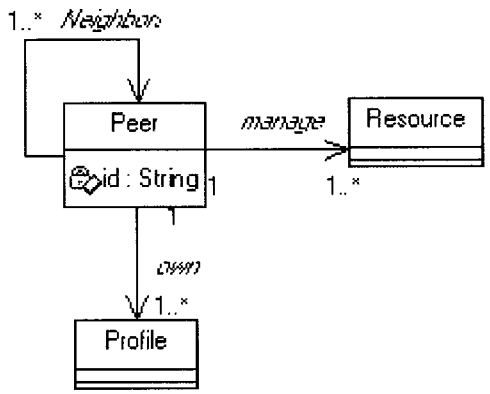

Figure 3.5: Peer construct

description of the semantic neighbourhood model construct described below. Figure 3.5 is a class view of a peer construct for the proposed SP2P reference architecture.

3.2.2 Resources

The Resources $r=<dm,\ i,\ md>$ are one of the fundamental building blocks of any SP2P system as such without resources there will be no meaningful SP2P system. Peer resources in SP2P systems comprise ontology or data model, *dm*, the actual data, *i*, and metadata, *md*, to improve search and content retrieval.

Peers could have their data represented in different data models. Examples of data model include Relational table, XML Schema, RDF data model, and OWL file. The choice of data modeling is important, and systems could be differentiated from each other based on the choice of their data model. This is due to the following two features of the highly structured data:

i. Support for Semantics The choice of data model determines data semantic transparency. Semantic transparency enables automatic machine processing of data as well as improving query result precision. For example, RDF and OWL data model languages have built-in support for defining and representing data semantics while other, XML and relational tables, do not support explicit data semantics representation. For example, when OWL language is used for data modeling, the system is capable of recognizing that "zip code" is same as "area

code", and "terminal" as a computer monitor is different than "terminal" as a station. This is possible because using OWL language a concepts is more than a label. It comprises of label, concept properties and relationships with other concepts.

ii. Support for Inferences The choice of data model determines the extent of the system's ability to answer queries. For example, data models such as RDF and OWL support knowledge *inferences*. Systems with these types of data modeling are able to answer queries where information is not explicitly stored in their repository. This might be difficult for other systems with different data models to do so.

Peers could bring only data model to the network, however, in most cases, peers share actual data instances, i, structured in a data model. Examples of resources artefacts include sets of documents files with a formally defined description. On the other hand, Meta-data, *md*, such as an ontology name space or peers knowledge about other peers' resources is a reference to external resources available on the network (see [20] for more information on reference to external resources). Meta-data enable resource reuse that are available somewhere on the net or stored in peers' repositories to improve resource sharing and freeing peers from storing large amount of resources locally.

Peers could share the content they bring to the network and/or the one's they gain from the network. In contrast to conventional P2P networks, resources in SP2P are neither replicated nor assigned to other peers in the network in order to be used by network peers for processing queries. However, in cases where SP2P is used for file sharing this characteristic or restriction might be relaxed. Figure 3.6 is a class view of the Resource construct for the proposed SP2P reference architecture.

3.2.3 Query formulator

In SP2P systems information is structured and formal and keyword based search is not meant to be the way search is conducted. Peers can query other peer's formally defined structured information. Hence, peers in SP2P systems need to support query formation. Query formulator qf=<*sc, cq, pq, q, l*>, often a graphical user interface

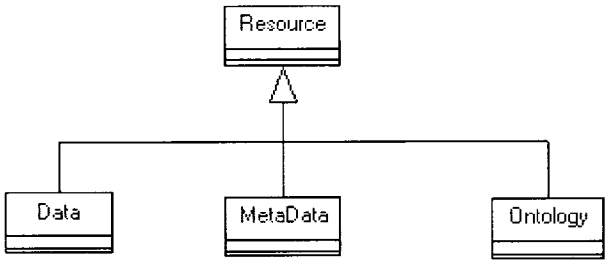

Figure 3.6: Resource construct

component, is a separate component on top of the resource layer which supports semantically enriched query formation. Peers use their own Query formulator component to *select concepts*, *sc*, from the local resource repositories, *compose queries*, *cq*, and *place queries*, *pq*, on the neighboring peers *n*.

Using the *select concepts* sub-component, peers can browse their local information to choose the desire concepts from the local repository for query formation. Once proper concepts selected for query, *compose query* procedure automatically adds the necessary background information to the query concepts. For example, if a concept *book* is selected for querying, the compose query procedure could automatically add to the query that book is a publication and it has properties such as author and publisher. The query compose procedure allow users to define query restriction as well. For example, when querying book, the user can define that she is only interested in books which are published after 2010. *Place query*, on the other hand, is about the actual query submission. Once proper query composed, place query enable users to trigger query submission.

Query objects, *q*, are diverse based on the system's endorsement for the query's explicit semantics, i.e., peer's data model (see subsection 3.2.2). For example, query could incorporate references to local or global ontologies for supporting query concept meanings or, when a tree-like data representation is used as a resource, e.g. XML format, a query concept could be replaced by a tree path to explicate concept meanings. A tree path refers to the concept, its ancestors, and descendant concepts.

Another important aspect relevant to the query formation is the need for using proper *query language*, *l*, to form queries. The choice of the query language restricts the

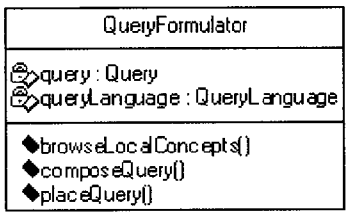

Figure 3.7: Query formulator construct

explicit semantic of query content, and an appropriate query language for SP2P systems is Sprql[3]. Figure 3.7 represents a class view of the Query formulator construct for the proposed SP2P reference architecture.

3.2.4 Semantic neighbourhood

SP2P network topology is unstructured and semantic based. Connected peers with compatible information resources comprise semantic neighbourhood, *sn*, and multiple connected semantic neighbourhoods comprise SP2P network. In SP2P systems, peers with compatible information resources connect and place query only on their neighboring peers *n*. Hence, the need for discovering and grouping together peers with compatible information resources, i.e., forming Semantic neighbourhood.

Semantic neighbourhood is a distinguishing characteristic of SP2P systems, and its creation involves decision making on some system aspects. These include choosing a method for connecting peers, either Autonomous joining method , *aj*, or Peer discovery method, *pv*, similarity function, *sim*, and the network connection degree, *d*, i.e., *sn*=<*aj*, *pv*, *sim*, *d*>. The description of the two popular methods used for forming a semantic neighbourhood can be summarized as follow:

Autonomous joining(*aj*): Peers autonomously select which other peers they are going to connect with. Peers are responsible for identifying semantically related peers, and constructing semantic mapping(s) between their own information resources or ontology and ontologies of related peers when their domain representations are different.

Peer discovery(*pv*): Peers exchange their profile *f* and use similarity function *sim* to

[3]http://www.w3.org/TR/rdf-sparql-query/

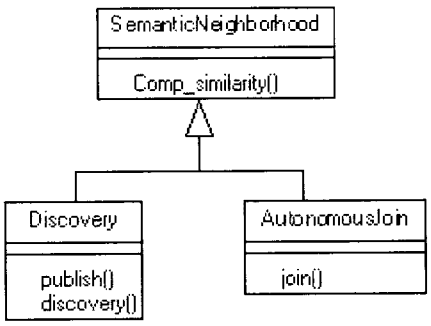

Figure 3.8: Semantic neighbourhood construct

discover-semantically related peers. The exchange of profiles can happen at network startup time or when new peers join an already established semantic based network. Peers interested in connection with other peers broadcast their profile and relevant peers respond to the querying peer by sending their *id* and profile. Querying peer computes the *strength* of the relation, i.e., the semantic affinity between the two profiles, and either accepts or drops the answer for connection. Peers can have only a limited number of connections. The *network degree d* represents this limitation in the model.

Peers need to apply only one of the two above described methods to create the semantic neighbourhood. The Autonomous joining method is suitable for stable SP2P systems in which peers invest large amount of resources to build mapping links with each other. Peer discovery method, on the other hand, is suitable for dynamic SP2P systems in which peers discover and connect to other peers based on their profile similarities. Figure 3.8 represents a class view of the Semantic neighbourhood construct for the proposed SP2P reference architecture. The above described two methods have a common behavior, compute profile similarity. The common behavior is been put in a common super class to be inherited by individual methods for easy implementation.

3.2.5 Routing

Routing *rt=<fs, ch, tp, lm>* is an essential component of any SP2P system. SP2P systems need Routing component for efficient search and content retrieval. The Routing component is responsible for delivering query q from the query initiator p_i, to one

or more query receiver in neighbourhood n and returning query answer. Query routing in SP2P systems involve decision making on three different design aspects of routing. These aspects are: i. query forwarding strategy, *fs*, ii. cycle handling, *ch*, and iii. routing termination policy, *tp*. Existing SP2P systems have different takes on these aspects. Below, each of these issues, their function and the importance, are described.

Forwarding strategy (*fs*) Regards the spreading queries in the SP2P system and there are different ways to do that. These include: flooding, expertise-based selection, adaptive query routing [64, 104, 110, 112]. These strategies are different from each other, among other things, on their usage of number of messages (queries) and time efficiency in retrieving query answers. Adaptive query routing (see e.g [64] for discussion on adaptive query routing strategies), is the most widely used technique for routing queries in SP2P systems. Adaptive query routing is widely used because of its efficiency to retrieve query answers. SP2P systems with an adaptive routing strategy utilize *learning* techniques to enable efficient routing, in other words, peers use their past interaction experience to determine future query routing. In this regard, each peer could consider only its *own experience* in making decisions on future routing, or in addition to its own experience, it could make use of other peers' *recommendation* as well. The central idea in the adaptive strategy technique is to make use of the extra information existing in the network, e.g. information about the peers that most likely provide correct query answers, to send queries only to the most relevant peers (experts).

Adaptive routing could also involve relationship management, *lm*, where relationship management refers to cutting (preserving) relationship with peers that provide incorrect (correct) answers repeatedly. A peer p could decide cease sending queries to a peer $\bar{p}$ in its neighbourhood after receiving several incorrect answers from that peer. In Section 5.5.2, we present different strategies for dealing with incorrect query answers and when a peer should cut ties with peers in its neighbourhood.

Cycle handling (*ch*) Another important issue of querying SP2P systems is how to deal with *query repetitions*, i.e., already seen queries. A peer may receive the same query from *different paths* or via a *cycle* in the network. Alternatively, a peer could receive a more specific query (or a more general one) via different paths or cycles in the network after multiple translations by semantically related peers.

SP2P systems need to mange query cycling. This is because, the way repeated queries are dealt with has an impact on the *number of query message* exchanges and *result completeness*. While terminating already seen queries possibly preclude the opportunity to provide some important answers, processing repeated queries increases the number of query messages a system exchange and results in generating a large number of repeated query answers. Query repetitions are dealt with commonly by using either query unique identifiers (*qid*) and/or query path information (*path*). In both cases, additional information is attached to the query to be used by peers for discarding repeated queries.

Query termination policy (*tp*) When query forwarding is going to stop is another important matter of routing queries in SP2P systems. Current common techniques for stopping query forwarding depend on either counting the number of *hops* or setting the query time-to-live (*TTL*). Using the hop-counting approach, a system administrator sets the length of a network path that a query message could traverse before terminating. On the other hand, the TTL approach is time based; a query message could traverse the network for the period of time that is specified in the query. As a query message traverses the network, its TTL value decreases. When the TTL value becomes zero, message forwarding stops. Note that these techniques have an impact on the query results that could be obtained. For instance, peers will continue to forward queries to their related neighbors even when they already have answers to the query as long as the specified constraints permit. As a result the number of query results will be affected. Yet, another query forwarding termination policy is to use query content to decide whether the Query should be forwarded or not. Using such a policy, queries will not be forwarded when their content becomes empty. Query contents become empty as result of concepts dropping during translation process. Based on the dropping/not dropping uncomprehended Query concepts during Query translation, we divided SP2P systems into two groups:Irreducible SP2P system(IRSP2P) and Reducible SP2P system (RSP2P) and they are defined as follows:

Definition 1 IRSP2P *system is an SP2P system with the property that it does not discard uncomprehended query concepts during query translation and forwarding among multiple peers.*

Definition 2 RSP2P *system is an SP2P system with the property that it does discard uncomprehended query concepts during query translation and forwarding among*

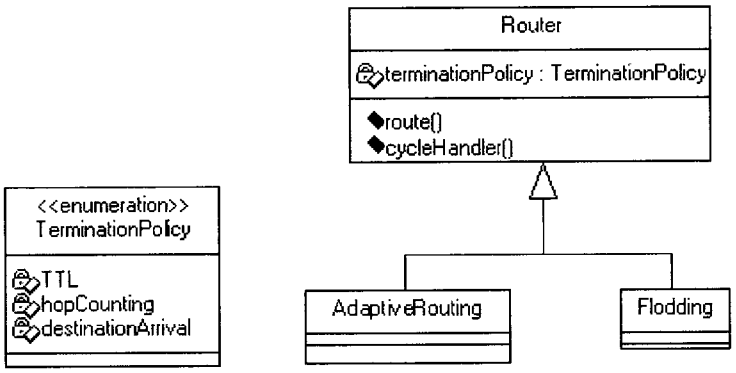

Figure 3.9: Routing construct

multiple peers.

Piazza system, for example, belongs to IRSP2P group and Chatty Web is an instance of RSP2P systems. Figure 3.9 represents a Router Class and its associated forwarding policy.

3.2.6 Query answerer

In SP2P systems query answers need to be evaluated for their correctness and proper query answer need to be selected when multiple query answers generated from a single query. Hence, the need for the Query answerer component, *qa*, in the SP2P reference model. In other words, the Query answerer component of the reference model *qa*=<*ae, qd, as, pai, av*> is required for dealing with two important aspects of query answers: i. query answer evaluation, *ae*, and ii. query answer selection, *as*.

Löser et al. [64] suggest that the query result evaluation strategy, among others, is an important aspect of adaptive query routing in semantic overly networks. Furthermore, we believe that for the SP2P networks to remain connected, they need to employ correct result evaluation function. Incorrect evaluation functions could prevent semantically related peers from teaming-up together. This in turn, based on the working application, could have far reaching consequences on the performance and dependability of the system.

The way query answer evaluations are *determined* , *qd*, is an important design issue in

SP2P systems. Answer determination could be attained manually or automatically. In manual query answer determination, Query answerer component present answers to the user, and the system user decide on the correctness (incorrectness) of query answer. Automatic query answer determination is about the system peer's ability to conclude the query answer's correctness (incorrectness). In the latter case, the system designer needs to design a set of criteria to empower SP2P systems with the ability to decide on the correctness (incorrectness) of query answers. An example of such measurement includes calculating the semantic relation between query answer concepts and query concepts.

Answer selection *as*=<*ap, lp, w*>, on the other hand, defines a set of criteria for selecting an answer when multiple correct answers are generated for a single query – each from a correct translation sequence. This could include the *precision of the answer*, *ap*, the *length of mapping path*, *lp*, and *peer weight*. Peer weight refers to the level of trust the querying peer has in the peers participating in the result generation and is represented as, *w*, in the model. The length of the query path and answer precision could be computed directly from the query answer. Peers' weight, on the other hand, is collected through prior peer encounters or from peers' recommendations.

Another important element of query answer handling is peers' capacity on *partial answer integration*, *pai*. Some query results might be partial answers, hence, the need for the peers' ability to integrate multiple partial answers. That is, peers need to be able to combine partial answers and give a uniform view of the results to users and other peers.

Answers could *arrive*, *av*, to the querying peer either directly or indirectly. Direct answers are those answers that responding peers provide directly to the querying peer and without passing through intermediary peers. Indirect answers refer to those that travel along query mapping path to reach the querying peer.

Query answer arrival depends on the routing behavior, i.e., the way query answers routed back to the query initiator. Therefore, query answer arrival could be part of the Router component. In our reference model, however, we decided to present answer arrival as a part of the Query answerer component for its relationship with the query answer. Figure 3.10 is a Query answerer construct of the proposed SP2P Model.

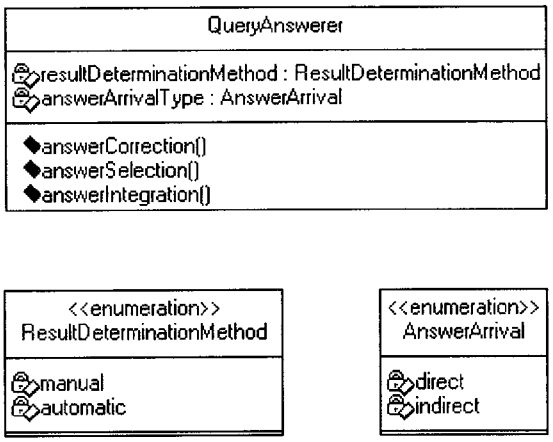

Figure 3.10: Query answerer construct

3.2.7 Mappings

The semantic mapping *m=<me, mi, mc, mw, mm>* refers to semantic relationship between concepts from independent information sources (ontologies). It is a fundamental design building block for any SP2P system, and a topic undergoing in-depth research. Mapping is a fundamental component of SP2P systems because peers' information representations are heterogeneous. Peers' local ontologies are developed independently reflecting peers' knowledge, interest and culture. Hence, the need for mapping component in the reference model to describe design aspects related to concept translation between ontologies. Using semantic mapping with SP2P systems involves decision making on various issues including mapping expressiveness, *me*, mapping implementation, *mi*, mapping correctness, *mc*, mapping ownership, *mw*, and mapping maintenance, *mm*. Below, a short description of each of these mapping constructs is highlighted. Furthermore, SP2P systems may support query reformulation, i.e., splitting the query, or completely reordering the query in a totally different but equivalent query. In such cases, the mapping component needs an additional query evaluator procedure to support query evaluation and reformulation.

Mapping expressiveness(*me*): Semantic mapping in its simplest form could be just a matter of finding query concept synonyms among different ontologies. In more expressive mappings, logical relations are used for finding relationships among concepts, concept properties and attributes.

The set of logical relations commonly used to define relationships among the peers' ontology concepts are $\{\equiv, \sqsubset, \sqsupset, *, \perp\}$. In this case, $c_1 \equiv c_2$ means that the two concepts are equivalent. In other words, c_1 and c_2 are different concepts with similar or identical meanings and are interchangeable. For example, *notebook* and *laptop* are equivalent concepts. The relation $c_1 \sqsupset c_2$ means c_1 is hypernym of c_2. That is, c_1 is more generic or broad than c_2. For example, the *system software* concept is more generic or broad than the *operating system* concept. The relation $c_1 \sqsubset c_2$, means that the c_1 have a hyponymyous relationship with c_2, i.e., c_2 is more generic or broad than c_1. For example, the *book* concept is less generic than the *publication* concept since there are publications other than book. The relation $\perp$ means that two concepts have no semantic relation with each other. For example, *bank* as financial institution and *bank* as river-bank have no semantic relation. Any other relations between concepts other than those described above can be captured by the $*$ relation.

Mapping expressions have an effect on the extent of query results. They could increase or decrease the extent of query results based on the permissible logical expressions of the mappings. Systems demanding exact mappings could relax some of their constraints (i.e., allow for less restricted mapping logics to take place) to increase query recall. For example, let us assume that the *University* concept from one ontology and an *Educational Institute* concept from a second ontology are synonymous, i.e., University $\equiv$ Educational Institute and mapping operation returns a value equal to 1, *map*(University, Educational Institute)=1.0. This assumption is valid since both concepts can be mapped to a common concept, *Institute*. Furthermore, let us assume that the semantic relationship between University or Educational Institute to a Research Institute has been defined as a related, i.e., the operation *map*(Research Institute, Educational Institute)=0.25. Now let us consider the following query been posed on either ontologies.

```
Query: list the name of all Research Institutes in the area.
```

The restricted query result will be null, since no synonymous relationship between *Research Institute* and *University* or *Educational Institute* can be asserted. However, if we relax the restriction of the query, i.e., instead looking for the exact relationship between Research Institute and Educational Institute concepts we satisfy with the relationship, the result of the previous query will be a set of University names

which they might carry out some research. This is because the relationship between Research *Institute* and Educational *Institute* will be asserted, both Research *Institute* and Educational *Institute* are *Institute*.

Mapping implementation(mi) How mapping is carried out is an important design issue. Peers could use a local copy of thesauruses such as WordNet, build their own dictionaries, construct mapping tables or when feasible, exchange ontologies (schemas) to translate concepts between ontologies. The choice of the approach to carry out mapping is affected by the scope of the application. For small and domain specific applications, peers could exchange local ontologies or build their own local dictionaries for translation. Larger applications on the other hand, may require local thesauruses which are capable of performing some inference rather than just simple concept-to-concept mappings associated with local dictionaries and tables. Mappings could be carried out automatically, semi-automatically or manually.

Mapping correctness measurement(mc) Correct semantic mapping is fundamental to SP2P systems. Various research efforts have been devoted to the classification of possible types of faults, measuring the quality of the mapping and estimating of information loss during query translation. The correctness of mapping is measured in two different ways: numerical and logical measurement. Numerical measurement pertains to the numerical values returned from the mapping tool. For example, a mapping operation could conclude that the semantic relationship between a *Laptop* concept and a *Notebook* concept is equal to 1.0: map(c1, c2)=1.0, and the semantic relationship between an *Operating system* concept and a *Software* concept is equal to 0.5: map(c3, c4)=0.5, or some other values. A detailed example related to the numerical values use in the mapping operation can be found in section 5.4.3. If the numerical value returned from a mapping operation is $\geq \delta$ (threshold), mapping is considered to be correct. The numerical values associated with semantic relationships between ontology concepts are a system designer decision. For example when a concordance table is used by an SP2P system for mapping, the values assigned to the relationships between any two concepts in the table will be declared and latter used in the mapping process.

The logical measurement, on the other hand, is the logical relationships that have been concluded during the mapping operation, that is, whether or not the relationship between two concepts satisfies one of these logical operations $\{\equiv, \sqsubset, \sqsupset, *, \perp\}$. For

example, the logical relationship between *publication* and *book* is $\sqsupset$. The two methods could be modified such that the logical relation could return numerical values and vice versa.

Mapping ownership (*mw*) An important decision that SP2P system designers must to decide is who (i.e., sender or receiver peer) is going to carry out the mapping. That is, whether query translation takes place *before* sending the query or *after* receiving the query. This is important because it will have an effect on query routing, to the extent that the querying peer will first perform mapping and then submit to only semantically related peers (i.e., if the outcome of mapping is above a certain threshold). This constraint can be used as a strategy for terminating query forwarding. Since the receiving peer performs mappings after receiving a query, this means that any query could be posed to any peer (i.e., there is no restriction on query forwarding). Query receiving peers either answer queries (i.e., if they could translate them to their local representation), or forward them to some other peers.

Mapping maintenance (*mm*) Recently, several studies have focused on the mapping maintenance issue and its effects on the SP2P systems reliability [6, 23, 70, 73]. These studies have concluded that mapping between different ontologies (schemas) need maintenance. This is because mapping could become outdated as a result of ontology changes. Outdated mapping puts the entire system at risk of failure. Hence, there is a need for 1. semantic mapping maintenance, 2. mapping corruption detection, and 3. mapping corruption tolerance: Mapping maintenance is needed to prevent it from corruption, corruption detection is required so it can be fixed, and lastly, mapping corruption tolerance is necessary in order to limit the level of the damage that mapping corruption may have done to the system. Figure 3.11 is a class view of the Mapping construct for the proposed SP2P reference architecture.

In Figure 3.12 all model constructs are put together and the model class diagram has been created. The class diagram shows the dependency between model components. The model diagram encompasses, among others, four enumeration classes. The SP2P implementer may choose one or more of the enumerated options and ignore others. For example, an SP2P implementer may decide to only implement "hopCounting" and not "TTL" or "destinationArrival" to stop query forwarding. The sequence of interaction between model components during query processing is represented in Figure 3.13.

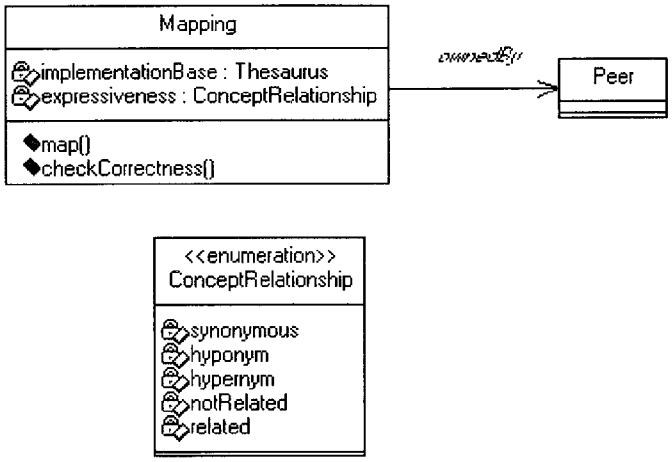

Figure 3.11: Mapping construct

3.3 Model applicability and validation

In order to show the model applicability, the system architectures described in this chapter (i.e., KEx, P2PSLN, Piazza, and Chatty web system architectures) are mapped onto the reference model. Individual systems are checked on whether they comply with the generic model and how they implement the generic features. The mapping tables (Tables 3.1, 3.2, 3.3, and 3.4) demonstrate that the described state-of-the-art systems posses the model's core component, however, they are different on the implementation of the component properties and component relations. The tables also manifest the comparative advantages (disadvantages) that the systems have over one another in relation to the model.

Furthermore, Table 3.5 illustrates that the identified components, features and properties of the reference model exists in the Edutella system as well. Edutella system was not used for deriving the reference model components, yet it has the components and functionality need in SP2P systems. In other words, Edutella demonstrates that the described reference model being valid and rigorous.

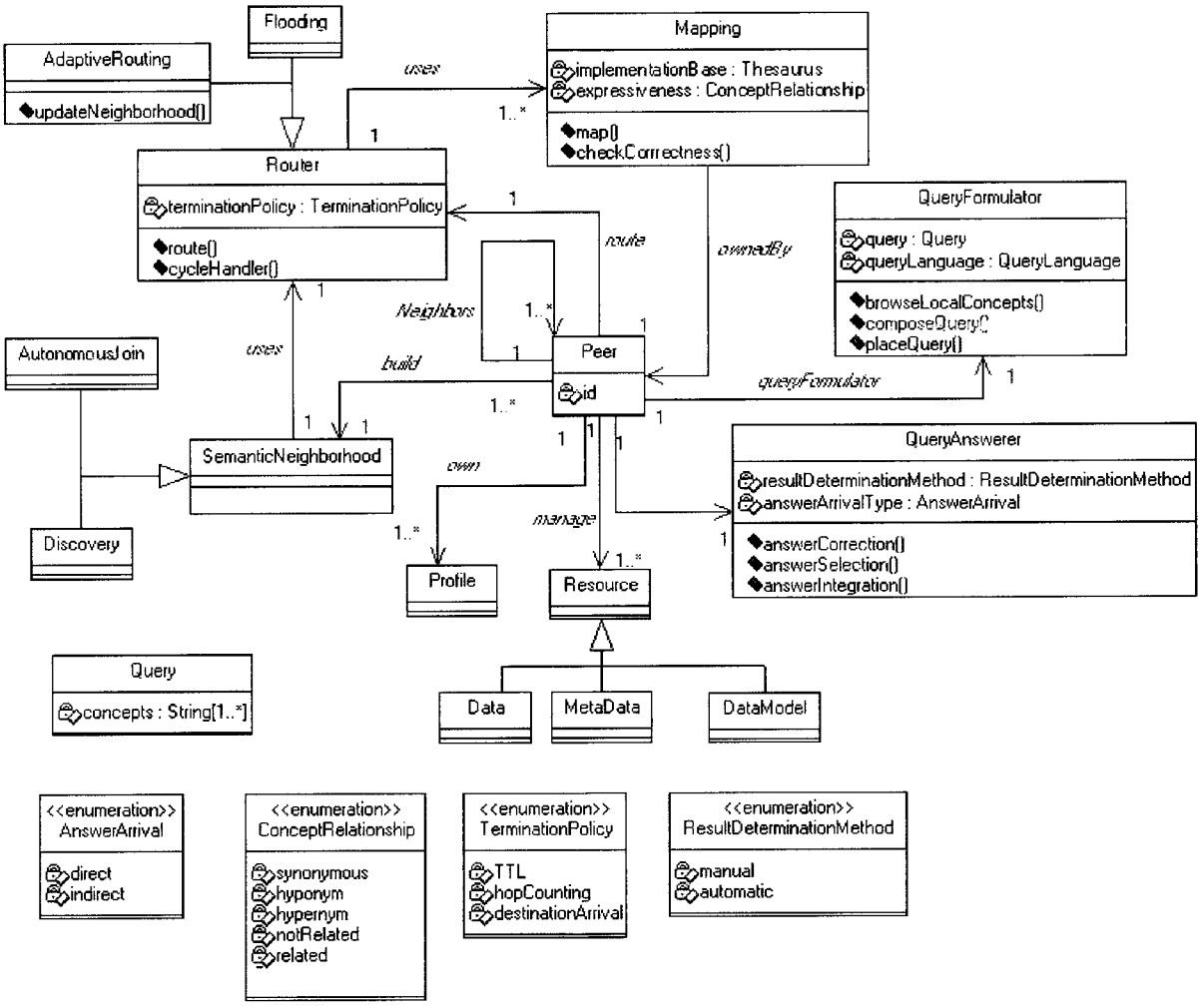

Figure 3.12: SP2P domain model

KEx system				
Peer	id	profile	neighbors	
	✓	✓(implicit)	✓	
Resource	data model	metadata	data instance	
	✓	✓	✓	
Query formulator	Select concepts	Compose queries	Place queries	
	✓	✓	✓	
	Query language			
	X			
Semantic neighbourhood	Autonomous joining	Peer discovery	Network degree	
	X	✓	X	
	Similarity function			
	✓			
Mapping	expressiveness	implementation	correctness	
	✓(high)	✓(WordNet)	X	
	ownership			maintenance
	✓(receiver)			X
Routing	Query forwarding	Cycle handling	Query termination	
	✓	✓	✓	
Query answerer	evaluation	selection	determination	precision
	✓	X	✓(manual)	X
	recall	integration	arrival	
	✓	X	✓(indirect)	

Table 3.1: Concepts and relationships manifested in KEx system

P2PSLN system				
Peer	id	profile	neighbors	
	✓	✓	✓	
Resource	data model	metadata	data instance	
	✓	✓	X	
Query formulator	Select concepts	Compose queries	Place queries	
	✓	✓	✓	
	Query language			
	✓			
Semantic neighbourhood	Autonomous joining	Peer discovery	Network degree	
	X	✓	X	
	Similarity function			
	✓			
Mapping	expressiveness	implementation	correctness	
	✓	✓(global dictionary and concordance table)	✓	
	ownership	maintenance		
	✓(sender)	✓		
Routing	Query forwarding	Cycle handling	Query termination	
	✓	✓	✓(implicit)	
Query answerer	evaluation	selection	determination	precision
	✓	X	✓(manual)	✓
	recall	integration	arrival	
	X	X	✓(indirect)	

Table 3.2: Concepts and relationships manifested in P2PSLN system

Piazza system				
Peer	id	profile	neighbors	
	✓	✓	✓(implicit)	
Resource	data model	metadata	data instance	
	✓	✓	X	
Query formulator	Select concepts	Compose queries	Place queries	
	X	✓	✓	
	Query language			
	✓			
Semantic neighbourhood	Autonomous joining	Peer discovery	Network degree	
	✓	X	✓(implicit)	
	Similarity function			
	X			
Mapping	expressiveness	implementation	correctness	
	✓(high)	✓(concordance table)	✓	
	ownership	maintenance		
	✓(sender/receiver)	X		
Routing	Query forwarding	Cycle handling	Query termination	
	✓	✓	✓(implicit)	
Query answerer	evaluation	selection	determination	precision
	✓	X	✓(manual)	✓
	recall	integration	arrival	
	✓	✓	✓(indirect)	

Table 3.3: Concepts and relationships manifested in Piazza system

Chatty Web system				
Peer	id	profile	neighbors	
	✓	✓	✓	
Resource	data model	metadata	data instance	
	✓	✓	X	
Query formulator	Select concepts	Compose queries	Place queries	
	X	✓	✓	
	Query language			
	✓			
Semantic neighbourhood	Autonomous joining	Peer discovery	Network degree	
	X	✓	X	
	Similarity function			
	✓			
Mapping	expressiveness	implementation	correctness	
	✓(low)	✓(reuse existing mapping)	✓	
	ownership	maintenance		
	✓(sender)	X		
Routing	Query forwarding	Cycle handling	Query termination	
	✓	✓	✓	
Query answerer	evaluation	selection	determination	precision
	✓	X	✓(automatic)	X
	recall	integration	arrival	
	X	X	✓(direct)	

Table 3.4: Concepts and relationships manifested in Chatty Web system

Edutella system				
Peer	id	profile	neighbors	
	✓	✓	✓(implicit)	
Resource	data model	metadata	data instance	
	✓	✓	X	
Query formulator	Select concepts	Compose queries	Place queries	
	X	✓	✓	
	Query language			
	✓			
Semantic neighbourhood	Autonomous joining	Peer discovery	Network degree	
	X	✓(implicit)	X	
	Similarity function			
	X			
Mapping	expressiveness	implementation	correctness	
	✓	✓	X	
	ownership	maintenance		
	✓(sender)	X		
Routing	Query forwarding	Cycle handling	Query termination	
	✓	✓(implicit)	✓(implicit)	
Query answerer	evaluation	selection	determination	precision
	✓	✓	X	X
	recall	integration	arrival	
	X	✓	✓(indirect)	

Table 3.5: Concepts and relationships manifested in Edutella system

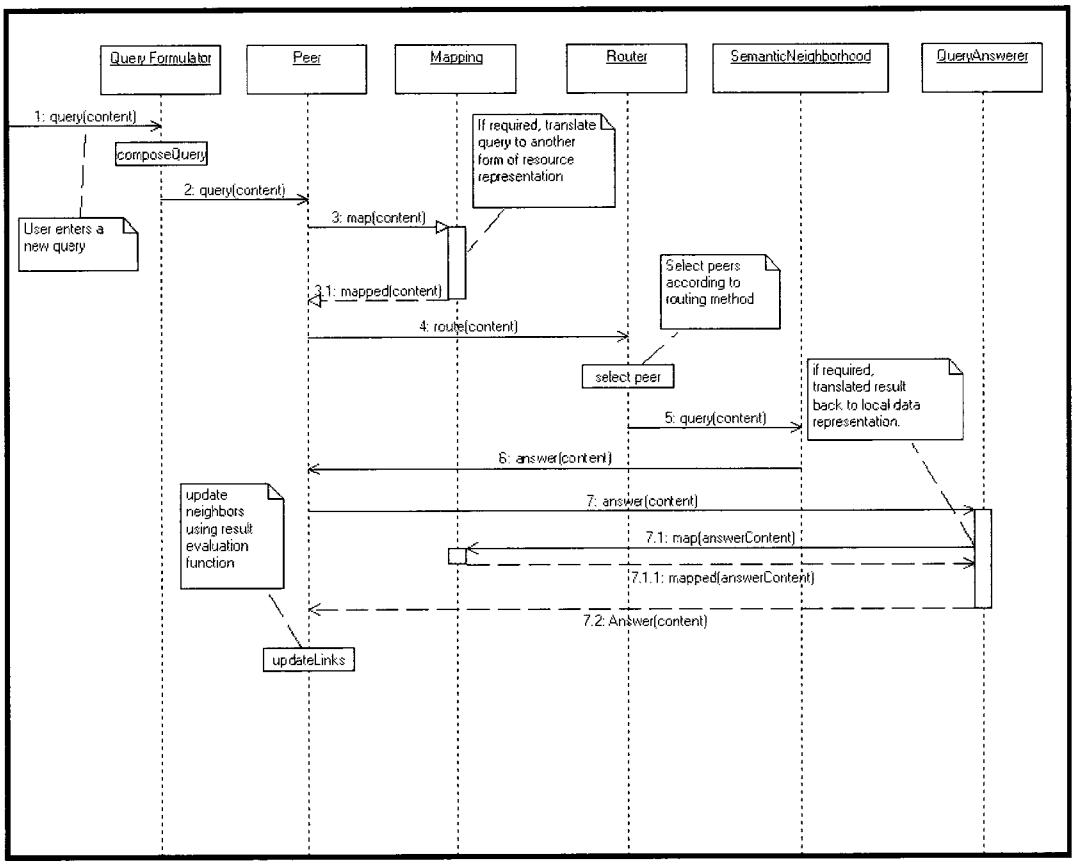

Figure 3.13: The sequence of interaction between model components during query processing

Chapter 4

SP2P:sim A semantic P2P simulation framework

SP2P:sim is a simulation framework based on the reference model. The framework is created by translating reference model components into implementation. For each individual construct from the reference model a correspondence class or multiple classes are created. The framework is generic which enables different SP2P systems to be simulated. Individual system are simulated through putting together the necessary system components from the framework. In this chapter an overview of SP2P:sim's design and features are described. The implementation of the key abstractions and their functionalities which correspond to the SP2P model specification, introduced in chapter 3, will be presented and discussed as well.

4.1 Introduction

SP2P:sim is written in Java language and uses several open source packages. These include Repast[1], Protégé[2], Jena[3], JavaFreeChart[4]. Specifically, SP2P:sim uses Repast's event scheduling, visual display and network components (see section 4.2 for more on Repast).

[1]http://repast.sourceforge.net/repast_3/index.html
[2]http://protege.stanford.edu/plugins/owl/api/
[3]http://jena.sourceforge.net/
[4]http://www.jfree.org/jfreechart/

SP2P:sim 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[5].

SP2P:sim is an event based simulation. In SP2P:sim, the execution of the control loop represents a single event. The simulation time clock advances by the length of time since the last event, and the execution of the loop will not occur when there is no event to process. Figure 4.1 is the execution flow chart of the SP2P:sim.

The rest of this chapter is organized as follows: In Section 4.2 repast aspects related to the SP2P:sim are described, Section 4.3 describes how the simulation framework is derived from the reference model, and finally, an overview of SP2P:sim packages and their classes are described in Sections 4.4 and 4.5.

4.2 Repast components reused

Utilizing Repast simulation has provided us with the necessary elements to avoid setting up the basics of a simulation, thus effort was put on developing the specifics of the SP2P:sim framework.

The REcursive Porous Agent Simulation Toolkit (Repast) is an open source API originally developed at the University of Chicago's Department of Social Science Research Computing Laboratory and now managed by Repast Organization for Architecture and Development. Primarily, it is a collection of objects and a model that sets up and controls the execution of these objects' behaviors according to a schedule. Repast is a discrete event simulator whose measure unit of time is a tick. Ticks are merely a way to order the execution of events relative to each other. In Repast, *events* or the some set of object behavior can be scheduled and executed at every tick. The *Gui classes* provide the capability for the graphical visualization of the simulation.

[5]The Beta version of the SP2P:sim code and the simulation's class hierarchy document are available for downloading at http://www.scs.carleton.ca/~armyunis/sp2p/

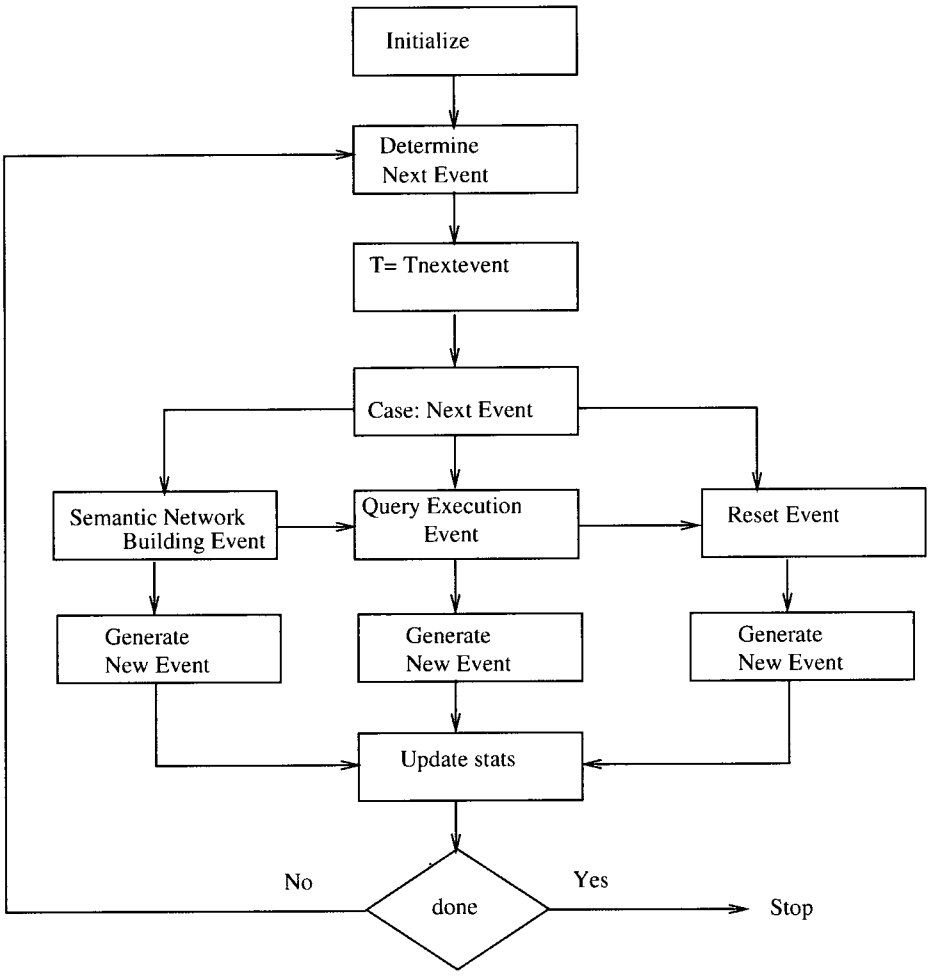

Figure 4.1: Repast's event flow chart as used by SP2P:sim

Repast also includes several varieties of charts for visualizing data (e.g. histograms and sequence graphs) and can take snapshots of running simulations. The *Network package* contains the core classes used to build network simulations. These include Node and Edge classes. A Node can determine its incoming and outgoing Edges and an Edge has knowledge of its source and target Node. In building SP2P:sim the Repast's event scheduling, visual display, and network components have been reused. More information on Repast can be found at [84].

4.3 Reference model and simulation framework parallelism

This section describes how the simulation framework is derived from the reference model. Table 4.1 shows the constructs from the reference model and their correspondence implementations from the simulation framework. For each individual construct from the reference model a class or multiple classes are created. For example, QueryAnswerer class and its derived subclass, GQA, TRQA and MVQA classes, from the simulation framework correspond to the QueryAnswerer construct and some of the possible behaviors of the QueryAnswerer class from the reference model.

Figure 4.2 is a class diagram for the SP2P:sim framework. The class model not only reveals the class correspondence between reference model and the simulation framework, but also presents all the additional classes which are created in order to implement a complete behavior or a particular design choice of a reference model construct. The additional class are shown in the light blue colors in the SP2P:sim class model diagram. Detailed implementation class discussion for each individual reference model construct and its corresponding implementation is provided in the section below. We first, however, describe the SP2P:sim's code organization.

4.4 SP2P:sim packages overview

Class packaging is helpful in code organization by grouping together classes with similar functionality. SP2P:sim code is partitioned into three main packages: *Peers*,

A reference model construct	**Simulation framework implementation correspondence**
Peer	Peer
Resource	Resource, ResourceAllocation, RandomAllocation, ComputedAllocation, ontologies, loadOWl, localOntologies
Profile	Profile, fromAFile, fromMultipleFiles, fromLocalData
QueryFormulator	QueryFormulator
Query	Query
Semantic Mapping	Mapping, Match procedure
Semantic Neighborhood	neighborhoodBuilder, publishDiscover, randomMeeting, anEdge
Query Routing	QueryRouting
Query Answerer	QueryAnswerer, GQA, TRQA, VotingQA

Table 4.1: Reference model classes and their implementation correspondence from the simulation framework

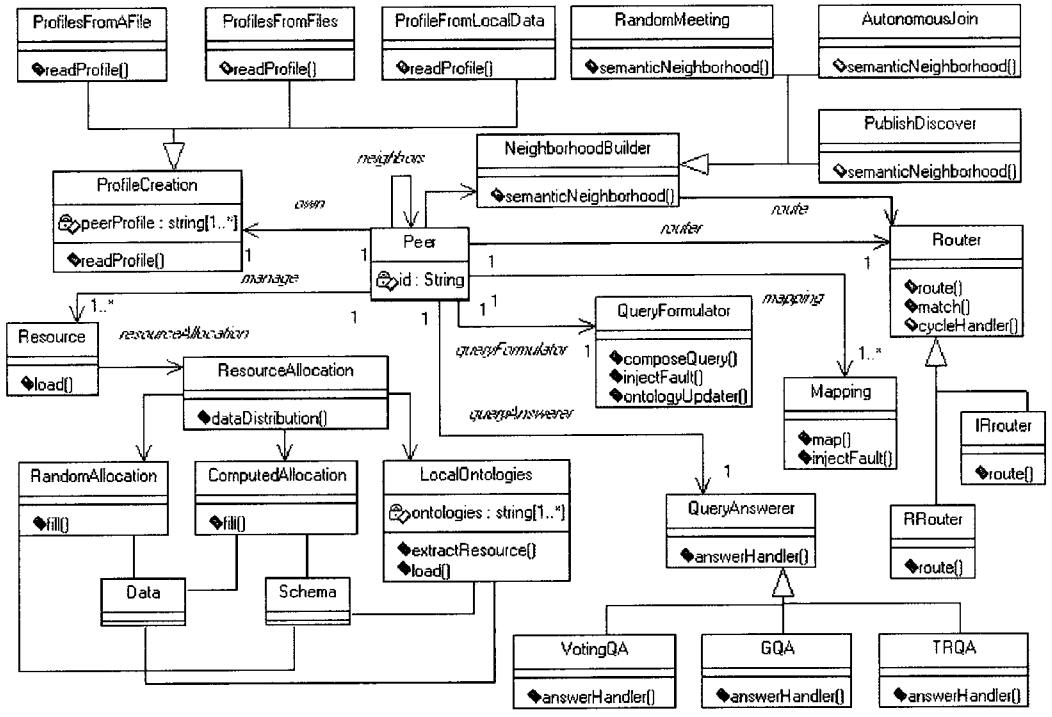

Figure 4.2: SP2P domain model implemented in the SP2P:sim framework

Semantic neighborhood, and *Util* packages. The Util package in turn contains two sub-packages, *Profile creation* and *Resource distribution*, and the Semantic neighborhood package contains a *NetworkBuilding* package. An overview of the SP2P:sim packages is provided in the following subsections.

4.4.1 Peers package

Some of the classes which are associated and managed by peer class are grouped together in the Peers package. These include anEdge, Query, QueryFormulator, and QueryAnswerer classes. The three QueryAnswerer class implementations, TimeR-Query Answerer, GTitQueryAnswerer and MVQuer Answerer, are also part of this package. Classes in this package are used for creating queries, posing queries and evaluating answer results.

4.4.2 Semantic neighborhood package

Classes in this package are used for creating semantic neighborhood, routing queries and mapping queries. The classes responsible for creating Semantic neighborhood are grouped together in a separate NetworkBuilding sub-package which holds AutonomousJoin, NeighborhoodBuilder, PublishDiscover, and RandomMeeting classes.

4.4.3 Util package

This package contains classes for resource distribution, profile creation, result analysis, and simulation launch. Classes for resource distribution and profile creation are grouped together in separate sub-packages: Resource Distribution and Profile Creation respectively. The class SimulationMain is the place where the simulation reads in the configuration file parameters and starts the simulation. The Result Repository class is the place where all simulation run results are stored and experimental figures are drawn. Figure 4.3 represent the package view of the SP2:sim.

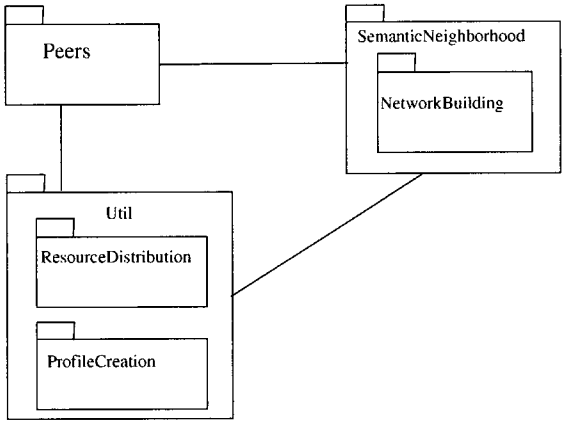

Figure 4.3: SP2P:sim package view

4.5 Inside SP2P:sim

In this section, implementation details of some key SP2P:sim components are described; some of the classes in the packages are considered in detail, and when applicable, different implementations for a particular component are discussed.

4.5.1 Assigning ontologies to peers

Securing unique data for each peer or a group of peers when the number of ontologies is less than the number of peers in the simulation is not a trivial task. This situation occurs because we do not make any assumption about the simulation network size. The simulation could have any number of peers; the network size is constrained by the computational power of the system.

To cope with the situation when the number of available data sets or ontologies to be used in the simulation is less than number of peer in the simulation, we have come up with the two different approaches to assign resources to peers. The choice of the approach depends on whether there is only a *single ontology* to be replicated and distributed among peers, or there are *multiple ontologies* to be used by the simulation. When the simulation is run with a single ontology, the approach is further divided into two different sub-approaches: i. random resource assignment, and ii. computed resource assignment.

Resource distribution comprises a significant part of the Util package. Classes related to the resources distribution are grouped together in a separate package called Resource distribution package, and classes responsible for carrying out the resources distribution include *Resources* and *ResourcesAllocation.*

4.5.1.1 Single ontology with random resources assignment

The random resources assignment correspond to a situation when all peers share the same view of the domain, i.e. use same concepts, attributes and relationship between concepts, to describe a domain, but each with partial knowledge. Peers knowledge of the domain is relevant to their expertise, experience, interest, etc. The random resources assignment comprises of the following three simple steps:

(1) Four partial ontologies o_1, o_2, o_3 and o_4, are derived from the original ontology ϑ[6]. The ontologies are different from each other in terms of the *size* and *content.* The size of o_j is jx where $j = 1, 2, 3, 4$ and x is equal to the 15% of the total number of concepts C in the ontology ϑ. The ontology derivation takes place in two steps:

   i. The concepts C of ontology ϑ are extracted and put in a set S.

   ii. Each o_j is assigned jx concepts from set S *randomly.* The random concept assignment causes partial ontologies o_j to vary in content.

(2) The simulation peers are divided automatically into g groups g_1, g_2, g_3, and g_4 ...g_y. The number of peers in each group is N/g, where N is the total number of peers in the simulation and g is number of groups.

(3) Based on which group g_j a peer p_j belongs to, it is assigned an ontology o_j. That is, all peers in group g_j will be assigned an ontology o_j.

Thus, multiple ontologies are derived from a single ontology where partial ontologies are different from each other in terms of *size* and *content.* The classes responsible for carrying out the described steps are Resources, ResourcesAllocation, and *RadomAllocation* in the Resource distribution sub-package inside the Util package. All

[6]In this study four partial ontologies were derived for the simulation. However, this number is replaceable and it is determined by the undergoing study

the experiments conducted in [68] use this type of resource assignment. The single ontology used for the resource assignment was developed by Standford University and it is located at *http://www.ksl.stanford.edu/DAML/laptops.owl.*

4.5.1.2 Single ontology with computed resource assignment

Different from the random resource assignment approach, section 4.5.1.1, the simulation also supports a computed resource assignment approach. The two approaches are similar except in the way partial ontologies o_i are constructed.

In this new approach, concepts added to partial ontologies o_i are not arbitrarily selected from the set S. The random concept selection is replaced with a systematic selection in which an *overlap function* is defined to ensure concept sharing among partial ontologies. The overlap function defines the number of concepts, v, common among partial ontologies. For example, a peer p_i with ontology o_i could share up to 5% of its concepts with at least another peer p_j. Thus, the four partial ontologies, o_1, o_2, o_3, o_4, defined in step 1 section 4.5.1.1, will be populated as follows:

The first ontology o_1 will be assigned an o_1 size number of concepts from set S. That is, concepts between index=0 to index=z_1, z_1 is the size of o_1, will be copied from set S and assigned to partial data set o_1. The concept indices to be copied to o_2 starts from index $z_1 - v$ to $z_2 - v$, for o_3 they start from index $z_2 - 2v$ to $z_3 - 2v$, and for o_4 they start from index $z_3 - 3v$ to $z_4 - 3v$; if the index gets bigger than the size of the set S, the index turns around to the beginning of the set. For any number of partial ontologies o_i, the start and end index for an ontology o_i would be defined as follows:

$$indices = \begin{cases} i = 1, & \text{start\_index} = 1 \qquad \text{end\_index} = z_1 \\ i > 1, & \text{start\_index} = z_{i-1} - (i-1)v \quad \text{end\_index} = z_i - (i-1)v \end{cases}$$

Using this strategy, *reproducing test results* become easier than the case where concepts are randomly added to the local ontologies. Figure 4.4 graphically demonstrates the resources assignment using the single computed resource assignment where ovals represent unique peers and filled circles inside ovals represent ontology concepts. The classes responsible to carry out this resource distribution approach are Resources, ResourcesAllocation, and *ComputedAllocation* in the Resource distribution sub-package inside the Util package.

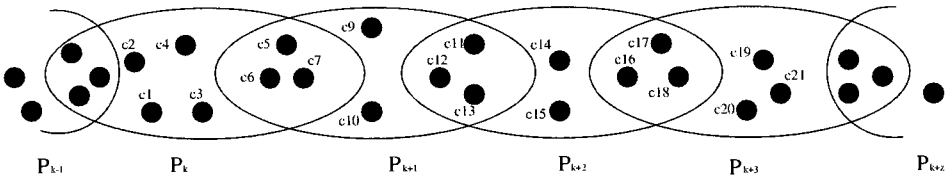

Figure 4.4: Single ontology with computed resource assignment method

4.5.1.3 Multiple ontology assignment

When nm ontologies exist for use by the simulation, and $nm > 1$ we avoid further dividing ontologies to data subsets. Thus, the task of data assignment is reduced to: 1. creating groups and assigning peers to groups, and 2. assigning ontologies to peers.

The first step is carried out by creating g groups where g is equal to the number of ontologies available for use. Peers are assigned to each group based on their id. For example, a peer p_i will belong to a group g_1 if its $id \mod g = 1$.

To carry out the second step, the classes of each ontology o_i are extracted and stored in a set s_i; a peer p_i in a group g_i then is assigned a data set s_i.

Multiple ontologies resource assignment corresponds closely to the real-life situation. There is always more than one ontology for a particular domain. We have used this resource assignment approach in all the experiments carried out in sections 5.5.2, and 5.5.3. Four different ontologies for the electronic device domain have been created. The ontologies are stored at *http://www.scs.carleton.ca/~armyunis/Owl_File*, and used in the experiments. The classes responsible for carrying out this resource distribution approach are Resources, ResourcesAllocation, Ontologies, loadOWL and LocalOntologies in the Resource distribution sub-package inside the Util package.

4.5.2 Peer profile

The description of peer's domain knowledge, the information content offered by a peer, description of peer's schema, or expertise and services a peer provide comprise

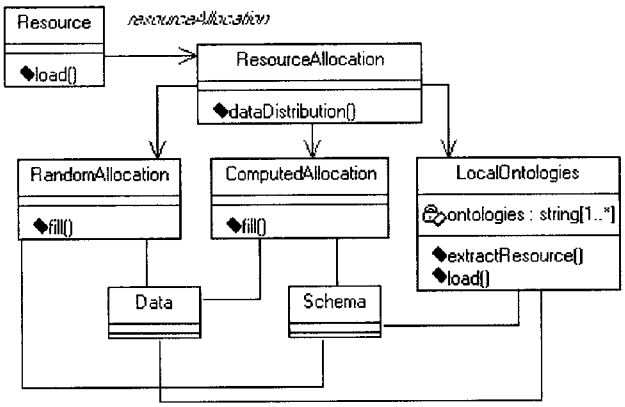

Figure 4.5: Resource assignment methods in the SP2P:sim

the peer's profile in the SP2P systems. The profile is utilized in the semantic neighborhood building, which in turn is used for routing queries to a subset of peers in the network which are more promising than other peers to return query results.

In the SP2P:sim, profiles are created in three different ways. They can be created from: 1. a single file, 2. multiple files or 3. automatically from peers' local resources. The profile creation is rendered using the following three classes implemented by the SP2P:sim: fromAfile, fromMultipleFiles, or fromLocalData. The classes are in the *profile creation* package which is in turn part of the Util package.

When *fromAfile* class is used, the user creates a profile file for each peer in the simulation. On the simulation startup, links between profile files and simulation peers are determined, profiles are loaded and used by the simulation.

Using the *fromMultipleFiles* class, multiple profile descriptions would be saved in a single file and each description is tagged with unique profile name. The file is loaded into the simulation on the startup. Each peer or multiple peers will be assigned a profile. This depends on the number of the descriptions in the file. When the number of profile descriptions is less than the number of peers in the simulation, multiple peers would have the same profile, otherwise each peer would be assigned a profile. We applied this method for tests conducted in [68].

The *fromLocalData* class, on the other hand, enables a peer to extract a set of concepts from its local resource and build a profile on the fly, i.e., at the run time. The size of

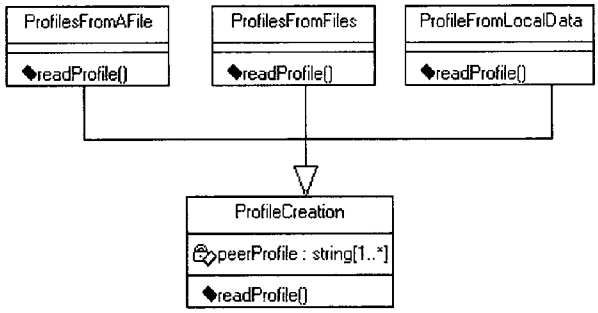

Figure 4.6: Profile creation methods in the SP2P:sim

the profile, i.e., the number of concepts constitute a peer's profile, is $\leq$ the number of concepts in the peer's resource, and it is defined by the simulation user. Pruning resource description to build profile is discussed in [114], and we used it when we were conducting experimental tests in sections 5.5.2 and 5.5.3. Figure 4.6 shows the class diagram for the profile creation in the SP2P:sim.

4.5.3 Semantic neighborhood

Semantic neighborhood construction in the SP2P:sim complies with the semantic neighborhood building with the *discovery* method discussed in the reference model, chapter 3. A fundamental component in building semantic neighborhood with the SP2P:sim simulation is peer's profile. Profiles are descriptions of the information content offered by peers, description of peers' schema (see section 4.5.2). Peers exchange their profiles to find semantically related peers. Based on the similarity between profiles semantically related peers are identified.

On the simulation startup, and after peers have been created, the profile comparison starts. The simulation is designed to run on a single machine. This facilitates the profile comparison through having all peers in a single place, one container. Each peer in the simulation compares all of its profile concepts to all concepts of all other peers in the simulation, and determines the intersection, if it exists, between its own profile and other peers' profiles.

Peers rank other peers with similar profiles that have been discovered during profile

comparisons. The ranking is based on the *similarity relation strength*. The relation strength counts the number of similar concepts that exist between profiles of any two peers in the simulation. For example, the relation strength between two peers p_i and p_j is equal to 3 when each of the p_i's profile, and p_j's profile have three equivalent concepts in them.

Peers make connections or build edges with the most related peers. The total connection a peer could make is constrained by the *connectivity degree d*, number of links a peer is allowed to have with other peers. Calls from a peer p_x to connect with another peer p_{yi}, $i = 1, 2, ...np$, np is the number of peers in the simulation, will be rejected by peer p_{yi}, regardless of the similarity relationship between peers' profiles, if p_{yi} have reached the allowed connection degree (i.e., if $i > d$). The connection degree is a simulation parameter and its value needs to be set on the simulation startup.

Peer connections are directed, but not necessarily bi-directional. This happens because of the connection degree constraints. For example, given that a peer p_j has accepted a call for connection from a peer p_i, does not necessarily mean that a call for connection from the peer pj to the peer p_i will be accepted as well. This happens if the peer p_i has already reached to its highest allowed connection degree. If the above described situation occurred, the peer p_i would be able to send queries to peer p_j, but not vice versa.

Semantic neighborhood creation in the SP2P:sim is limited to peers with related profiles. Semantic neighborhood in the SP2P:sim simulation does not require ontology mappings to exist or to be created in order for peers to be able to connect. This is clearly a limitation, however, the way semantic neighborhood is created in the SP2P:sim fulfills our need to test architecture changes in the SP2P systems. Similar limitations also exist in existing systems for example Chatty Web [3]. Classes used to build semantic neighborhood include NeighborhoodBuilder, PublishDiscover, and Randommeeting. These classes are in the NetworkBuilding package which is in turn part of the SemanticNeighborhood package.

4.5.4 Query formulation

In the SP2P:sim queries are created using query formulator component. In order to represent queries in a high level of abstraction they are created without using any

particular query language. In the SP2P:sim a query is an object which could hold up to *query size* number of concepts. The query size denotes the maximum number of concepts a query is comprised of; it is a simulation parameter and its value needs to be set on the simulation startup.

Concepts comprising queries in the SP2P:sim are chosen *randomly* from the peers' local data sets, and any peer could initiate a query. At each simulation tick, i.e., at the end of each scheduled event, a new query is created and a new initiator is selected. The probability for the new query concepts and query initiators to be different from prior concepts and initiators are high and determined as follow:

The probability (prb_1) of a peer p_i to be a query initiator in each simulation execution is $1/N$, where N is the number of peers of the simulation.

The probability (prb_2) of the query q_i to be same as another query q_j in each simulation execution is $\binom{nC}{q_c}$ where nC is the size of data set or the number of concepts in the local ontology each peer possesses, and q_c is number of concepts that comprise the query content. The probability (Prb) of the same peer p_i to pose the same query q_i in two different simulation executions is then the multiplication of both above terms, i.e.,

$$\frac{1}{N}\binom{nC}{q_c}$$

For example, for a simulation network with 100 peers, each peer with a local ontology of minimum 40 concepts and each query of 4 concepts, the probability of the same peer to pose the same query in two different simulation events is

$$\frac{1}{100} \times \frac{1}{\frac{40!}{4!(36)!}} \tag{4.1}$$

which is equal to $1.094 - 7$. This value is low and indicates that the SP2P:sim ensures opportunity for all peers to participate in query formulation as well as using all concepts owned by peers in the simulation.

Furthermore, in rare cases the random peer selection procedure could repeatedly choose the same peer as query initiator. In order to avoid such a situation, peer participation in the SP2P:sim is controlled through a *peerDomination* simulation parameter. This aids in controlling peer engagement in posing queries. The peerDomination parameter allows each peer to generate only k_q number of queries during

each *simulation run*. The value of k_q is determined as follow:

$$k_q = \frac{N_q}{N} \times peerDomination$$

where N_q is the total number of queries set for execution in a simulation run and N is the total number of peers in the simulation. For example, if query execution number is set to a 1000, the number of peers in the simulation to 100, and peerDomination parameter to 3, then the k_q value would be $((1000/100) * 3)$ which is equal to 30. This means that during the 1000 query execution, one simulation run, no peer would generate more than 30 queries. This leads to controlling the query distribution among peers which in turn helps in better understanding the true behavior of the system aspects which are under investigation.

In the SP2P:simulation, queries could be created without user intervention. This enables running the simulation repeatedly and for as long as needed when various simulation parameters and the SP2P systems aspects are tested. Classes used in the SP2P:sim for query formulation include Query and QueryFormulator from the Peers package.

4.5.5 Semantic mapping

The SP2P:sim provides a procedure for mapping query concepts to peers' local resources and determines the similarities and differences between peers' local resources and queries concepts. The matching procedure takes place on the *receiver* side. Query concepts are matched to the local resources before being sent to the receiver peer.

In SP2P:sim the resources managed by peers are different. When multiple ontologies are used as the simulation resource (see section 4.5.1 for resource assignment in SP2P) the differences arise from the fact that individual groups will have different ontologies. Likewise, when a single ontology is used as a simulation resource, peer group resources remain different. This time the differences are due to the size of the individual ontologies.

Differences in peers' local resources lead to dropping query concepts during query forwarding when concepts from queries do not exist in local dataset of a queried peers. Checking the existence and the similarity between query concepts and peers' local resources takes place at the *Matching* procedure associated with routing component.

Furthermore, the SP2P:sim also includes a separate component called *mapping* where it could be used for mappings which involve more computational ability than matching procedure. The mapping component could also be used for injecting faults into the simulation and selecting whereabouts to inject fault. The location of injecting fault becomes important when we use *voting* technique to evaluate query results. Both Mapping class and the matching procedure inside the Router class are part of the semantic neighborhood package. Currently, Mapping class is merely a place holder which remains to be implemented.

4.5.6 Query routing

In the SP2P:sim each peer keeps information, e.g. peer *id*, about some other peers in the simulation. The known peers (neighboring peers) are identified during semantic neighborhood building. In the SP2P:sim, querier peers pose queries on all neighboring peers. The less relevant peers are ignored during network building, and queries are sent only to the most relevant peers. SP2P:sim utilizes learning to enable efficient query routing, i.e., peers use their past interaction experience to determine future query routing. Peers which return correct answers are rewarded, considered in future querying, and peers which return irrelevant answers are punished, less considered in future interactions. The punish/reward strategy depends on the employed approach by the query answerer component, discussed in detail in chapter 3.

The query forwarding process starts when a peer *maps* concepts in the received query to its local resource. The translated query is then sent over to the neighboring peers. Queries are prevented from being trapped into cycles through labeling. The utilized links between any two peers are marked *used*. Thus, each link is used only once, and while the query answer retrieval is progressing, used links are prevented from forwarding queries.

The query forward process continues until a query reaches a peer which has no neighbor (an out-going link). The peer sends an answer description (translated query) to the initiator peer. In the SP2P:sim the query routing terminates after queries traverse all relevant paths in the semantic neighborhood (no TTL or number of hops were used for query termination). Sending queries only once over the out-going links during a simulation event is sufficient to terminate query forwarding. Query Routing

class from the semantic neighborhood package is responsible for query routing in the SP2P:sim.

4.5.7 Query result assessment

In the SP2P:sim, peers return modified/translated queries as query results. A query result is an abstraction of a resource or a service description that a queried peer in the network can provide, and for which the user must contact the result provider directly in order to download the resource. The query answer could also be seen as an abstraction of an actual data result that could be retrieved from a data repository.

Query results can be evaluated for the ($\equiv$, $\sqsupset$, $\sqsubset$, $*$, $\perp$) relations between concepts in query answers and concepts in querying peer's local ontology. For example, when a peer p sends a query with four concepts $Q(A, B, C, D)$, and receives a response 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 the query and query response concepts is 100%.

In the SP2P:sim, multiple answers could be generated from a single query, each from a query routing path; query answers are send directly to the querier peer. Once a last peer on each of the routing paths receives the query, it will send the result back directly to the initiator peer. The query initiator's *id* is included in the queries to facilitate sending results back directly to the querier peer. Querier peers evaluate answers without user intervention. Automatic answer evaluation expands the SP2P:sim applicabilities and uses for evaluating different aspects of the SP2P systems. These include for example, emergent semantics and reliability.

The SP2P:sim provides three fault-tolerant query result evaluation implementations: *generous query answer evaluation* (GQA), *time redundant query answer evaluation* (TRQA) and *majority voting query answer evaluation*(MVQA). The implementations are different from each other in the way they approach faults. The generous query answer evaluation utilizes the *generosity tit-for-tat* algorithm developed in game theory. The time redundant query answer evaluation is based on the *time redundancy* method widely employed for preventing software and hardware failures. Both implementations are described and tested in details in sections 5.5.2 and 5.5.3.

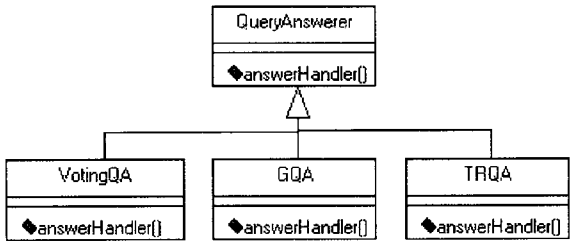

Figure 4.7: Result handling components in the SP2P:sim

Query initiator	Cycles
P_i	$[P_j, P_k, P_l, P_m, P_i]$, $[P_j, P_k, P_l, P_n, P_i]$, $[P_j, P_o, P_l, P_m, P_i]$, $[P_j, P_p, P_r, P_m, P_i]$, $[P_j, P_p, P_l, P_q, P_i]$

Table 4.2: Multiple query answer paths for a single query

The majority voting method is different than the other two approaches in two aspects:

i. It waits for all answers to arrive to the querier peer before making a decision on whether a neighboring peer is returning a correct answer or not.

ii. It takes the view of majority peers, all query path answers, to decide whether an answer is correct or not.

For example if a peer p_i posed a query on a neighboring peer p_j, and as result five answers are returned, each from a path such as the ones described in the Table 4.5.7, peer p_j waits for all answers to arrive or for time out to occur and starts by evaluating query answer generated from each path. If a majority of answers, three out of five in the example, are correct, then the link between peer p_i and p_j is rewarded, peer p_i considers the peer p_j to be a valuable peer regardless to the incorrect answers returned. The four classes QueryAnswerer, GTitQueryAnswerer, TimeRQueryAnswerer, and MVQueryAnswerer are implementation codes of the Query Answerer and its three behavior, GQA, TRQA and MVQA respectively. The classes are part of the peers package. Figure 4.7 represents the SP2P:sim's query result assessment components.

Chapter 5

Fault-tolerant SP2P systems: Case study

5.1 Introduction

In this chapter, we demonstrate how the SP2P:sim framework can be used to build SP2P system simulations, and how the simulations can be used for studying different aspects of existing SP2P systems. Specifically, the simulations will be used for determining the effect of introducing new Query Answer components, the fault-tolerant Query answer component, on the three different SP2P systems, namely, Chatty Web, Piazza and P2PSLN systems. The lack of fault-tolerance capability of the SP2P systems is selected for detailed study because research works have underlined the significance of fault occurrences, specifically semantic mapping faults, and the need to address them [24, 29, 33, 66, 73, 117, 119].

The simulated SP2P systems are different from each other in various aspects. We model different SP2P systems by using different SP2P:sim component implementations. Having been able to derive simulation models from the generic simulation framework and use the simulation models for studying the disconnection problem issue of the SP2P systems is an illustration of the suitability of SP2P:sim framework as a tool for studying architecture changes and other aspects of the SP2P systems.

The rest of chapter is divided into four sections. The sections provide descriptions of: 1) building SP2P simulations using SP2P:sim framework, 2) fault and fault type

SP2P systems	Query Routing		Query Answerer with built-in query retry
	Reducible	Irreducible	
Chatty Web	✓	X	X
Piazza	X	✓	X
P2PSLN	✓	X	✓

Table 5.1: Characters which distinguishes SP2P systems from each other

definitions, 3) SP2P systems' execution process and a demonstration, through a walk-through of a detailed example, of the negative impacts of a lack of fault-tolerance in the SP2P systems, and 4) two solutions for the lack of fault-tolerant problem in the SP2P systems.

5.2 Simulation model building

The first step in simulating a particular SP2P system is to determine the implementation of the seven constructs, identified in the reference model, for the system under consideration. The second step then is building a configuration file. In that file all the classes needed for modeling the system are identified. On the simulation startup, these classes are put together to construct the system.

In the following, we demonstrate how the SP2P:sim component implementations can be used to model three different SP2P systems. The systems are Chatty Web, Piazza, and P2PSLN. Table 5.1 identifies the main characteristics that distinguish the three systems which are chosen for the simulation. The system characteristics have been determined through structure analysis of these systems.

Table 5.1 indicates that these systems employ different Routing algorithms, Query Answerer components, Mapping, etc. Thus, the modeling of each individual system needs to be set differently. Using SP2P:sim component implementations, Tables 5.2, 5.3, and 5.4 represent component configuration file for the systems, Chatty Web, Piazza, and P2PSLN respectively. The classes which spotlight the difference between the SP2P systems are in bold letters.

Based on the simulated system, a fault-tolerant algorithm with a proper policy will be selected for inclusion into the simulation. For example, when simulating Chatty Web system, a GQA algorithm with the partial policy is selected for inclusion in the

SP2P components	**Component implementation**
Profile creation	fromLocalData
Resource Distribution	localOntologies
Semantic Neighborhood	publishDiscover
Query Formulator	queryFromulator
Mapping	match procedure
Query Routing	**Reducible**
Query Answerer	GQA (Partial policy)

Table 5.2: Chatty Web's configuration components

SP2P components	**Component implementation**
Profile creation	fromLocalData
Resource Distribution	localOntologies
Semantic Neighborhood	publishDiscover
Query Formulator	queryFromulator
Mapping	match procedure
Query Routing	**Irreducible**
Query Answerer	GQA (partial policy)

Table 5.3: Piazza's configuration components

SP2P components	**Component implementation**
Profile creation	fromLocalData
Resource Distribution	localOntologies
Semantic Neighborhood	publishDiscover
Query Formulator	queryFromulator
Mapping	match procedure
Query Routing	Reducible
Query Answerer	**TRQA**

Table 5.4: P2PSLN's configuration components

simulation. The partial policy mimics the way Chatty Web handles the incorrect query answers.

The simulation models and model components capture the main characteristics and behaviors of the simulated systems. The simulations would run with any OWL data model built with a Protege ontology building tool. When desired, the simulation component can be replaced with user components. For example, a dictionary for mapping between ontology concepts is manually constructed. In prior version of SP2P:sim, a dictionary was designed to map between concepts from known ontologies, our personal ontologies. If deployed, this would prohibit the simulations from generating large number of unique queries. Thus, for the sake of rendering large number of unique queries and using the SP2P:sim for any domain ontology, concept mapping is replaced with a simple procedure, *Match* procedure in Router class. The procedure merely checks whether or not a query concept exists in the local ontologies. The Match function in the two Router sub-class, RRouter or IRrouter class, enables simulating both IRSP2P system and RSP2P query routing behaviors(see chapter 3 for more information on SP2P type definitions). Class diagrams for the three systems simulated are depicted in the figures 5.1, 5.2, and 5.3 respectively.

5.3 Fault definition, types and causes

Existing SP2P systems are not fault-tolerant because often they fail to distinguish between permanent and transient mapping faults. The lack of fault-tolerance capability may result in erroneous labeling of peers as having incompatible ontologies, which in turn can potentially have serious and far-reaching consequences on system reliability, efficiency and inclusiveness.

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 employs adaptive query routing methods treats temporary mapping faults as permanent mapping faults.

One way to reduce the impact of disconnection failures is through building SP2P systems with fault-tolerant query result evaluation functions. We need to improve the state of neighborhood connectedness of SP2P systems in the presence of semantic

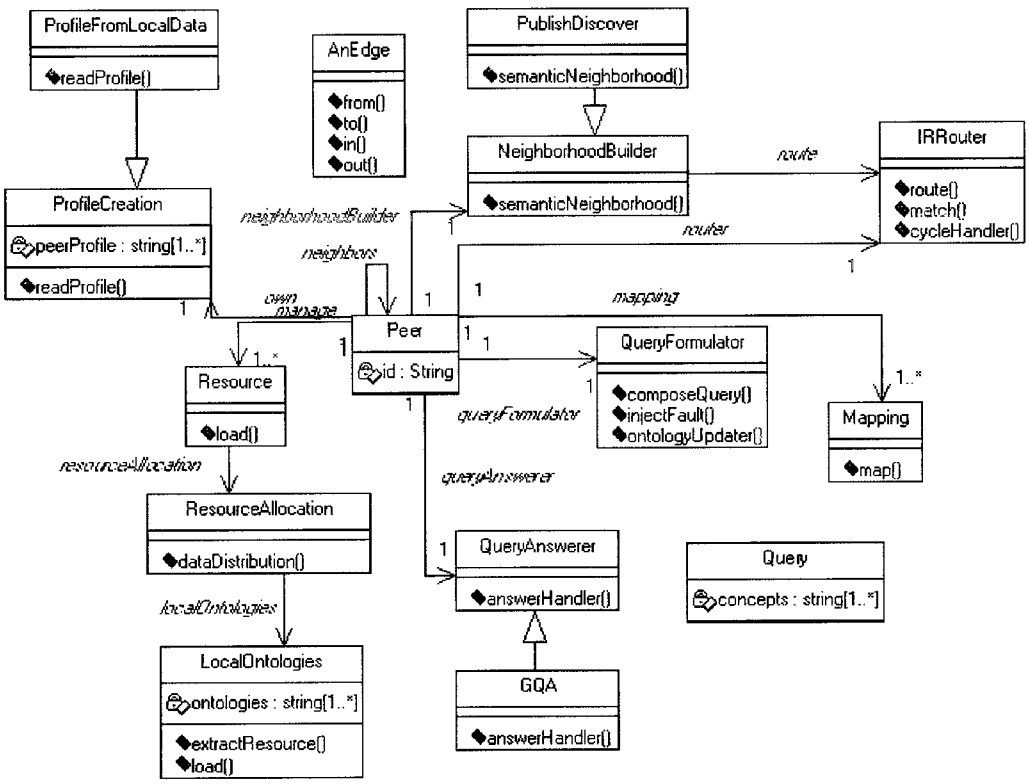

Figure 5.1: Class diagram for Chatty Web system simulation

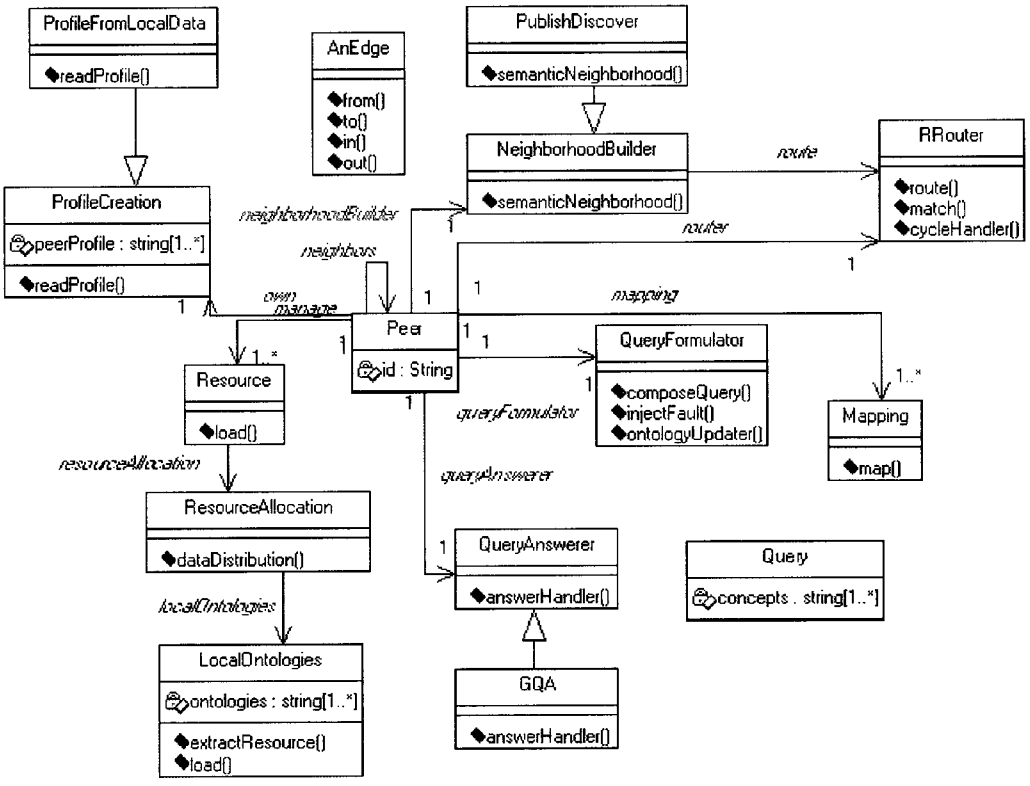

Figure 5.2: Class diagram for Piazza system simulation

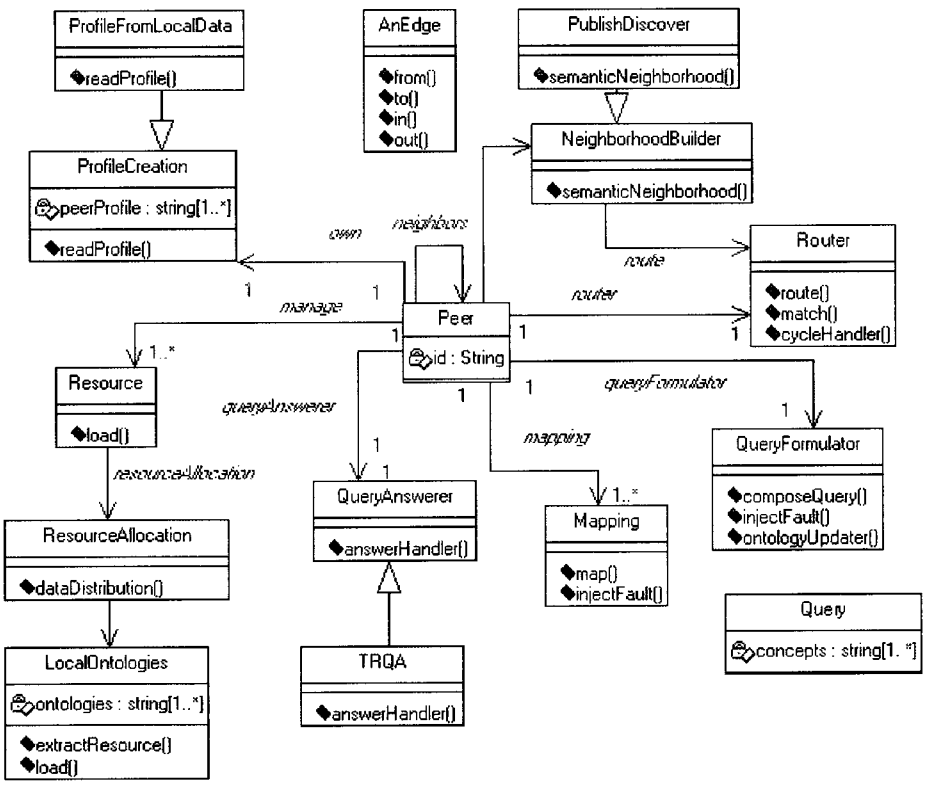

Figure 5.3: Class diagram for P2PSLN system simulation

diversity and ontology changes. That is, the query result evaluation function needs to tolerate temporary mapping faults, otherwise there is a risk that the network connectivity of the system will deteriorate.

In this section, we define what we mean by a semantic mapping *fault*, identify different types of faults, list fault causes, and classify them along the temporal dimension.

5.3.1 Definition of semantic mapping faults and types

Definition 3 *A* fault *is an incorrect semantic mapping, or the failure to map between concepts from different ontologies. We say that a fault occurs when (i) a concept in one ontology is mapped onto a semantically unrelated concept in a different ontology, or (ii) a concept in one ontology cannot be mapped onto an existing semantically related concept in another ontology.*

Formally we can express this definition as follows. Assume we have two ontologies $\vartheta 1 = \{C, Cp, Cr\}$ and $\vartheta 2 = \{\bar{C}, \bar{C}p, \bar{C}r\}$, where C and $\bar{C}$ are sets of concepts, Cp and $\bar{C}p$ are sets of concept properties, and Cr and $\bar{C}r$ are sets of relations between concepts.

Given two semantically equivalent concepts or their instances[1], $c \in C$ and $\bar{c} \in \bar{C}$, $c \equiv \bar{c}$, we say that a fault occurs if either one of the following is true:

- c is mapped onto a semantically unrelated concept $x \in \bar{C}$, $x \not\equiv \bar{c}$;
- c cannot be mapped onto a semantically related concept $\bar{c} \in \bar{C}$, i.e., the mapping process incorrectly leads to nil.

The fault-tolerant literature classifies faults based on the duration of the fault. We distinguish between permanent, transient and intermittent faults.

Definition 4 *A* permanent fault *is a fault that continues to exist, unless some outside action takes place to remove its underlying cause.*

[1]For information on instance data and schemas see
http://jena.sourceforge.net/ontology/common-problems.html#$aBox - tBox$;
http://www.w3.org/TR/owl-guide/, [59].

For example, any attempt to map between two concepts from two unrelated ontologies, i.e., two ontologies from different domains, will result in a permanent fault. This situation will continue indefinitely unless, for example, a change is made in the mapping semantics linking the ontologies.

Definition 5 *A* transient fault *is a type of fault that appears once, and remains in place for a short period of the time.*

A transient fault may corrupt the data of a system, but the system will remain operational. This is a statistical fault and it is hard to predict when exactly it will happen. For example, the change of a company's stock symbol can result in a transient semantic mapping fault, if either the propagation of the change notification to related peers or applications is delayed, or the related peers or applications were unable to incorporate the change immediately.

From the fault tolerance literature we know that treating a transient fault as a permanent one, or vice versa, carries a high risk and cost [14, 15, 55, 63, 92]. We argue that this is also true of semantic mapping faults, especially in the context of SP2P systems for business-to-business and security applications.

Definition 6 *An* intermittent fault *is a fault that occurs randomly. It appears for a short period of time, disappears, and then reappears repeatedly.*

For example, in a situation where ontology modification is not a full substitution of one ontology by another, it is possible for semantically related peers to continue operating. In the described scenario, there can be intermittent faults. Faults will occur because there are situations when related peers are unable to interpret the meaning of concepts in the modified ontologies.

Although transient and intermittent faults manifest very similarly, they are quite different. While the former one is generated from temporally condition, the latter is the result of unstable system. The intermittent fault could be fixed by removing the unstable component from the system, but the transient faults cannot be eliminated.

The diagram in Fig. 5.4 is one way that the three types of errors can be visualized. A semantic mapping can either be correct (no fault) or incorrect (faulty). In the case

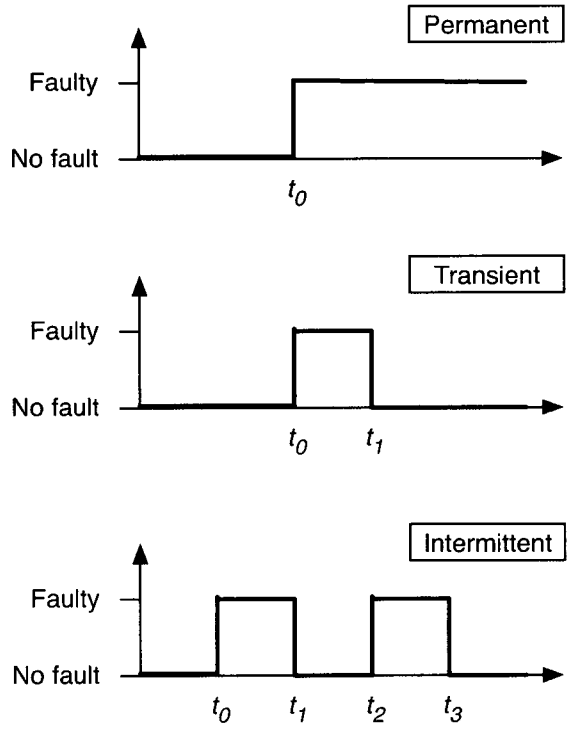

Figure 5.4: Fault types

of a permanent fault, once the status of the mapping changes from no fault to faulty, it remains faulty. For a transient fault, the mapping will be faulty for some time interval. In the case of an intermittent fault, the status of the mapping repeatedly changes between fault free and faulty.

Fault type definitions can be formally represented as follow:

$$f(m) = \begin{cases} Tr, & \text{for } t = t_1 \\ Int, & \text{for } t = t_i\text{: } i \subset [1..n],\ \sum_{i=1}^{n} t_i < T_s \\ Pr, & \text{for } t = t_1, t_2, ..., T_s \end{cases}$$

where Pr , Tr and Int stand for permanent, transient and intermittent fault types respectively, t_i is the duration or the period in which mapping is corrupted, and T_s is an entire system operation duration.

5.3.2 Causes of non-permanent semantic mapping faults

In this section, we discuss situations that can cause non-permanent (transient or intermittent) semantic mapping faults. Our intention is not to be comprehensive, but to illustrate the need for handling (either by tolerating or guarding against) temporary mapping faults. The causes of faults discussed below include ontology evolution, query context and static mapping, temporal nature of data, unavailability of data sources, and misbehavior of peers.

5.3.2.1 Ontology evolution

Ontologies evolve as existing concepts are replaced with new concepts, or concepts are modified. The evolution of software and its consequences on system functionality has received much attention in the software engineering and database design communities [96]. Observations about software evolution can also be applied to the evolution of ontologies.[2] Klein [58] has studied the effect of ontology evolution on Web applications and concludes that it strongly impacts system operability and the interpretation of data.

There are several scenarios in which different types of semantic mapping faults could occur as a result of ontology evolution:

- *Adding new concepts* to an existing ontology, e.g. adding a newly discovered class or type of drug to existing relevant ontologies.
- *Deleting concepts* from an existing ontology. Outdated concepts, or concepts that are no longer used or useful may be deleted from the ontology.
- *Changes in concept meaning.* A changed meaning can result in the removal or addition of a concept relation or property.

An example of a change in ontology by adding new properties would be attaching a concept for hydrogen as a new type of fuel to the concept car. Removing a relation that links the concept floppy drive to the concept PC is another example of a change

[2]Noy [85] argues that the issue of versioning and evolving are one and the same in the context of ontology mapping. What we see as important is that both versioning and evolving introduce modifications to the existing ontology.

in ontology, in this case for a PC maker who no longer supports floppy drives in its product line.

5.3.2.2 Query context and static mapping

Static mapping is a mapping without consideration of the *context* in which a concept is used, i.e., the relations and properties of a concept. It is a *term-by-term* association. For example, if a concept x is mapped statically to another concept y, mapping will always produce the same results no matter the context of x. Static mapping may generate faulty answers to queries when used in different contexts. This can explained by way of the following example.

Consider two concepts (shown in Figs. 5.5 and 5.6) from different ontologies that represent information about *Students* at a *University* and *Members* of a *Research Institute*. Assume the following relations between the two concepts:

1. Some *Members* of a *Research Institute* are *Students* of a *University*, and the *Employee* concept represents this relation.

2. The relationship of the *Research Institute* to the *Institute* and the relationship of *Educational Institute* to *Institute* from the two ontologies are depicted in Figure 5.7.

   One can see that the *University* concept from the first ontology and the *Research Institute* concept from the second ontology become semantically equivalent, i.e.,

   $$University \equiv ResearchInstitute$$

   This is possible, as the *Institute* concept from both ontologies can be declared in the mapping table as equivalent concepts.

Consider the effect of a static mapping from *University* to *Research Institute* on the following two queries:

```
Q1:  List the Names of all  Members of Institutes
```

When this query is posed to both ontologies, it asserts that $University \equiv Institute$. However, consider the second query:

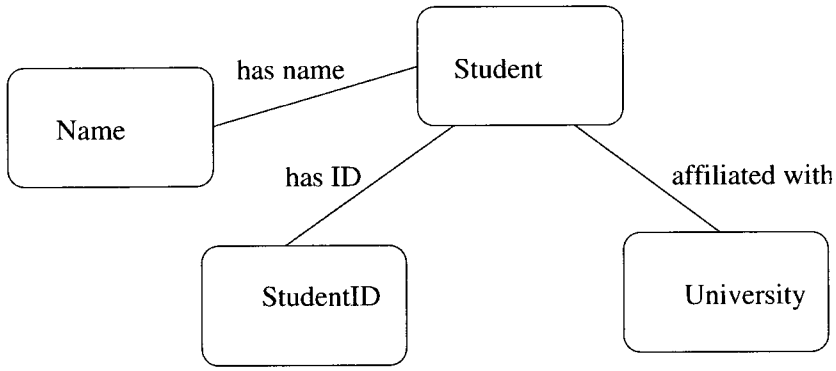

Figure 5.5: Representation of the concept *Student* at a *University*

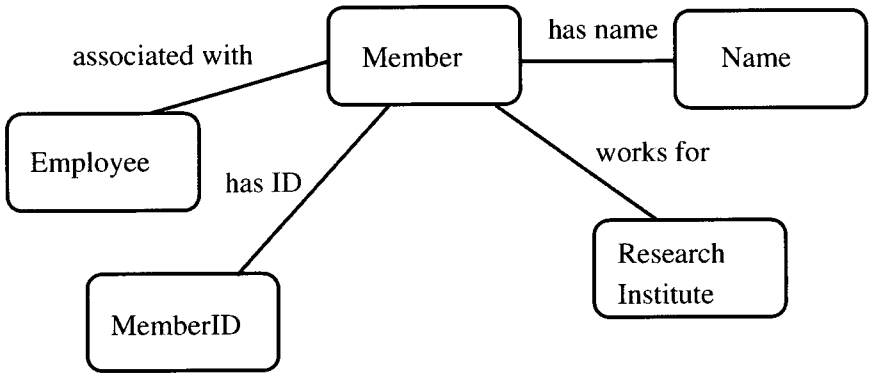

Figure 5.6: Representation of the concept *Member* of a *Research Institute*

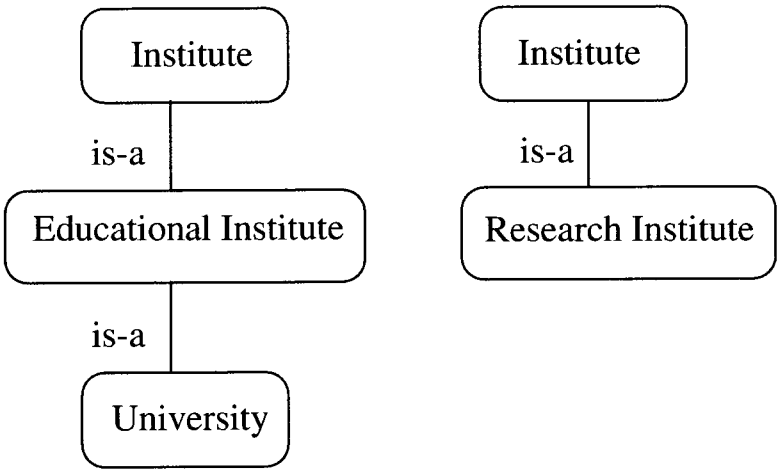

Figure 5.7: Two different representation for *Institute* concept

`Q2: List the Names of all Members of Educational Institutes`

The relation $University \equiv ResearchInstitute$ no longer holds, and this assumption will result in a fault. While the semantic correspondence between the concepts resulted in a correct answer to the first query, it resulted in an fault for the second query. This scenario is a good example of an *intermittent* fault. Every time the static mapping between *University* and *Research Institute* is used, a fault will occur, but there is no fault otherwise. The work of [19] and [88] further elaborates on the effect of context and static mapping on faults.

5.3.2.3 Temporal nature of data

While in previous examples we referred to *concept mapping faults*, concept instances could also lead to transient or intermittent faults. Even though different researchers have different views on whether instances should be part of an ontology or not[75], an important source of faults during query answer evaluation involves changes over time in the concept instances; this is true whether instances are part of ontology or not.

The issue of temporal data is of high importance in situations where data are changing continuously such as stock prices or weather condition. A query answer evaluator that compares temperature values or stock prices represented in two different ontologies may produce different results at different points in time. Not accounting for time dependency can lead to faulty query answer assessments.

Assume that there is a network of peers that provide weather information for different cities, each with a *weather* ontology similar to that shown in Fig. 5.8.[3] Also assume that we want to find the coldest city in the network. One way to achieve this is by running a query similar to the following over all related cities and subsequently comparing the results:

`Q: Find the Location with the lowest temperature`

If query propagation is delayed for some reason, or queries are posed at different times to each peer, the result will not reflect the correct weather temperature. This

[3]Note that this weather ontology is a partial ontology with instances.

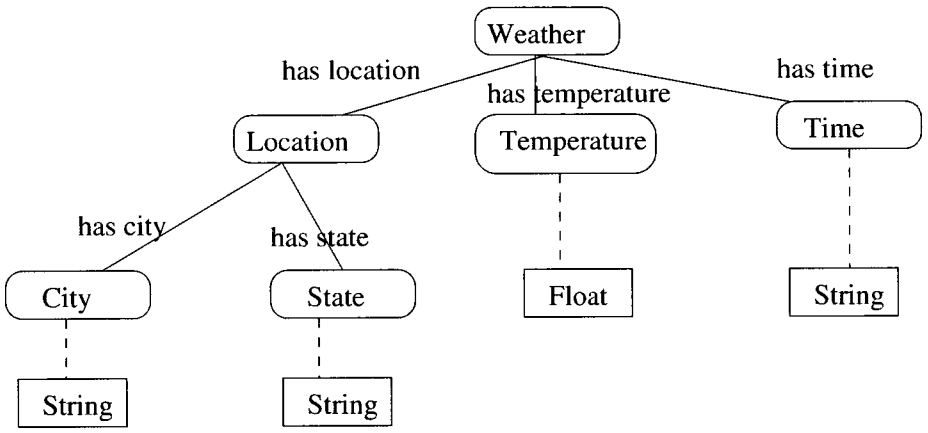

Figure 5.8: Partial weather ontology

fault is not the result of differences in semantic representation (all peers use the same ontology), but rather due to the temporal nature of the temperature concept. This fault could be temporary or permanent, based on whether temporal values are accounted for or not in the ontology. Something similar could be said about a query to find the cheapest stock price. Other examples related to temporal changes of ontology concepts are presented in [118].

The temporal issue is not limited to the concept instances. Similar issues also apply to temporal ontologies. However, while the temporal schemas have been extensively studied in Database research for example, *temporal ontology* is still an open area of research [45, 52].

5.3.2.4 Unavailability of data sources

It has been pointed out by Gal [38] that the design of the conceptual schema for information services possesses special properties. These include i. a rapid change of data sources and metadata; and ii. instability, since there is no control over the information sources. The *availability* of information sources is solely dependent upon information source providers. A possible scenario is the temporary unavailability of information when such information is needed. This possibility is particularly acute during query execution.

5.3.2.5 Misbehavior of peers

Correctness of semantic mapping depends on the honest conduct of peers. A peer could be dishonest or biased in its interaction with other peers during the mapping procedure for reasons such as selfishness or greed. There are various ways through which a peer could influence the mapping process. These ways include i. not forwarding a query to other peers during transitive mapping process ii. not forwarding answers to the other peers during mapping; or iii. altering or delaying queries (results) before forwarding them to other peers. In all of these situations the query answer will be incorrect.

There is some similarity between the information source unavailability described in the subsection 5.3.2.4 and peers misbehavior, but they are not quite the same. While the former is caused by information unavailability, the latter results in the information unavailability. Thus, we decided to present them separately.

Working in an hostile or uncooperative environment gives rise to situations where peers can be permanently hostile or uncooperative. This may lead to permanent faults. However, in the case of unintentional misinterpretation or incorrect implementation of mappings, faults are produced from "noise-like" actions, and it would be correct to assume that they are non-permanent.

In the above scenarios we need to differentiated between permanent and temporary mapping faults. The knowledge about different types of faults along the temporal dimension will help determine when peers should be excluded from further interaction. This helps in building reliable SP2P systems

5.3.3 Classification of temporal mapping faults

In this section, we re-examine the different fault causes that have been listed in previous sections to find out under what circumstances each individual fault cause could result in transient, intermittent or permanent faults; we then classify these fault causes along the following two dimensions:

1. *Type*: As described in Section 5.3.1, we distinguish three types of faults: permanent, transient and intermittent faults.

2. *Cause*: Fault causes are identified in Section 5.3.2, below, when they occur and identify their associated fault types.

Since we assume that local mappings between ontologies already exist, our classification will focus on what faults may occur during *mapping execution*, rather than on faults that may occur because of errors in the *mapping logic*, e.g. substituting a concept by its hypernyms or hyponyms. Mapping faults caused by meaning and representation of concepts are not included in this classification. For these types of fault we refer the reader to [79, 94, 41]. Also, in order to simplify the analysis, we sometimes refer to both intermittent and transient type errors as non-permanent faults.

5.3.3.1 Permanent mapping faults

The following situations could result in permanent mapping faults:

- Mapping *temporal concepts* without a representation of time constraints in the ontology leads to permanent faults. This is because temporal ontology concepts are continuously changing with time. Even if the mapping (per chance) produces some correct mappings without consideration for time constraints, eventually the system will fail completely.
- The *degree* of ontology modification (versioning and evolution), and whether or not the modified concepts will be *used* in the mapping process, will determine the mapping result. A high degree of modification and the frequent use of the modified concepts may prevent semantically related applications or peers from working with the modified ontology.
- If the system is *unavailable*, the mapping process cannot be performed. Unavailability may be the result of a network or peer failure.
- Working in a *hostile or uncooperative* environment can create conditions where peers are permanently hostile or uncooperative. [4]

[4] If multiple peers cooperate and misbehave intentionally, this will create a different type of fault known as *Byzantine* fault, which is not considered in this study.

We would like to point out that *query context* and *static mapping* will less likely lead to permanent faults. If this were not the case, it would indicate that the existing mapping is incomplete. Hence, a better concept mapping would be required.

5.3.3.2 Non-permanent mapping faults

Except from those situations identified in the first case, all other situations will result in non-permanent faults. These situations include:

- A change in *query context* can give rise to intermittent faults. This is because every time mapping is used in contexts other than the contexts for which the relation was defined for, an error may occur.
- A *denial of service* request due to temporary server crashes or the disappearance and reappearance of peers will result in a non-permanent fault.
- In situations where *ontology evolution* is not a complete substitution of the previous ontology, it is possible for related peers or applications to continue operating. In this scenario there can be intermittent faults. Faults will occur because there are situations where semantically related peers are unable to interpret the meanings of concepts in a modified ontology.

  Moreover, the ontology modification procedure also has an impact on the fault type. The modification procedure could result in either i. the *unavailability* of a peer for a short period of time, while the ontology is locked or ii. a *race* condition between the information source and information users, if the ontology user is informed about the change in advance of the modification. That is, the modification problem becomes an instance of the unavailability or temporal problems described above. From this observation we may conclude that every ontology modification can lead to a non-permanent fault.
- Unintentional misinterpretation or incorrect implementation of mappings gives rise to an incorrect mapping. Since the faults are produced from "noise-like" actions, it will be correct to assume that they are non-permanent.

The observations about ontology modification, unavailability and temporal ontology concepts can be generalized as follows:

- The effect of an ontology modification is not as severe as the effect of unavailability. This is because we assume that modifications to ontologies are less frequent than an information source becoming unavailable.
- The probability of transient faults may be higher than that for intermittent faults. Again, this is for the same reason.

It is important to note that, in this section, we have considered causes of faults one cause at a time. For example, we studied the effect of query contexts, temporal aspects, and ontology modifications separately. It will be interesting to explore whether a fault can be the result of multiple causes, and whether we need to distinguish between different fault causes, when a fault occurs. However, the approach that we will take in the next section to detect and remedy faults does not require knowledge of the underlying cause. Table 5.5 summarizes this classification.

	Transient Fault	**Intermittent Fault**	**Permanent Fault**
Temporal Semantic Conflict	One-time message delay	Frequent message delays	Unsupported time constraint
Ontology Evolution	During changes	During changes	Unsupported change management
Query Context and Static Mapping	Unsupported Query Context	Unsupported Query Context	Disqualify
Unavailability of Data Sources	Unavailability $\geq$ Timeout	Frequent unavailability $\geq$ Timeout	Unavailability = ∞
Misbehavior of Peers	One time misbehavior	Repeated misbehavior	Permanent misbehavior

Table 5.5: Classification of temporal semantic mapping faults

5.4 SP2P's lack of fault tolerance illustrations

In this subsection, a description of the execution process of the current SP2P systems is provided. Two scenarios along with a detailed example to demonstrate the lack of a fault-tolerant problem in the existing SP2P systems are discussed as well.

5.4.1 Execution process of SP2P systems

Execution process of SP2P systems comprises the following steps:

1. Peers join a network after finding the first peer with compatible knowledge representation. That is, peers establish mappings to the semantically related peers. Subsequently, peers submit queries to their neighboring peers using concepts in their own personalized local ontologies.

2. Upon receiving a query, each peer starts processing the query locally. If the concepts used to formulate the query are compatible with concepts in its local ontology, it sends back query results to the querying peer. Otherwise, it *forwards* the query to other peers for which they have a direct mapping, after invoking the mapping or translation procedure. Query forwarding will continue, until either (1) the query reaches the query initiator,[5] (2) the query exceeds a specified number of query forwards ("hops"), or (3) the time to live for the query message expires.

3. The querying peer (query initiator) collects all answers returned, and *evaluates* them. If the answers are satisfactory, the query initiator increases its confidence in its mappings and neighbor(s) to provide correct answers[6]. The query initiator could also inform the neighboring peers about the query result. Thus, the entire query forwarding paths will be informed of the result of a un/successful query. Figure 5.9, depicts the described steps.

[5]The query must stop here, otherwise an infinite forwarding loop would be possible.
[6]A successful query result implies a successful series of mappings.

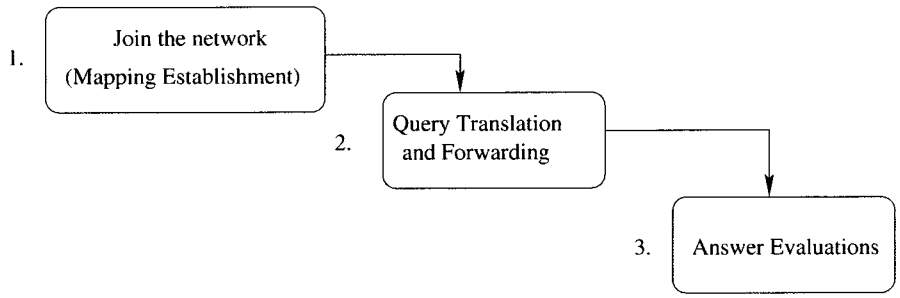

Figure 5.9: The main steps of the SP2P execution process

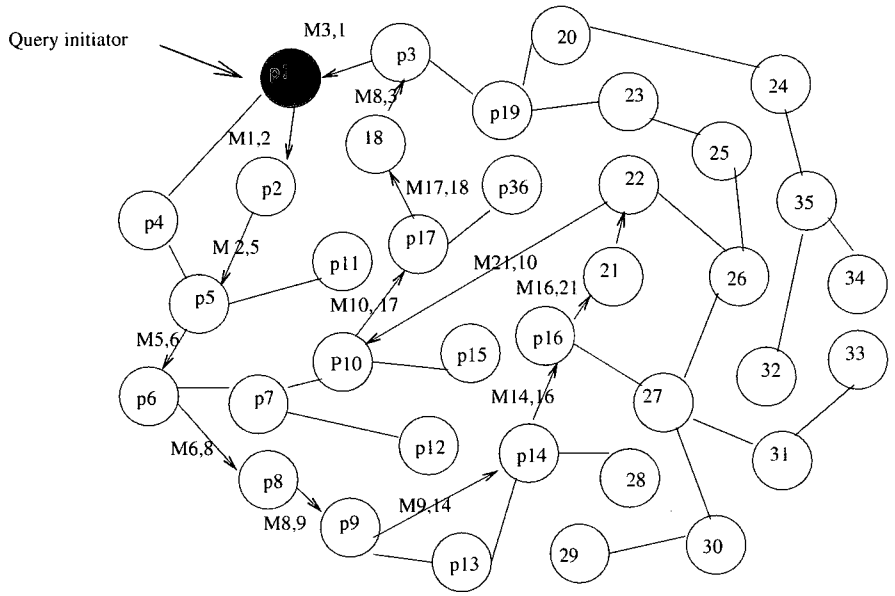

Figure 5.10: Semantically related peers without temporal fault handling

The described steps could be conceived as a process of constructing a directed graph, where anytime a local peer p encounters another peer $\bar{p}$ that provides a correct answer to its query (i.e. a peer with a comparable semantic representation), the existing semantic mapping between these peers will be further reinforced. That is, semantically related peers are discovered and linked to one other during the normal operation of the system — search and query forwarding. Fig. 5.10 depicts such a graph. In the figure, the filled peer is the query initiator, labels on the links represent a mapping from source to target and semantically related peers are connected by a directed link. The graph will be used by peers for future collaboration, e.g. when initiating or forwarding a similar query.

A fundamental prerequisite for the creation of the described semantic graph is the existence of *local mappings* between peers with different ontologies and the *correctness* of those local mappings. Thus, when peers are unable to answer queries or provide correct answers to them, the way in which this failure is handled can become the source of problems. We need to make a subtle distinction between permanent and non-permanent semantic mapping faults or risk the erroneous labeling of peers as having incompatible data representations.

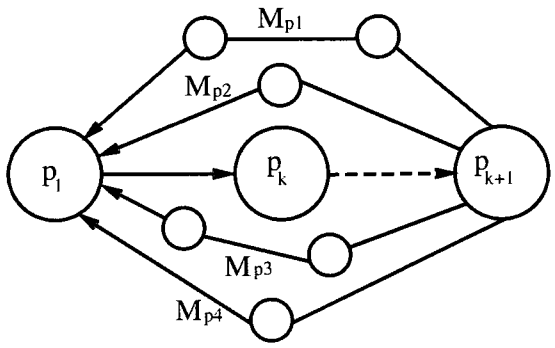

Figure 5.11: A peer on the mapping path has one outgoing link

5.4.2 Critical review of SP2P systems

To illustrate the consequences of erroneously labeling peers as incompatible we consider the effect on the number of outgoing mapping links each peer has to other peers in the network. We will consider two cases:

Case 1 In this case, one of the peers on the mapping path used to answer the query, has only *one* outgoing link. By mapping path we mean the chain of translations used to produce the query result. This case is represented by Fig. 5.11, where peer p_1 is the query initiator, peer p_k the peer with one outgoing link M_k, and all links from peer p_{k+1} form different paths participating in query answers returned to the initiator peer p_1. Small circles on the edges of the graph indicate that different peers participated in forming the results.

It is clear from Fig. 5.11 that unless the system can distinguish between transient and permanent mapping faults, if the mapping M_k between Peer p_k and peer p_{k+1} is not successful, even only for a short period of time, peer p_1 will conclude that the outgoing mapping link M_1 is not entirely reliable, that is, its confidence in the outgoing mapping link M_1 will be reduced.

This is because even a temporary failure of one mapping link, M_k results in the incorrectness of all paths MP_1, MP_2, MP_3 and MP_4, following that mapping link. In other words, mapping faults are dependent. Hence, all the results originated from peer p_k will be considered incorrect for a particular query. Based on (1) the current state of the link M, i.e. its prior value, and (2) the rate of fault occurrence, peer p_k and all other peers on the mapping paths going through peer p_k could be excluded

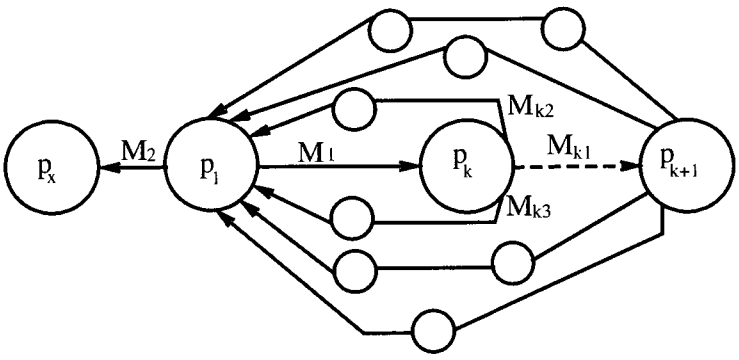

Figure 5.12: Peers on the mapping path have multiple outgoing links

from participation in SP2P execution process.

Case 2 In this case, we are considering a situation when peers have k outgoing mapping links and $k > 1$. Fig. 5.12 represents this case. It shows that p_k has three outgoing mapping links { M_{k1}, M_{k2} ,M_{k3} }. Hence, the decision on the reliability or trustworthiness of the outgoing link M_1 does not depend entirely on the outgoing link M_{K1} as it was the situation in Case 1. Nonetheless, not distinguishing between transient and permanent mapping faults (i.e., treating the mapping link M_{k1} as permanently faulty) will have an impact on the perception of the correctness of the outgoing mapping link M_1.

The wrong perception about any outgoing mapping link, when peers have k outgoing links, could impact the way subsequent queries will be routed. Consider the situation shown in Fig. 5.12. If the original trust in the outgoing links M_1 and M_2 were X and Y respectively and $X - Y = d$, then, if a transient fault on the mapping link M_{k1} downgrades the trust value of M_1 by a value Z, where $Z \geq d$, the peer p_1 will favor M_2 over M_1 the next time it needs to forward a query. This could, in turn, isolate other peers from participating in future collaborations, and lower the precision and recall of query results because of a lower number of peers participating in answering the query.

5.4.3 A demonstrative example of an SP2P execution process and lack of fault-tolerance problems

In the previous two subsections, we described the SP2P execution process and provided a critical review of existing SP2P system. In this section, the described steps will be further explicated through a detailed example.

5.4.3.1 Groundwork

Concepts used for modeling Laptops by Future Shop, Sony, BestBuy stores and Ebay have been used to create different ontologies for the demonstrative example. three of these ontologies are represented graphically in Figures 5.13, 5.14 and 5.15. Two instances of store specific ontology are provided below. Ontology instances are created using Protege editor[7] with OWL DL sublanguage[8]

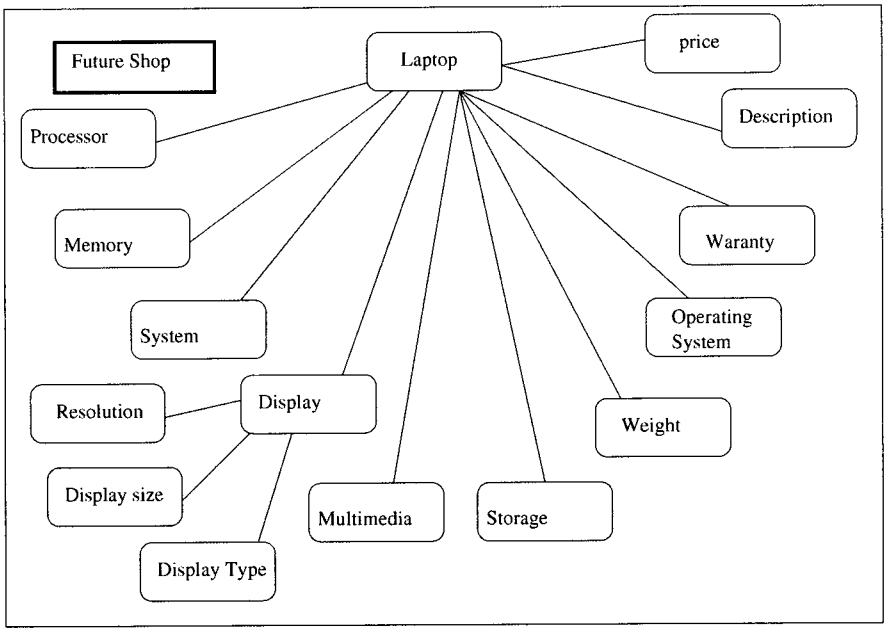

Figure 5.13: FutureShop store laptop ontology

[7]http://protege.stanford.edu/
[8]http://www.w3.org/TR/2003/PR-owl-guide-20031215/

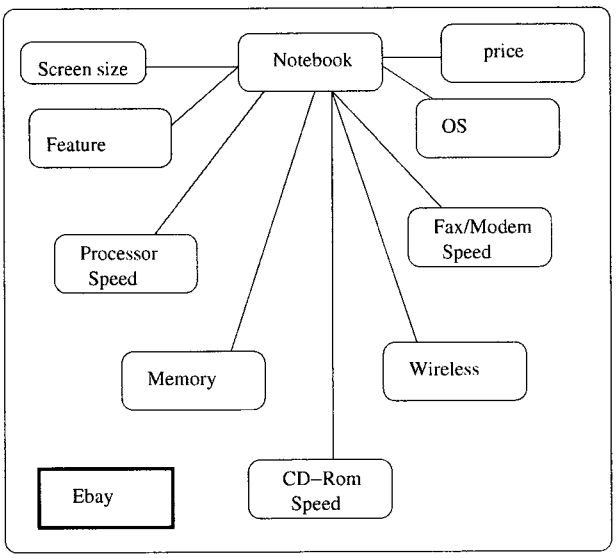

Figure 5.14: Ebay laptop ontology

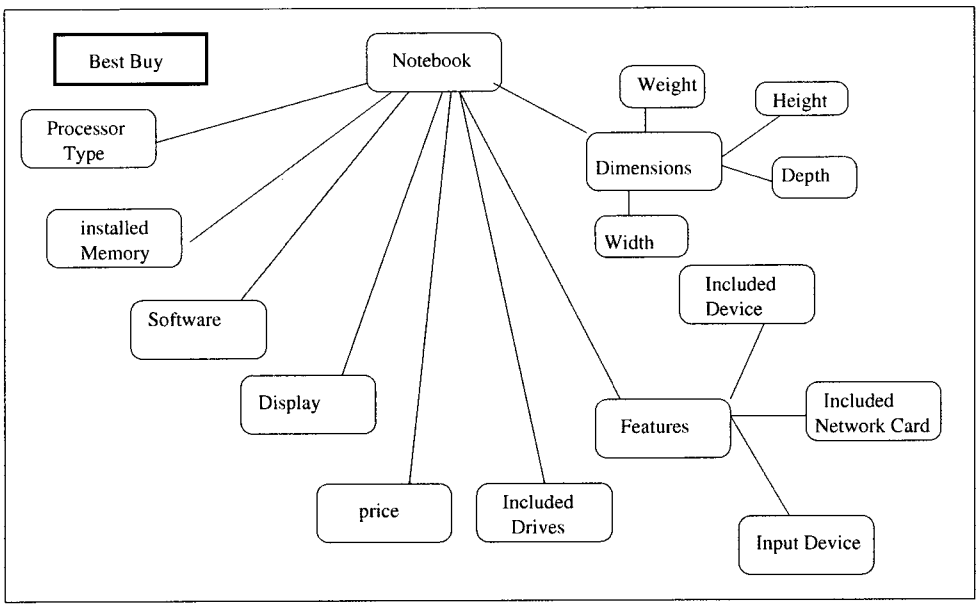

Figure 5.15: BestBuy store laptop ontology

```
< owl : Ontology   rdf : about = "" / >
   < owl : Class   rdf : ID = "laptop" / >
   < laptop   rdf : ID =" 12356789"
      < name   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#string" / >
      SZseries < /name >
      < price   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#float" >
      $1999.99 < /price >
      < features   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#string" >
      1.5   GHZ < /feature >
      < make   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#string" >
      Sony < /make >
   < /laptop >

< owl : Ontology   rdf : about = "" / >
   < owl : Class   rdf : ID = "notebook" / >
   < notebook   rdf : ID =" 12334br456"
      < brand   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#string" / >
      TecraSeries < /brand >
      < price   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#float" >
      $999.99 < /price >
      < qualities   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#string" >
      AMD Turion 64 X2 Duo Core TL-50< /feature >
      < tradeName   rdf : datatype =
      "http : //www.w3.org/2001/XMLSchema#string" >
      Toshiba < /tradeName >
   < /notebook >
```

Fig. 5.11 and 5.12 represent two mapping relation settings among four different store specific ontologies. Tables 5.6, 5.7 and 5.8 present the semantic mappings among

these stores for concepts used in the SPARQL[9] query provided below.

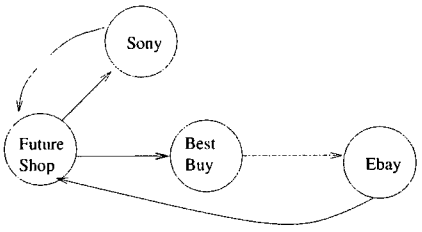

Figure 5.16: Peers with one semantic mapping

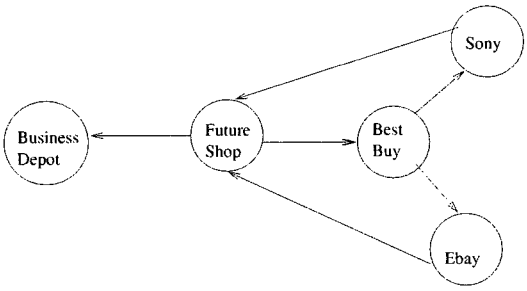

Figure 5.17: Peers with multiple semantic mappings

For the purpose of the mappings, we assume the following logic relations are implemented in the mapping procedure: $\{\equiv, \sqsubset, \sqsupset, *, \perp\}$ where, $c_1 \equiv c_2$ means that the two concepts are synonyms. In other words, c_1 and c_2 are different concepts with similar or identical meanings and are interchangeable. We consider semantic affinity between synonyms concepts to be 1.0. The relation $c_1 \sqsupset c_2$ means c_1 is a hypernym of c_2 (i.e., c_1 is more generic or broad than c_2). The relation $c_1 \sqsubset c_2$, means that the c_1 has a hyponymous relationship to c_2. The semantic affinity for hypernym and hyponym is set to 0.5. The relation $\perp$ means that the two concepts have no semantic relationship with each other. The semantic affinity between two un-related concepts is set to 0.0. Any other relationship between concepts other than those described above will be captured by $*$ relation. The semantic affinity between concepts having $*$ relation is set to 0.25.

5.4.3.2 Execution

5.4.3.2.1 Single mapping link and no fault-tolerance Having a query Q posed on a Future Shop store in a network of stores depicted by Fig. 5.16, concepts included in the query (Laptop, Operating system, Display and Weight) will be translated along a mapping path Fshop→BestBuy →Ebay→ FutureShop.

Assume that BestBuy does not have a Laptop which satisfies the query constraint (i.e. a laptop worth less than $1000), then it will provide no answer to the Future Shop store; it will forward the query to its *only* semantically related store, Ebay.

[9]http://www.w3.org/TR/rdf-sparql-query/

Future Shop	affinity type	BestBuy Store	affinity ratio
Laptop	$\equiv$	Notebook	1.0
Operating System	$\sqsubset$	Software	0.5
Display size	$\sqsubset$	Screen size	0.5
Weight	$\sqsubset$	Dimension	0.5

Table 5.6: FShope → BestBuy

Best Buy	affinity type	Ebay	affinity ratio
Laptop	$\equiv$	Notebook	1.0
Dimension	$\perp$	null	0.0
Screen size	$\sqsupset$	Screen size	0.5
Software	$\sqsupset$	OS	0.5

Table 5.7: BestBuy → Ebay

Future Shop	affinity type	Sony	affinity ratio
Software	$\sqsupset$	System Software	0.5
Display	$\sqsupset$	Display	0.5
Dimension	$\sqsupset$	Description	0.5

Table 5.8: FShope → Sony

```
Q =
PREFIX com < http : //www.../ ~#computer/ >
SELECT ?Operating System ?Display ?Weight
FROM < http : //www.../ ~/computer.owl/ >
WHERE
{
   ?laptop com:Operating System ?Operating System;
           com:Display ?Display;
           com:Weight ?Weight.
   FILTER (com : Price < 1000).
}
```

Future shop will receive the query answer from Ebay because the semantic relation for query concepts along the translation path are high enough for the query to be continuously forwarded and to reach Fshop store.

By comparing the list of query concepts to the list of concepts of the product description result, we could conclude that a semantic relationship along the translation path for the Laptop concept (Laptop → Notebook → Notebook) has been preserved and its equivalent to 1.0. Other product attributes included in the end result product description are: *System Software* with semantic relation equal to 0.5 (Operating System → Software → System Software), *Screen size* with a semantic relation equal to 0.5 as well (Display→Display→Screen size) and one of the attributes, the *weight* attribute, has been dropped by the translation process. Hence, the value of the weight attributed will be set to 0.0. The semantic similarity vector for query result will contain the following values: | 1.0 | 0.5 | 0.5 | 0.0 | for the concepts: Laptop, Operating System, Display and Weight respectively.

For the purpose of query answer validation, we consider an answer to be correct if: i. it satisfies the query constraints, and ii. the sum of semantic relation values for concepts returned in the product description answer divided by number of concepts

be $\geq k$, where k is a system defined value. For instance, if k value were set to ≥ 0.5, then the query result returned by Ebay will considered correct since (1+.5+.5+0)/4 = 0.5 which is equal to k. The result of the correct answer fed-back to the system and beliefs in the correctness of the mapping link Fshop $\rightarrow$ BestBuy will be increased.

The above-described situation assumes a perfect world. In other words, no changes or modification either to mappings or to local ontologies during query process, perfect peers behavior, continues service availability and probably no use of temporal data is assumed. If all these assumptions hold, then belief in the correctness of the out-going mapping link FShop$\rightarrow$BestBuy will increase and it will be considered for subsection queries.

Unfortunately, the world is not perfect. As soon as one of these conditions does not hold, the above described scenario will not work properly. For example, if it happens that Ebay is blocking the incoming queries temporarily- while it updates its local ontology, then the answer to the query Q will not return any result. Depending on the current value of the mapping link Fshop$\rightarrow$BestBuy, this mapping link could be labeled as a faulty link, thus, resulting in a *permanent disconnect between Fshop and sub-network starts from BestBuy.*

5.4.3.2.2 Multiple mapping links and no fault-tolerance In this case we want to study the effect of transient mapping faults a in situation when peers have more than one mapping link to others. Building multiple mapping links is costly but depending entirely on one mapping link is not trouble free either. We have illustrated the latter case in subsection 5.4.3.2.1.

Figure 5.17 represents the discussed example in subsection 5.4.3.2.1 with BestBuy having more than one mapping link (two mapping links).

Following the query forward steps described in subsection 5.4.3.2.1, once BestBuy receives query Q from Future shop, it will forward it to its semantically related peers (i.e., Ebay and Sony store). Assuming Ebay is blocking incoming queries temporarily, Future shop will receive a query vector result from Sony store for concepts included in the query | 1.0 | 0.5 | 0.5 | 0.5 | [10] and null value from Ebay.

[10](Laptop $\rightarrow$ Notebook $\rightarrow$ Notebook), (Operating System $\rightarrow$ Software $\rightarrow$ System Software), (Display$\rightarrow$Display$\rightarrow$Screen size) and (Weight$\rightarrow$Dimension$\rightarrow$ Description)

The null value returned from Ebay will have a negative impact on evaluating the correctness of the out-going mapping link Fshop→BestBuy. The intensity of this negative impact depends on:

- the other result returned from the path (Fshop→BestBuy→Sony→Fshop) and,
- the way returned results are used in evaluating the correctness of the out-going mapping link.

In general we will have the following three cases(although other sub-cases are also possible):

(1) if the result returned by the mapping path (for instance the Fshop→BestBuy→ Sony→FShop mapping path) is not correct, then the decision about trustworthiness of the out-going mapping link Fshop→Sony will depends entirely on the result retuned from Ebay. Hence, the temporary failure will have drastic effect, i.e. possibility of total abandoning of the out-going mapping link. This is not the case in our example.

(2) if the correctness value of an out-going mapping link is determined by taking a median or average of the results (the average value is used in this example), then the transient fault will have a severe impact and, it could result in abandonment of out-going mapping link. *This case is applied to our example.*

(3) If the original confidence in the outgoing link FutureShop → BestBuy and FutureShop → Business Depot were X and Y values respectively, and $X - Y = d$, then if a transient fault on the mapping link BestBuy → Ebay downgrades the confidence value of FutureShop → BestBuy by a value Z where Z ≥ d then, Future Shop could favor FutureShop→Business Depot over the FutureShop→BestBuy for next query forwarding.

The described example demonstrates that, regardless of the number of out-going mapping links, there is still a need for handling the temporary faults in order to achieve emergent semantics.

In a situation where an out-going mapping link continues to be used as long as one correct answer is returned, i.e. the required correct value for using an out-going mapping link is > 0, applied in [3], then the temporary mapping fault will have no effect on the validation of the out-going mapping links unless all links which are

returning correct results infected by temporary faults. We consider forwarding query on mapping links with substantially low confidence values to be another deficiency in the existing systems. This is because it will lead SP2P systems close to broadcasting queries, something SP2P systems are trying to avoid in the first place.

We conclude this section with the anticipation that once the effect of the temporal semantic mapping faults are eliminated, a denser graph than the once presented in the Fig.5.10 emerges. This is because all the peers that were excluded from participation in the first place, because of the temporary semantic mapping faults, will be included this time. Fig. 5.18 is an instance of the expected new graph. Here, filled peers are the new peers which have been added to the first graph.

In the following section, two solutions to fault-tolerance will be described and the anticipated results will be examined.

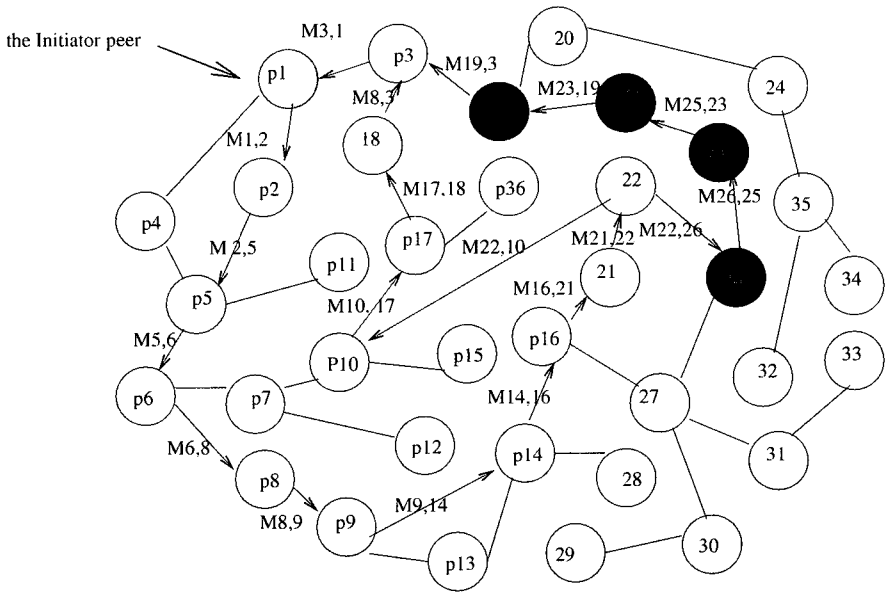

Figure 5.18: SP2P network when mapping fault handled

5.5 Lack of fault-tolerance problem solutions

Overcoming semantic mapping faults is a vital issue for the success of SP2P systems. There have been various research efforts which address the classification and the resolution of the semantic mapping fault problem, but all of the previous research related to semantic mapping faults demonstrate one significant shortcoming. This flaw is the inability to discriminate between non-permanent and permanent semantic mapping faults, i.e., how long do semantic incompatibilities stay effective and are the semantic incompatibilities permanent or temporary? In this section, two solutions, Generous Query answerer Algorithm (GQA) and Time Redundancy Query answer Algorithm (TRQA), are described for the lack of fault-tolerant problem in the SP2P systems. The effect of the solution on three SP2P systems: Chatty Web, Piazza and P2PSLN systems are simulated along with studying the effect of the solutions on the systems that are only possible to be built using our simulation framework. The similarities and differences between the two solutions will be explored. The algorithms' cost in term of the number of messages used along with detailed experimental results validating the algorithms' effectiveness are provided as well.

5.5.1 GQA and TRQA fault recovery approaches

The time redundancy query answer algorithm (TRQA) described in section 5.5.3 and the generous query answer algorithm (GQA) described in this section are two different ways to endow SP2P's Query Answer component with the ability to tolerate faults. There are some differences and similarities between these two approaches, below these are explored.

Peers employing the GQA algorithm postpone their decision about their cutting/preserving relationship with their neighbors until further interactions take place. That is, peers test their neighboring peers with *new queries*. Using the TRQA algorithm, the *same query* posed on the neighboring peers will be resubmitted after some time to get new answers, and, hence the decision to cut/preserve the relationship with the neighboring peers.

The TRQA algorithm's objective is to prevent treating transient faults as permanent

ones. Hence, the TRQA algorithm requires an additional procedure to adjust relationships with neighboring peers. The GQA algorithm, on the other hand, deals with faults without requiring any additional procedures.

The TRQA is a suitable method for reducing faults caused by peers' unintentional misconduct (e.g. message delay conveying ontology changes information, and when used in the Web environment for web unavailability). In addition to these situations, the GQA algorithm is also effective in situations where faults are caused by static local mapping between schemas and when query answers depend on the query context(see chapter 5, section 5.3.2.2 for faults caused by query context static mapping).

Despite the described differences between the GQA and TRQA algorithms, there is one similarity between the two algorithms. Both algorithms are based on the peers' own assessment of other peers. In both algorithms, peers determine future relationship with other peers based on their own experience. Other peers' experience in the system is not accounted for.

5.5.2 Generous query answerer algorithm

Query result evaluation strategy is an important aspect of adaptive query routing in SP2P systems. In order for SP2P networks stay connected, they need to have a fault-tolerant query answer components that employ a correct result evaluation function. An incorrect evaluation function prevents semantically related peers from teaming-up together. In this section we describe the Generous Query Answerer algorithm (GQA) for solving the SP2P disconnection failure problem. The GQA algorithm is simple in concept, easy to implement and highly effective; it is inspired by the generosity tit-for-tat algorithm developed for game theory.

5.5.2.1 GQA Algorithm's steps and procedures

Several design decisions have been made during the algorithm development. These include:

i. *Use of local knowledge.* Peers have only knowledge of their immediate neighbors. A peer's knowledge is related to its belief in the reliability or ability of neighboring

peers for 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, i.e., lose confidence in each other, when the reliability reaches 0, hence they drop their mapping links.

ii. *Normalization is not applied.* Sending a query on an outgoing link can 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.

iii. *Use of average values.* As initiator peer receive query answers, it updates the confidence value in its out-going mapping links. Treating query answers equally in updating out-going mapping links is equivalent to averaging query answer values.

These decisions are made to ensure the algorithm is easily understood. Future revision of these decisions is possible. The algorithm, made up of three essential functions: (1) *initialization*, (2) *result evaluation*, and (3) an *update* function, 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 ($\equiv$, $\sqsupset$, $\sqsubset$, $*$, $\perp$) relations between concepts in query answer and concepts in querying peer's local ontology. That is, whether the semantic relationship between query answer concepts and a peer's local concepts are *exactly the same*, *subsumed*, *related*, or *totally not related.* The result verification could also be interpreted as checks on whether the query response satisfies all, some or none of query constraints. 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 $as(\bar{A}, \bar{B}, \bar{C}, \bar{D})$, 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 the 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 from each other in two aspects:

i. whether or not peers consider partially correct answers to be faulty or correct, and

ii. how much peers are tolerant toward faults.

These policies could be described as follows:

Under the *Complete* policy only exact answers are accepted by the querying peers. Peers returning exact answers are rewarded by querier peers, i.e., querier peers increase the confidence value in the related out-going links by Re, and $Re = 1$. The peers that return partially correct answers are treated as if they were returning incorrect answers. Peers implementing this policy are not tolerant to faults, thus, whenever neighboring peers return incorrect or partially correct answers, their confidence values will be reduced by Pn, where $Pn=1$.

The *Firm* policy is similar to the Complete policy in accepting only exact answers. However, peers employing the Firm policy are more tolerant toward partially correct answers. Peers will reduce the confidence in their outgoing links by $\hat{Pn}$ values whenever they receive partially correct answers, $\hat{Pn} < 1$. The values used to deduct the confidence in the connected neighbor depend on the type of query answers, i.e., $\{\sqsubset, \sqsupset, *\}$. Table 5.10 provide some instances of these values.

The *Partial* policy neither rewards nor punishes neighboring peers when they return partially correct answers: it is neutral toward partially correct answers. Thus, no changes will be applied to the confidence values of the related out-going links. However, the Partial policy acts like the Complete and the Firm policies toward the incorrect answers, i.e., it reduces the confidence value of outgoing neighbors by Pn, where $Pn = 1$.

The *Benevolent* policy is a fault-tolerant policy. It accepts partially correct answers and rewards related out-going links by $\hat{Re}$ value when $\hat{Re} < Re$. The Policy follows the generosity tit-for-tat approach in dealing with incorrect answers. The generosity

Update Policies	correct	incorrect	partially correct
Complete	reward (Re)	punish (Pn)	punish (Pn)
Firm	reward (Re)	punish (Pn)	less severe punish ($\hat{Pn}$)
Partial	reward (Re)	punish (Pn)	neutral (Nu)
Benevolent	reward (Re)	less severe punish ($\hat{P}$)	partially reward ($\hat{Re}$)
Failure Guarded	reward (Re)	less severe punish ($\hat{P}$)	partially reward ($\hat{Re}$)

Table 5.9: Summary of GQA policies reaction toward answers

feature entails punishing incorrect answers by $\acute{P}n$ where $\acute{P} < Pn$ and $\hat{Pn}$; querying peers tolerate up to *kf-1* faults in sequence. For a neighboring peer to be disconnected from a querying peer, it has to return *kf* incorrect answers in sequence. The number of the faults that the policy could handle, *kf*, is the system variable. The value of the variable is set by the system users according to their needs and application purposes. Once a permissible number of faults are exceeded, the policy follows the tit-for-tat policy, and ultimately disconnects from neighboring peers that return incorrect query answers.

The *Failure guarded* policy is similar to the Benevolent policy in dealing with partially correct results and in tolerating faults. Furthermore, the Failure Guarded approach provides peers with an additional capability preventing them from reaching a state of total isolation, thus preventing the network disconnection. This is achieved by changing the result-evaluation function: peers check their outgoing connection degree prior to cutting ties with their neighbors. 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 continuous connection policy is not achieved without cost. Peers employing this strategy have to accept a high rate of faults for the sake of the connectivity.

These policies are summarized in Tables 5.9 and 5.10. A sample of numerical values that could be used for updating the confidence values in outgoing links for all five policies are presented in Table 5.10. These values are system parameters: their values are set by the system administrator in such a way that application needs can be met in the best possible way. Algorithm 1 below is the pseudocode for the GQA's steps and procedures.

Algorithm 1 Answer handling algorithm pseudocode

```
1. procedure Initialize
2. /* the policy is set to Benevolence */
3. ChangeInStrength [ ] ← {0.2, 0.1, 0.1, 0.05, −0.2}
4.  While(aPeer.haveMoreLinks)
5.          aPeer.setOutGoingMappingLink ←1
6.  End While
7. End procedure

8. procedure Result_Evaluation
9. int count ← 0
10. int nocq ← getQueryOriginSize
11. Set <String>temp ← getAnswerContent
12. IF( temp.size() > 0) {
13.    For (i←0 To i < tempSize)
14.         IF ( (initiatorResource).understood(temp.elementAt(i)) )
15.             count ← count + 1
16.         EndIF
17.    EndFor
18. EndIF

19. // ChangeInStrength value represents a fault-tolerance policy, Table 1 hold these values
20. IF(count = nocq) {Update (anEdge, ChangeInStrength[0]) }
21. Else IF (count = (nocq -1)) { Update (anEdge, ChangeInStrength[1]) }
22. Else IF(count = (nocq -2) ) { Update (anEdge, ChangeInStrength[2]) }
23. Else IF (count = (nocq -3)) { Update (anEdge, ChangeInStrength[3]) }
24. Else { Update (anEdge, ChangeInStrength[4])}
25. EndIF
26. setEdgeColorStrength (anEdge)
27. removeUnusedLinks (initiator, anEdge)
29. End procedure

30. procedure Update(AnEdge anEdge, double nstrength)
31. double newStrength ← 0
32.         newStrength ← anEdge.getStrength() + nstrength
33.         anEdge.setStrength(newStrength)
34. End procedure
```

Update Policies	Update Values
Complete	[1, -1, -1, -1, -1]
Firm	[1, -0.5, -0.25, -0.05, -1]
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]

Table 5.10: An instance of numerical values that could be used by policies

5.5.2.2 Fault simulation

A list of situations that could raise faults in SP2P systems are surveyed in Section 5.3.3. In our simulation, however, the focus is placed on the *ontology changes* or ontology updates to generate faults. Peers' local ontologies are updated by *adding* new concepts to the existing ones, and faults are generated by forming queries using the newly added concepts to the local ontologies.

When a query is created with a set of the peer's local concepts, and that query is presented to other peers that have only a subset of the query concept constituents (not newly added concepts) the query answers will contain only the minimum common denominator of concepts that denote either: i. meta-data about actual result documents, or ii. returned modified queries.

The difference between the number of concepts in the query and the number of concepts in the answer, and/or the inability of querier peers to understand the query result concepts, are conceived as an indication of the fault occurrence. The reproduction of fault is possible. This is due to the fact that the user of the simulation has control over how to add concepts to the peers' local resources and how to select concepts from local resources to compose queries.

Query answer concepts will be understood by querier peers if they use the same concepts to describe their resource or have mapping capability which defines the relationship between returned concepts and their local concepts.

5.5.2.3 Test settings and experiment results

SP2P system reliability is related to the *network connectivity*, i.e., network *component count*. A highly connected network is reported as a reliable network, whereas

a network with a high number of components,(i.e., disconnected and fragmented network) is perceived as an unreliable network. Preventing SP2P network from fragmentation while peers are executing queries manifests the effectiveness of a particular fault-tolerance policy.

The simulated SP2P systems include known systems such as Chatty Web (sections 5.5.2.3.3), Piazza (sections 5.5.2.4.3), and P2PSLN (section 5.3), and systems that could be built using the SP2P:sim framework. Test results are for two types of system settings: an SP2P system with built-in fault-tolerant capability (Benevolent and Failure guarded test results), and an SP2P system without fault-tolerant capability (Complete, Firm, and Partial test results).

For each test, a network of 150 nodes are used, 1000 queries are executed in each simulation run, and the simulation run is repeated thirty times; the median value of runs are reported as test results. Table 5.11 shows the parameters which are used to configure the SP2P system simulations and experimental tests.

5.5.2.3.1 Complete semantic relation maintenance When only exact answers are considered by the querying peers, peers that return partially correct (relevant) answers will be treated as if they were returning incorrect answers. Peers implementing this policy are not tolerant toward faults, thus, whenever neighboring peers return partially correct answers, their confidence values are reduced by 1.

Figures 5.19, 5.20, and 5.21 show that network deterioration trends are almost identical for cases where the semantic relationship between query result concepts and querier peers' local concepts are $\sqsubset, *$, and $\perp$. It takes a similar number of queries, ≤ 300, in all three cases for the network to reach a total disconnection state, i.e., individual peers drop-off all their connections. This is because all three policies follow exact same strategy toward incorrect answers, i.e., reduce their confidence values by 1. Figure 5.22 compares the network behavior for the three relation types.

Figures 5.23, on the other hand, shows that when there are no faults in the system, peers with Complete semantic agreement remain connected. Network deterioration stops once peers drop their connections to less compatible neighbors. The number of network components is one-tenth (fifteen components) of the number of components

Simulation Parameter	Parameter Description
runs	sets the number of times that the same simulation execution is repeated. This parameter is used to improve result accuracy. The default value is 30.
numQueryGenerated	Sets the number of queries executed in each simulation run. The default value is 1000.
maxNumberOfConceptsInQuery	sets the size of the query, maximum number of concepts comprised each query.
numNodes	Sets the network size. The default value is 150.
maxDegree	Sets the number of neighbors each peer could have.
sRelation	Sets the relationship between query and query answer: not related, related, subset(superset), and synonymous value. This is achieved through controlling query constituent, i.e., whether the newly added concepts to the peer's local ontology used in the queries or not.
changesInString	Set the GQA's applied policy: Benevolence, partial, firm, or Complete.
failureGuardedPolicy	This boolean parameter checks whether the Failure Guarded policy is applied or other policies defined by changesInString parameter.
sp2p	Sets SP2P system type: Reducible or Irreducible type system.
peerDomination	Ensures peers participation. Each peer is allowed only to generate K number of queries during each simulation run. The value of K is: $K = ((numQueryGenerated/numNodes) * peerDomination)$ number of queries during each simulation run.
fault_tolerance_algorithm	Sets the applied algorithm: GQA, MVoting (majority voting) or TimeR (time redundancy).

Table 5.11: SP2P configuration parameters for GQA algorithm

for the other types of concepts relations, i.e., when fault occurs. The high network connectivity result is due to the absence of the fault.

The test results show that in the presence of the frequent ontology changes, SP2P systems are not dependable. Thus, for the SP2P systems to be reliable and trustworthy, they need to integrate a highly effective fault-tolerant query answer component into their system structure.

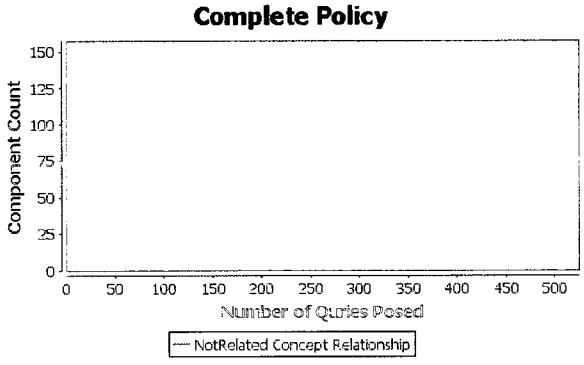

Figure 5.19: Network deterioration under a *Complete* policy when the answer is completely faulty ($\perp$)

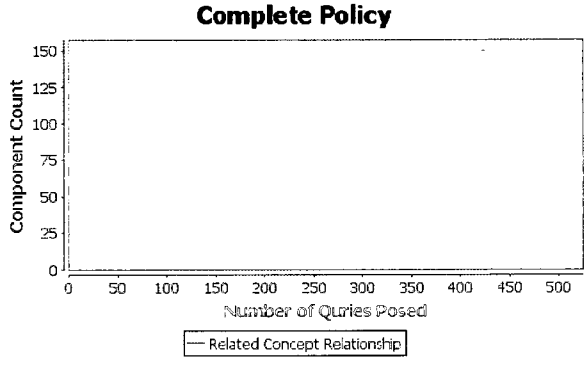

Figure 5.20: Network deterioration under a *Complete* policy when the answer is related to the request ($*$)

5.5.2.3.2 Firm semantic relation maintenance Figures 5.25 and 5.26 show that the trend of the network deterioration is similar to the Complete semantic maintenance case (subsection 5.5.2.3.1). One difference between this situation and the previous is that the number of queries executed for the Firm policy case is larger

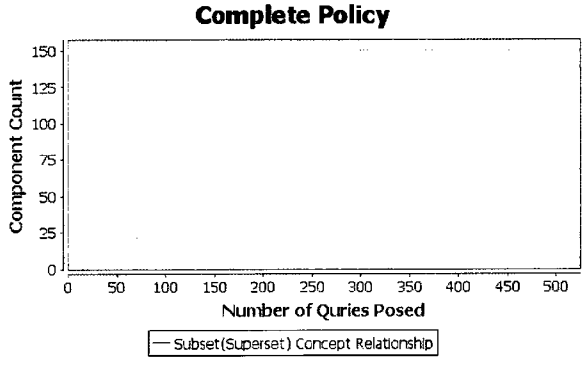

Figure 5.21: Network deterioration under a *Complete* policy when answer is partially asserted ($\sqsubset$)

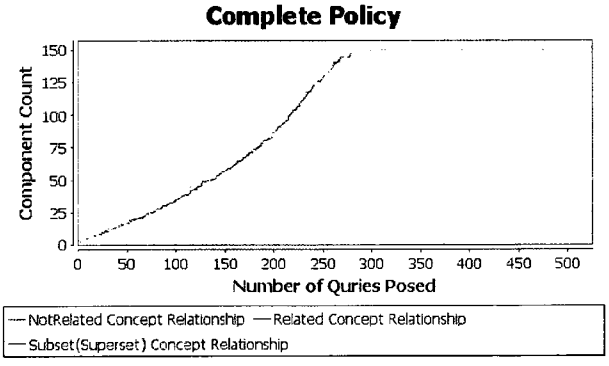

Figure 5.22: Comparing the network behavior for the three relation types ($\perp$,$*$,$\sqsubset$)

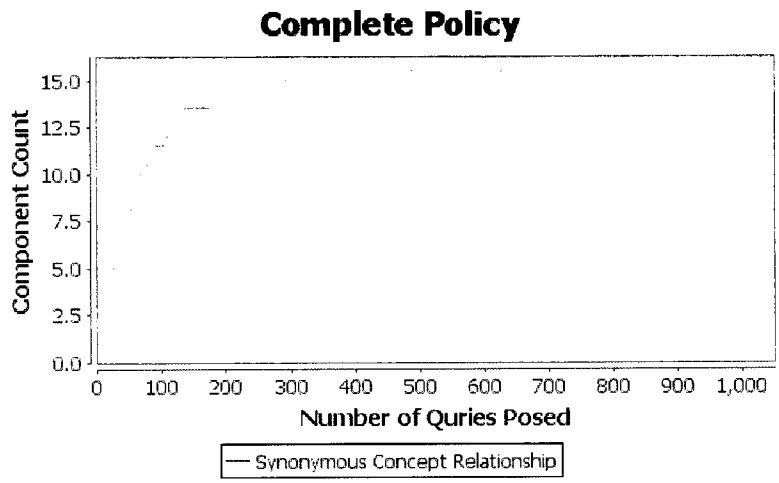

Figure 5.23: Peers with *Complete* semantic agreement remain connected under the *Complete* policy

than the previous case before the network reaches a complete disconnection state. This is due to the fact that the firm maintenance policy is more tolerant towards partially correct answers than the Complete maintenance policy. Peers will reduce the confidence in their outgoing links by values < 1 whenever they receive partially correct answers.

Another difference between this case and the previous is the noticeable difference in the number of executed queries among the three type of answers $\{\perp, *, \sqsubset\}$ before the networks reach the failure state. This implies that given a network where peers are able to return partially correct ($\sqsubset$) 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 ($\perp$) answers. As a matter of fact, the last case is the same as the Complete answer maintenance, see Figure 5.24. This is because, in the last case the query answer contains no partially correct result of any type. Figure 5.27 compares the network behavior for the three relation types.

Figure 5.28, on the other hand, shows that 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. The network becomes stable after excluding partially compatible peers.

As noticed in the previous set of tests (section 5.5.2.3.1) the current test results prove

again that a fault-tolerant query answerer component is an essential component for reliable SP2P systems.

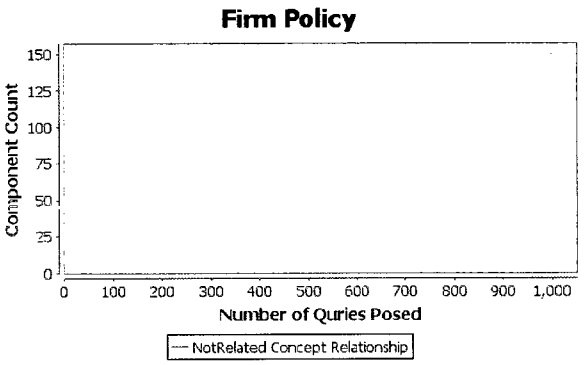

Figure 5.24: Network deterioration under a *firm* policy when the answer is completely faulty ($\perp$)

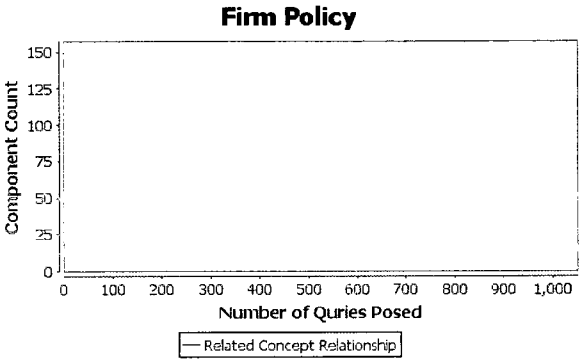

Figure 5.25: Network deterioration under a *firm* policy when the answer is related to the request ($*$)

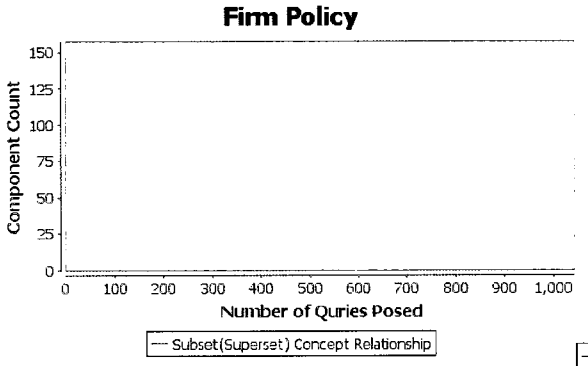

Figure 5.26: Network deterioration under a *firm* policy when the answer is partially asserted ($\sqsubset$)

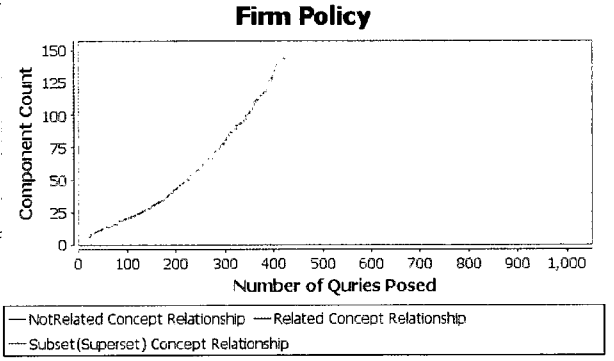

Figure 5.27: Comparing the network behavior for the three relation types ($\perp$,$*$,$\sqsubset$)

5.5.2.3.3 Partial semantic relation maintenance Figures 5.29 and 5.30 show that when SP2P peers return only partially correct answers, but not faulty answers, the network stays connected. The network connectivity remains very similar to a situation where there is no fault in the system (see Figure 5.31). Furthermore, the degree of network connectivity for a situation where the relation between query result concepts and peers' local concepts are $\sqsubset$ and $\perp$ is higher than the previous two cases.

However, Figure 5.32 shows that when the peers return faulty answers, the network deteriorates quickly, and partial maintenance answers become close to those of the Complete semantic maintenance case. This is because, partial relation maintenance

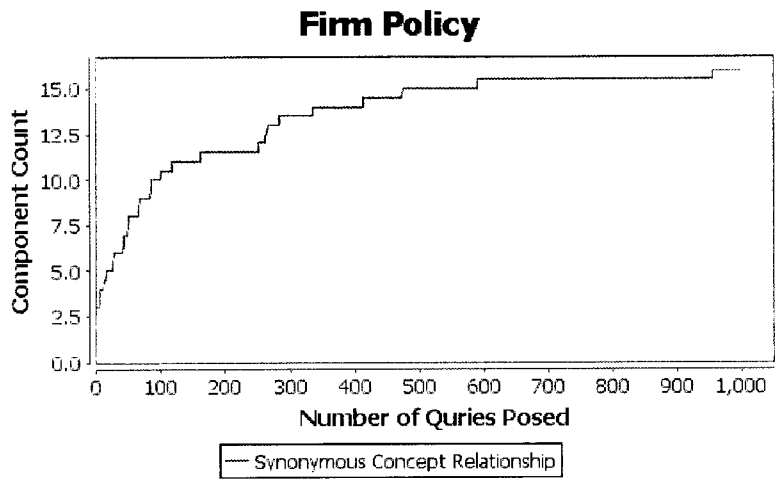

Figure 5.28: Compatible peers stay connected under a Firm policy

does not differentiate between different types of faults. Partial relation maintenance treats all faults as permanent.

Figure 5.33 compares the network deterioration behavior of four types of relationships between query result concepts and peers' local concept, ($\perp$,$*$,$\sqsubset$, $\equiv$), when partial policy is applied. The figure illustrates the main drawback of the partial maintenance policy and the systems that rely upon it, for example Chatty Web system [3]. In the presence of fault in the system, the Partial policy behaves similar to the Complete and Firm policies.

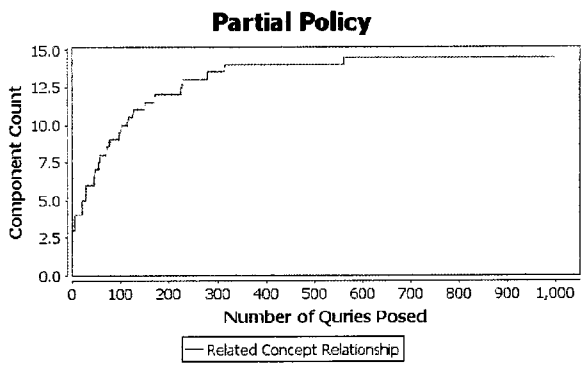

Figure 5.29: Network deterioration under a *partial* policy when the answer is related to the request ($*$)

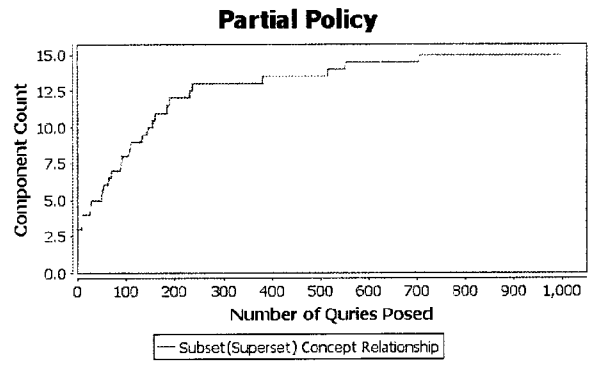

Figure 5.30: Network deterioration under a *partial* policy when the answer is partially asserted ($\sqsubset$)

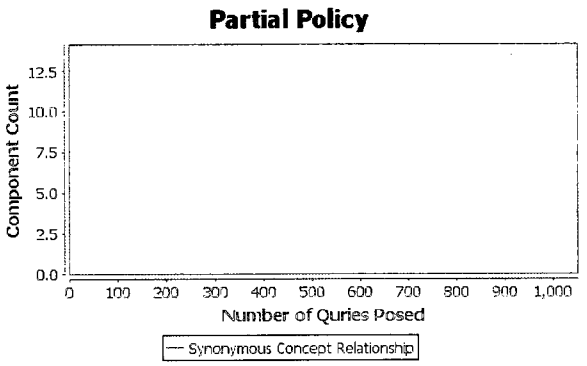

Figure 5.31: Network in a stable state under a *partial* policy

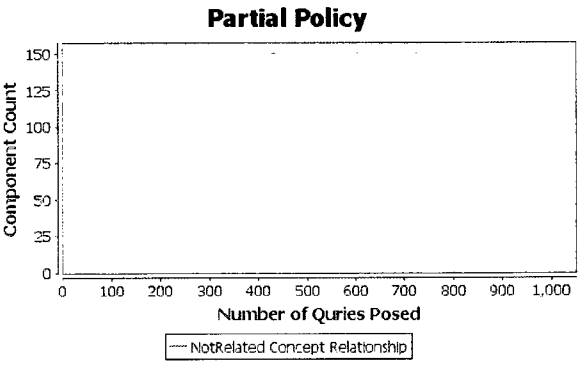

Figure 5.32: Network deterioration under a *partial* policy when answer is completely faulty ($\perp$)

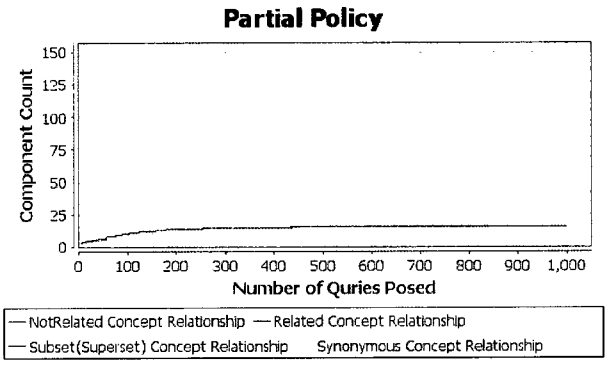

Figure 5.33: Comparing the network behavior for the four relation types ($\perp$,$*$,$\sqsubset$, $\equiv$)

5.5.2.3.4 Benevolent semantic relation maintenance Similar to the Partial answer maintenance policy, Figures 5.34 and 5.35 show that when SP2P peers return only partially correct answers, but not faulty answers, the network stays connected. However, the difference between Partial answer maintenance and Benevolent answer maintenance policies arise when peers return faulty query answers (see Figure 5.36). The number of queries executed in the cases where SP2P systems employ the Benevolent semantic maintenance policy is larger than situations where partial semantic maintenance is used. More specifically, when using the Benevolence policy, the system would be able to process three times more queries than would be processed using a partial policy. A comparison between Figures 5.32 and Figure 5.36 illustrates this improvement in the system fault-tolerance capability.

Comparing Figure 5.38 to its counterparts using other policies shows that the Benevolence policy is the most generous policy resulting in building highly robust and fault-tolerant SP2P system. Furthermore, Figure 5.37 indicates the highest network connectivity degree that a SP2P system could achieve in the described context.

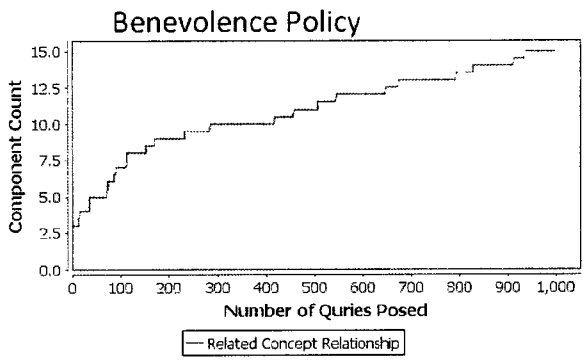

Figure 5.34: Network deterioration under a *benevolence* policy when the answer is related to the request ($*$)

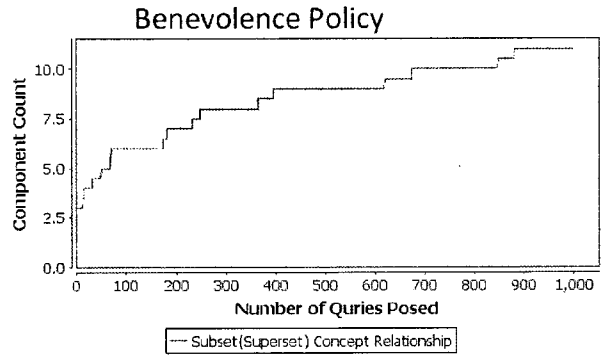

Figure 5.35: Network deterioration under a *benevolence* policy when the answer is partially asserted ($\sqsubset$)

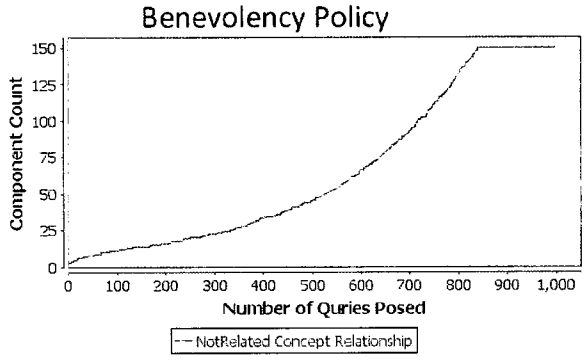

Figure 5.36: Network deterioration under a *benevolence* policy when the answer is completely faulty ($\perp$)

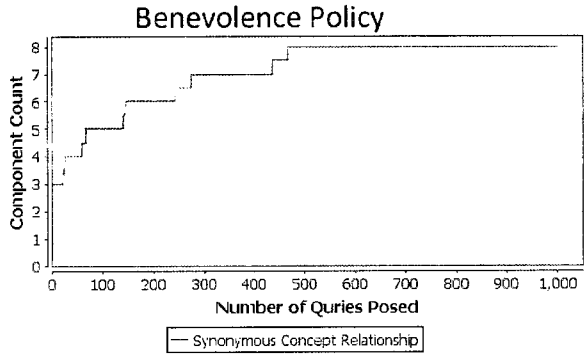

Figure 5.37: *Benevolence* policy enables the highest network connectivity degree

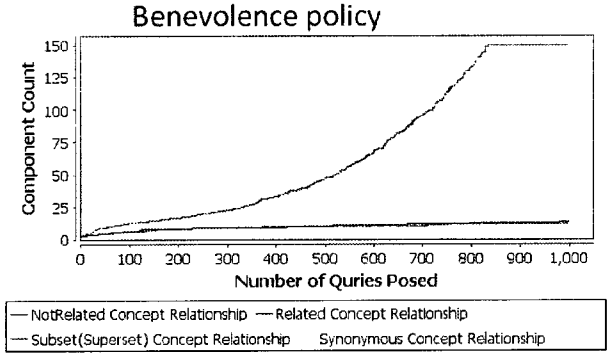

Figure 5.38: Comparing the network behavior for the four relation types ($\perp$,$*$,$\sqsubset$, $\equiv$) using *benevolence* policy

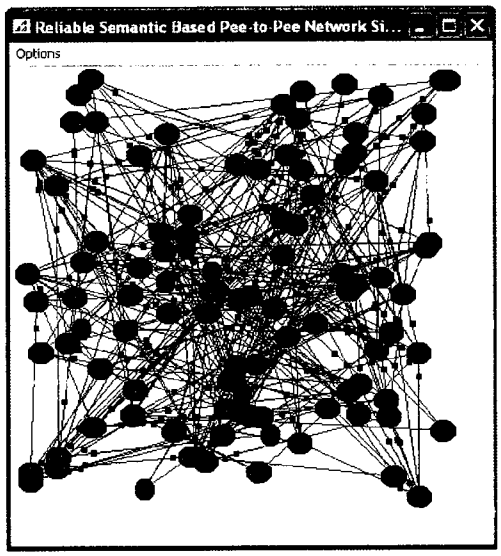

Figure 5.39: Initial SP2P network under a *failure guarded* policy

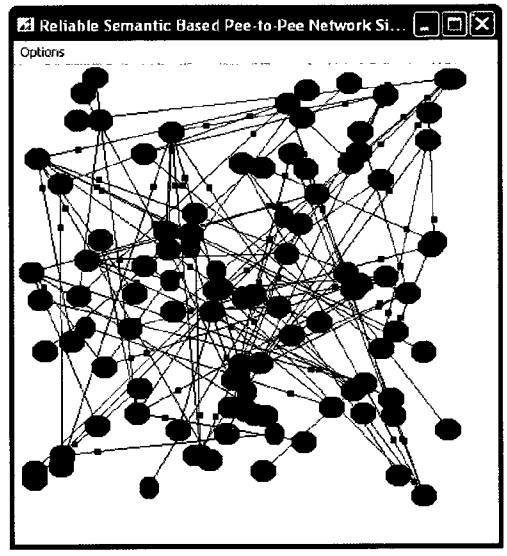

Figure 5.40: Peers always stay connected under a *failure guarded* policy

5.5.2.3.5 Failure Guarded semantic relation maintenance Figures 5.39 and 5.40 show that when peers apply the failure guarded strategy to prevent isolation, they will stay connected even though they might have fewer connections. Figure 5.41 shows that the network component count stays unchanged. As stated before, the continuous connection policy is not achieved for free, in that peers employing this strategy have to accept a high rate of faults for the sake of the connectivity.

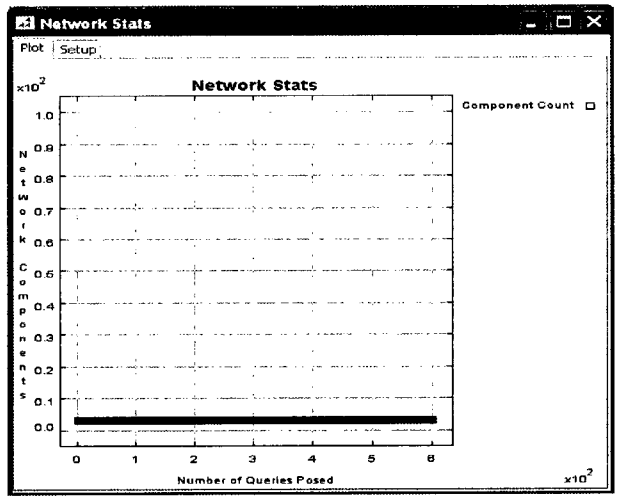

Figure 5.41: Network stays connected under a *failure guarded* policy

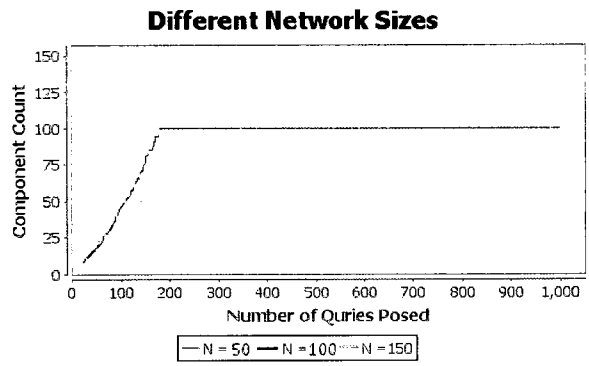

Figure 5.42: SP2P system with *complete* policy for different network size

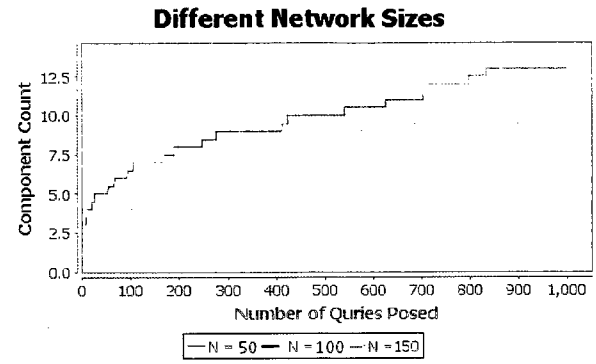

Figure 5.43: SP2P system with *benevolence* policy for different network size

5.5.2.4 GQA's characteristic with different system parameters

So far we have studied the impact of the GQA algorithm and its various policies on the SP2P system reliability. In the following subsections we will study the impact of other SP2P system components on the GQA's fault-tolerant characteristics. These include network size, number of links a peer manages, and SP2P system type.

5.5.2.4.1 Network size We carried out two tests to determine the impact of the network size on the algorithm's ability to improve reliability. In the first test we use a non-fault-tolerant (Complete) policy, and a fault-tolerant (Benevolent) policy in the second one. The number of peers in both tests varied from 50, 100, and 150. In each test, 1000 queries were executed per simulation run, and the experiments were repeated 30 times. The test results report the median values.

Figures 5.42 and 5.43 show the results. The figures indicate that except for scale, network behavior does not change with an increased number of peers in either case, and the fault-tolerant policy is effective in improving network reliability regardless of the number of peers used in the network.

This result is expected, since we believe that the number of query answer responses are constrained by the number of correct answers in the network and the TTL value, instead of network size. The latter limits the number of peers contacted during query propagation, as demonstrated by the test results.

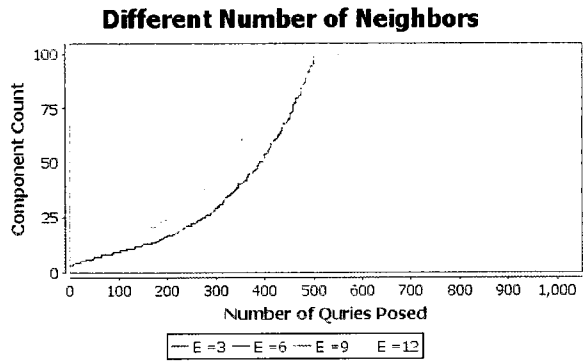

Figure 5.44: Network deterioration under *firm policy* when answer is related to the request (∗) for different link numbers

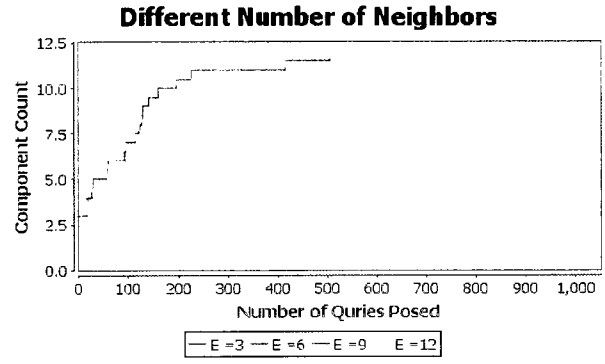

Figure 5.45: Network deterioration under *partial policy* when answer is related to the request (∗) for different link numbers

5.5.2.4.2 Number of links a peer manages In this section, we describe test results for experiments which were carried out in order to determine the effect of the number of neighbors, E, that a peer could have on the network reliability using different GQA's policies. The experiments were carried out with networks of 100 peers and with different numbers of neighbors, E =3, 6, 9, 12. During each experiment 1000 queries were executed, each experiment was repeated twenty times, and median values are reported as results.

Tests results indicate that once peers have a higher number of neighbors, i.e., when E increases, peers take more time to drop-off their links. This result is self-evident since peers with a higher number of neighbors will have to receive a higher number of faulty query results in order to be disconnected entirely (see Figure 5.44).

The results indicate as well that *the choice of the policy is more important than the number of the links.* Figure 5.44 shows when a peer uses the Firm policy, all networks reach a total state of disconnection regardless of the number of neighbors a peer might have. As Figure 5.44 shows, peers reach a disconnection failure state regardless of whether E is equal to 3 or E is equal to 12.

Figures 5.45 and 5.46, on the other hand, show when a more generous and fault-tolerant policy is employed, the system reliability improves. The figures also show that the Benevolence policy outperforms the Partial policy in improving SP2P system reliability, and this result applies for all tested cases in Figures 5.45 and 5.46.

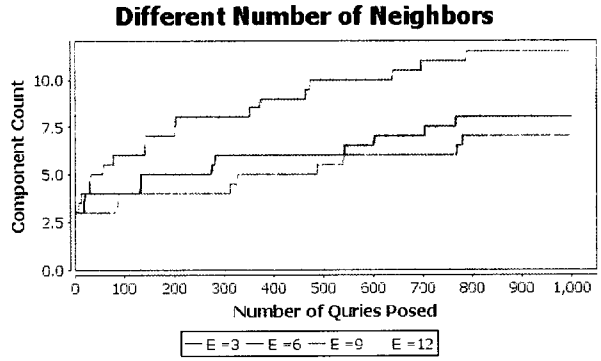

Figure 5.46: Network deterioration under *benevolence policy* when the answer is related to the request (∗) for different link numbers

5.5.2.4.3 SP2P system types Experiments described in the previous subsections that check the effectiveness and suitability of various fault-tolerant policies with regard to the given semantic relationship between query result concepts and peers' local concepts have been reproduced for different type of SP2P systems, namely Irreducible SP2P (IRSP2P) systems.

The environmental settings used in previous testing are reproduced here as well, i.e., a network of 150 peers, a 1000 query execution during each simulation run, simulation are repeated 30 times for each experiment, and the median values of the results are applied again.

The experimental results indicate that the results for an IRSP2P system are not different from the results of an RSP2P system simulation. Figure 5.47 shows that when there is no semantic relationship between query result concepts and the peers' local concept, i.e., $\perp$ relationship, the network deteriorates quickly and entirely. The figure also shows that when the Benevolence policy is employed the network takes longer to deteriorate. This is because of the generous behavior of the Benevolence policy as has been noted repeatedly.

Figures 5.48, 5.49, and 5.50 show as the relationship between the query result concepts and peers' local concepts improve from $*$ to $\sqsubset$ to $\equiv$, (i.e., fewer faults are injected into the system), the differences between fault-tolerance policies emerge, and they are similar to the noted behaviors for the RSP2P system.

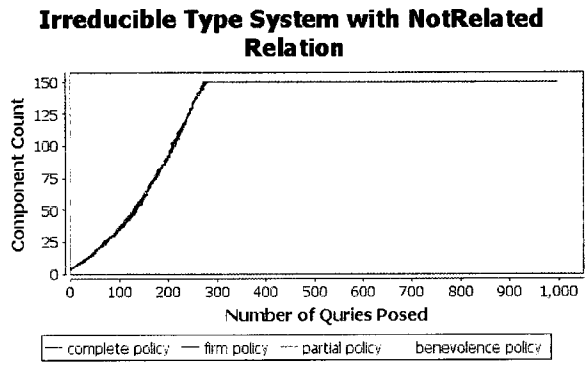

Figure 5.47: Policies' behavior for $\perp$ relationship between query result concepts and peers' local concepts

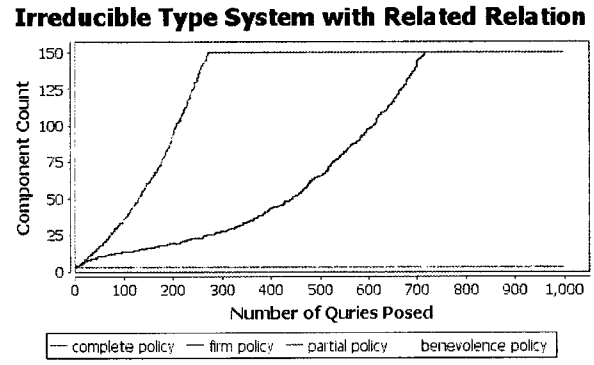

Figure 5.48: Policies' behavior for $*$ relationship between query result concepts and peers' local concepts

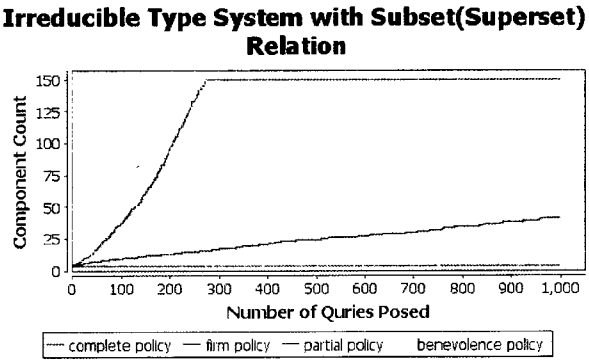

Figure 5.49: Policies' behavior for $\sqsubset$ relationship between query result concepts and peers' local concepts

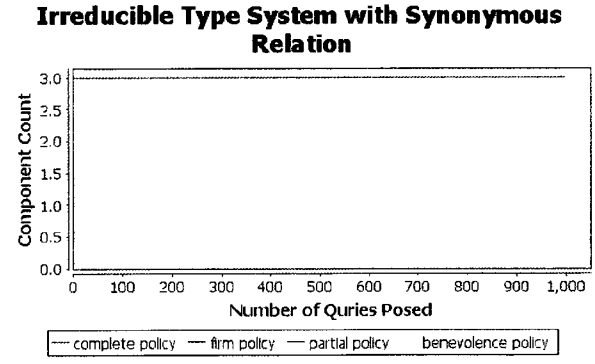

Figure 5.50: Policies' behavior for $\equiv$ relationship between query result concepts and peers' local concepts

5.5.2.5 GQA cost analysis

In this section, we determine the cost of the GQA algorithm in terms of the number of messages required to build a reliable SP2P system. The reliability cost will be demonstrated by considering the message (query) complexities of two different GQA policies: 1. Complete policy, the worst case scenario of system reliability, and 2. Benevolence policy, the best case scenario of system reliability. The message complexity of the other policies could be conceived from these two policies. For example, in the Partial policy, message complexity can either be the same or higher than that of the Complete policy, while it can be the same or lower than that of the Benevolence policy. In the case of Firm policy, message complexity can either be the same or higher than that of the Complete policy, while it can be the same or lower than that of the Benevolence policy. For the Failure Guarded policy, message complexity can either be the same or higher than all other policies.

In order to compute the message complexity of the GQA algorithm, it is important that we recall the following three properties of the SP2P systems described in chapter 3.

1. Peers are connected to d other semantically related peers (acquaintances). The maximum number of peers that can connect to a particular peer p is system parameter and is equal to D, $d \leq D$.

2. The extent to which a query can travel in an SP2P network is controlled by a system parameter, e.g., TTL.

3. The Benevolence policy enables peers to tolerate up to k incorrect query answers before cutting ties with neighboring peers.

Figure 5.51 is an example of an SP2P network where Peer p_1 is connected to d other peers. The thick lines indicate the links which the message (query) can navigate during query forwarding starting from link l_{12}. The length of a path is constrained by TTL and equal to 4.

Definition 7 *We refer to the total links that could be navigated in an SP2P System for a given TTL value by message coverage* $\vec{C}$.

An example of message coverage $\vec{c_1}$ starting from peer p_1 and link l_{12} with a TTL value equal to 4 is highlighted with dotted line in Figure 5.51. Dashed arrows in the figure indicate that peers at the end of dashed lines are not queried. This is because the queries already have reached the maximum number of hops set by the TTL parameter.

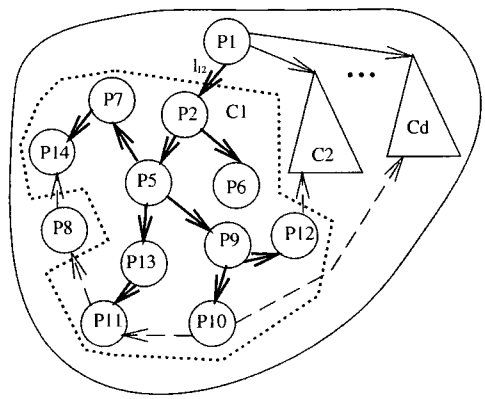

Figure 5.51: An SP2P system with various message coverage

The message complexity using link l_{12} for the two policies: Firm and Benevolence, where query answers are (1) correct (partial or Complete), (2) incorrect, and (3) a combination of correct/incorrect answers are as follows:

A) When the query answers are correct, the message complexity is $\leq \vec{c_1}$ for both policies.

B) When the query answers are entirely incorrect, the number of query hops between the two policies vary as follow:

   i. The Firm policy uses $\leq \vec{c_1}$ messages. This is because the Firm policy does not give more chances for faulty peers.

   ii. The Benevolence policy could use up to $k\vec{c_1}$ messages. That is, $(k-1)\vec{c_1}$ more messages than the previous case, where k is Benevolence policy's constant. It determines the number of times the link l_{12} will be reused before being dumped when faulty answers are returned. The extra $(k-1)\vec{c_1}$ messages could be wasted when the fault is permanent and all subsequent queries do not result in any correct answer.

C) When the query answers alternate between correct and incorrect, the number of query hops between the two policies vary as follow:

i. For the Firm policy the message complexity would be $\vec{c_1}$, but the link l_{12} between peers P1 and P2 might or might not be cut. This depends on the the order of receiving query answer, (i.e., when incorrect answers are received first, the link would be cut).

ii. The message Complexity for the Benevolence policy varies between $\vec{c_1}$ to $k\vec{c_1}$, based on how many faulty messages are transient or permanent.

When using links l_{12}, l_{13}, l_{14}, ... , l_{1d} the message cost for building a reliable SP2P system in the worse case scenario would be KC where $C = \sum \vec{c_1}, \vec{c_2}, \vec{c_3}, ..., \vec{c_d}$, and $K = \sum k_1, k_2, k_3, ..., k_d$. C is smaller than the total number of links L in the network. This is because $d < L$. This gives us a message complexity (MC)
$c_i \geq MC \geq KC$, $i = 1, 2, ...d$, $MC \Rightarrow O(L)$

5.5.2.6 Summary

We conclude this section by stating that the Benevolence policy remains the most desirable policy for most SP2P applications. The Partial policy could be used for applications when there is no chances for transient-type faults to occur (an almost impossible situation), and Complete and Firm policies should not be applied under normal circumstances. They would be acceptable for applications where there is a high demand for query retrieval accuracy and no room for computational resource and/or message wasting.

Furthermore, despite the effect that network size, the number of neighbors a peer could have, and the SP2P system type could have on the system's behavior (for example on the length of the time that a network gets into a particular defragmentation state), the choice of fault-tolerant policy remains as a number one factor that determines system reliability.

5.5.3 Time redundant query answerer algorithm

In this section we present a novel approach for maintaining the semantic relationship among peers in the SP2P systems. The solution enables SP2P systems to detect the transient semantic mapping faults and avoid their negative impact on peer's collaboration in SP2P systems. The solution is based on the *time redundancy* technique where time redundancy refers to using an additional time to replicate queries and verifying the query answers' consistency. The solution is embedded into an SP2P's Query Answer component and is called Time Redundant Query Answerer Algorithm (TRQA).

5.5.3.1 TRQA algorithm's steps and procedures

The procedure of our proposed algorithm for tolerating non-permanent semantic mapping faults comprises of two main parts: fault detection and fault recovery. However, we start first by describing two of the design decisions made during algorithm development. The driving force in our design decisions was the balance between the following three factors: a) obtaining the best possible results, b) lowering the cost of the algorithm, and c) the simplicity in the implementation.

The design decisions are:

1. The resubmitted queries are sent over the same paths which have generated incorrect answers. The ID for the peers which have been queried during query forwarding are stored in the query and presented to the query initiator at the end of the query forwarding. The path information is then used for query resubmits.

2. Only complete answers,(i.e., no partial answers) are considered to be correct when query answers are evaluated for correctness. This decision enables us to determine the effectiveness of the TRQA algorithm in the worst case scenario. The answer evaluation is implemented by reusing the Complete policy defined for GQA algorithm.

The algorithm's steps are:

1) To detect faults, peers will be tested with repeated queries as follows:

a) Submit up to k sequential queries in place of one query every time an incorrect query answer is received. Queries are separated from each other by a time Δ_1. For instance, if k is set to 2 then the origin query and its clone will be separated by Δ_1 time. That is, the second query will be posed at $t_0 + \Delta_1$, where t_0 is the time for initial query and Δ_1 is the delay time between the two sequential queries. The system designer determines the maximum transient-pulse duration Δ_1 that the system must tolerate.

b) Query answers from replicated queries are compared for consistency. The inconsistency among answers for the same query is a deciding criterion for the transient fault occurrences. The consistency checking leads to the following two cases:

(i) If query answers are consistent and incorrect then the querying peer concludes that the queried peer is incapable of providing an answer to the query. Hence, it is *permanently faulty* relative to the posed query.

(ii) If query answers are inconsistent, then a *transient fault* must have occurred, and an action should take place to eliminate its negative impact.

2) A transient fault recovery action is as follows:

In order for queries to eliminate/reduce the impact of the transient faults, query resubmission needs to take place. This happens after waiting for Δ_2 length of the time from the last time a transient fault is detected and query re-submission could take place. The query re-submission can be repeated up to k times. The Δ_2 value and the number of query re-tries, k, are system parameters. These values will be set by the system administration in such a way that a system will maximize recall for the least additional queries. When needed, these values could be determined experimentally. The activity diagram for the TRQA algorithm is provided in Figure 5.52 shows the algorithm's procedures and the dependency between them, and how the operation could be achieved.

5.5.3.2 Pseudocode for TRQA algorithm

In this section we provide TRQA algorithm's pseudocode that consists of seven procedures. The pseudocode for each procedure is provided. The procedures are: 1. answerHandler, 2. ProcessResult, 3. compareResults, 4. isTransient, 5. resubmitQuery, 6. transientHandler, and 7. removeUnusedLinks. The sequence of procedure

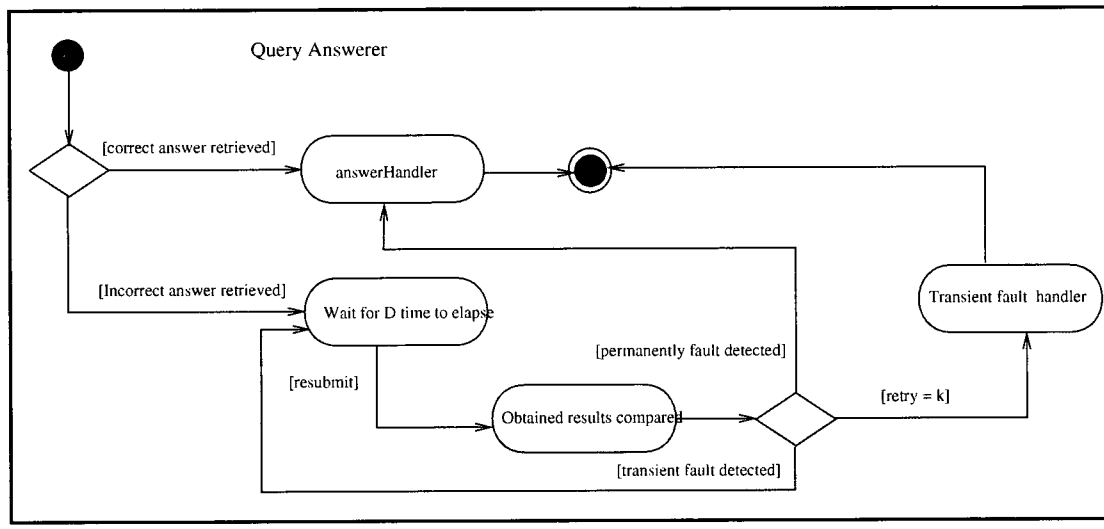

Figure 5.52: Activity diagram for TRQA algorithm

execution follows the activity diagram described in Figure 5.52.

Algorithm 2 Answer handler procedure which calls on a function for further checking incorrect answers

```
procedure ansawerHandler (Peer initiator, QueryAnswer result, AnEdge
                                                usedEdge, Peer sender)
1:  fault_tolerant_policy ← TimeR_fault_tolerant_policy
2:  int count ← 0
3:  int nocq ← Query_size /*number of concepts originally in the query */
4:  Set temp ← QueryResultContent
5:  If(temp.size > 0)
6:    For (i ← 0 To temp.size)
7:        If ( initiator.Resource.contains(temp.elementAtPosition(i)))
8:          count ← count + 1
9:        End If
10:   End For
11: End If
12: double newStrength ← 0
13: If(count = nocq)
14:       newStrength ← anEdge.getStrength + awardValue
15:       anEdge.setStrength(newStrength)
16: Else
17:       If (fault_tolerant_applied = true)
18:       /* function call */
19:         ProcessResult(initiator, query, anEdge, sender)
20:       Else
21:         newStrength ← anEdge.getStrength - punishValue
22:         anEdge.setStrength(newStrength)
23:       End If
24:   End If
25:   /* function call */
26: removeUnusedLinks(initiator, anEdge)
27: End procedure
```

Algorithm 3 Process-result procedure enable queries re-try for detecting transient faults.

```
procedure ProcessResult(Peer initiator, Query query, AnEdge anEdge)
1:  //the initial result
2:  result1 = initiator.retrieveQueryResult(query)
3:  Set tempPath ← φ
4:  tempPath.add(initiator.getResultPath)
5:  wait(Delta1)
6:  //resubmit after waiting for Delta1
7:  result2 ← resubmitQuery (tempPath, query)
8:  //function calls
9:  If(isTransient(compareResults(result1, result2)))
10:   do{
11:       Set tempPath2 ← φ
12:       tempPath2.add(initiator.getResultPath)
13:       wait(Delta2)
14:       //result after other tries
15:       resubmitQuery( tempPath2, query)
16:       result3 = Initiator.getResult
17:   } while(retry < k ) // k is constant and its value ≥ 2
18:   //function call
19:   transientHandler (initiator, result3, anEdge, query)
20: Else
21:   newStrength = anEdge.getStrength - punishValue
22:   anEdge.setStrength(newStrength)
23: End procedure
```

Algorithm 4 Compare result procedure compares repeated answers for their consistency

```
procedure String compareResults (Query firstResult, Query secondResult)
1:  String repeatedAnswers ← φ
2:  Set resultContent1 ← φ
3:  Set resultContent2 ← φ
4:  If (((firstResult = null) And Not(secondResult = null))
5:    OR( Not(firstResult = null) And (secondResult = null)))
6:    repeatedAnswers ← "different"
7:  Else
8:    int size1 ← resultContent1.size
9:    int size2 ← resultContent2.size
10:   boolean aConcept ← true, allconcepts ← false
11:   IF Not( size1 = size2)
12:     repeatedAnswers ← "different"
13:   Else
24:     For (i← 0 To size1)
15:       Concept concept1 = resultContent1.elementAtPosition(i)
16:       Concept concept2 = resultContent2.elementAtPosition(i)
17:       If Not (concept1 = concept2)
18:         aConcept ← false
19:       End If
20:     End for
21:     If(aConcept= true)
22:       allconcepts ← true
23:     End If
24:     If(allconcepts OR repeatedAnswers.isEmpty
25:       repeatedAnswers ← "same"
26:     End If
27:   End If
28: End If
29: return repeatedAnswers
30: End procedure
```

Algorithm 5 A boolean function isTransient returns true if the result of the same query at different times were different

procedure boolean isTransient (String repeated_answer)
1: *//result of the same query at two different times are different*
2: *//there is a possibility of transient fault*
3: boolean transientFault ← false
4: If(repeated_answer = "different")
5: transientFault ← true
6: End If
7: return transientFault
8: **End procedure**

Algorithm 6 The resubmit procedure sends the same query along the paths were the faulty answers have generated

```
procedure void resubmitQuery (Set path, Query query)
1:  Query result ← φ
2:  Set QueryOriginContent = query.getOriginContent
3:  query.setQueryContent(QueryOriginContent)
4:  int length ← path.size
5:  If(length > 0)
6:    Peer p ← (Peer)path.firstElement
7:    p.checkPaths (path, query)
8:  End If
End procedure

procedure void checkPaths (Set path, Query q)
1:  Set concepts ← match (q.getQueryContent)
2:  If(q.forwardMethod = "Irreducible")
3:    //function call
4:    concepts ← replaceDroppedConcepts (concepts, q)
5:  End If
6:  //sets query content to the outcome of the matching function
7:  q.setQueryContent(concepts)
8:  path.removeFirstElement
9:  IF( path.size > 0) // not end of the re-try path
10:   Peer p = path.firstElement
11:   //recursive call on different peer on the path
12:   p.checkPaths (path, q)
13: End IF
14: pathResult ← q
End procedure
```

Algorithm 7 After k query re-try, the transient handler procedure evaluate results for their correctness /incorrectness

```
procedure void transientHandler(Peer initiator, Answer latestResult,
                                AnEdge anEdge, Query query)
1:  int count ← 0
2:  //number of concepts in original query
3:  int nocq ← query.getOrigincontent.size
4:  If Not(latestResult = null)
5:    Settemp ← latestResult.getContent
6:      If (temp.size > 0)
7:        For(i ← 0 To temp.size)
8:          If ( initiator.Resource.contains (temp.elementAt(i)))
9:            count ← count +1
10:       End if
11:     End for
12:   End If
13: End If
14: Float newStrength ← 0
15: If(count = nocq)
16:   newStrength ← anEdge.getStrength + awardValue
17:   anEdge.setStrength(newStrength)
18: Else
19:     newStrength ← anEdge.getStrength - punishValue
20:     anEdge.setStrength(newStrength)
21: End If
End procedure
```

Algorithm 8 Remove-unused-link procedure cut links between peers when the semantic relationship drops below a given threshold

```
procedure void removeUnusedLinks (Peer initiator, AnEdge edge)
1:  int strength ← edge.getStrength
2:  IF(strength ≤ 0)
3:    Node toneighbor ← edge.getTo() /*call to function getTo */
4:    toneighbor.removeInEdge(edge)
5:    initiator.removeOutEdge(edge)
6:  End If
End procedure
```

5.5.3.3 Fault simulation

We mentioned previously in section 5.5.2 that for fault simulation we focus on *ontology changes* to generate faults. However, the aim of the TRQA algorithm is different from GQA algorithm. The TRQA aims to detect the transient semantic mapping faults and avoid their negative impact on peer's collaboration in SP2P systems. Thus, the way ontology change is used to simulate faults here is slightly different from the way faults were generated for testing GQA algorithm.

Each peer is provided with two different versions of the same ontology, O_1 and O_2. These ontologies are different from each other on the number of concepts each ontology comprises; O_2 is an extended version of O_1. Peers are assigned one of the ontologies every t milliseconds (ms), t is a simulation parameter initialized on the simulation start up. That is, peers' resources alternate between O_1 and O_2 every t milliseconds.

During simulation, queries are created, forwarded to other peers, and mapped against local resources, Hence, an alternate in peers' resources, as described above, could result in fault generation as follows:

A query that has been created with version 1 of the ontology O_1, and presented to a peer that has a version 2 of the same ontology, or vice versa, could result in an answer which would not be worthy to the querier peer.

The TRQA algorithm tries to detect the described scenario and eliminate/reduce its negative impact on peers' collaboration.

5.5.3.4 SP2P system simulations

Three instances of the SP2P system will be simulated and used for testing. The system instances are: 1. Reducible SP2P system , 2. Irreducible SP2P system, and 3. Fault-tolerant SP2P system.

The difference between Reducible and Irreducible SP2P systems is in how each one of these two systems deals with the query concept dropping during query routing in the SP2P network. The query concepts could possibly be dropped when the query receiver peers do not use the same or similar concepts.

The Reducible SP2P system allows queries to drop concepts during routing. Query

concepts which do not match any of the query receiver's local concepts will be removed from the query. The Irreducible SP2P system, on the other hand, does not drop query concepts. Query concepts which do not match any of the query receiver's local concepts will be retained in the query. An example of the first instance of the SP2P system is the Chatty Web [3] system, and instance of the second SP2P system is Piazza [49] system.

When the Reducible or the Irreducible SP2P system is combined with verification of incorrect query answers, we have a new instance of an SP2P system. We call this new SP2P instance a Fault-tolerant SP2P system. P2PSLN [48] could be considered an example of a Fault-tolerant SP2P system. This is because P2PSLN calls, even though never implemented, for *checking* the semantically related peers for ontology changes when they repeatedly return incorrect query answers.

Our reference framework based simulation enables peers to resubmit queries for verifications once they have received incorrect answers. Hence, in addition to building systems such as Chatty Web and Piazza, the simulation allows building the P2PSLN system as well. All the three systems will be simulated and tested for the peer relation improvements they gain when employing the TRQA algorithm.

5.5.3.5 Test settings and experiment results

Continued collaboration among SP2P peers (i.e., preventing SP2P network from fragmentation) due to an unintentional event, while peers are executing queries, manifests the effectiveness (i.e., is the metric) of a TRQA algorithm on handling transient mapping faults. Several tests are carried out to determine 1. the effectiveness of the TRQA algorithm, and 2. the effect of the algorithm's parameters on the algorithm's behavior. These parameters include Δ_1, Δ_2, Ontology update time t, and ontology update rate u (see Table 5.12 for the description of these parameters).

For each test, a network of up to 100 nodes are used, 1000 queries are executed in each simulation run, and the simulation run is repeated up to thirty times; the median value of runs is reported as test results. In addition to the parameters need for the simulation configuration which is provided in Table 5.11, Table 5.12 shows the list of additional parameters needed for testing the TRQA algorithm along with their descriptions.

Simulation Parameter	Parameter Description
Δ_1	The length of time that a peer P waits before it re-submits a query q along the path T where an incorrect answer was received.
Δ_2	The length of time that a Peer P waits before it re-submits a query q for the k times along the path T where an incorrect answer was received.
ontologyUpdateTime (t)	Peers are assigned a new version of ontology after **ontology update time** elapses.
ontologyUpdateRate (u)	Sets the percentage of the peers that are assigned a new version of ontology after ontology update time elapses, e.g. 100%, 50%, 20%, etc.
k	Sets the number of times a query is re-tried when transient faults are detected. In our simulation k is constant and its value set to 2

Table 5.12: SP2P configuration parameters for TRQA algorithm

5.5.3.5.1 Fault-tolerant semantic mapping In this section, we test the effectiveness of the TRQA algorithm on endowing SP2P systems with the ability to tolerate transient semantic mapping faults. Tests are performed on both **Reducible** and **Irreducible** SP2P systems. Irreducible systems are different from Reducible SP2P systems by the fact that Irreducible SP2P systems do not discard uncomprehended query concepts during query translation and forwarding among multiple peers. Table 5.13 lists the values of variable used for both tests. For each test, a network of 100 nodes was used, 1000 queries were executed in each simulation run, and the simulation run was repeated twenty times; the median value of runs is reported as test results.

In both cases, the results indicate that the TRQA algorithm improves SP2P system's ability to tolerate transient mapping faults substantially. The semantic relationship for both Reducible and Irreducible SP2P systems is improved by 20% to 27% respectively. That is, the semantic relationship drops among peers in SP2P systems employing the TRQA algorithm is lower by 20% to 27% than the SP2P systems running without TRQA algorithm. The number of component counts for SP2P Reducible type systems dropped from 80 to 58 components, and for Irreducible type systems from 65 to 53 (see Figures 5.53 and 5.54).

Δ_1	Δ_2	ontology update time (ms)	Ontology update rate
1000	300	0	100%

Table 5.13: Fault-tolerant semantic mapping test parameters

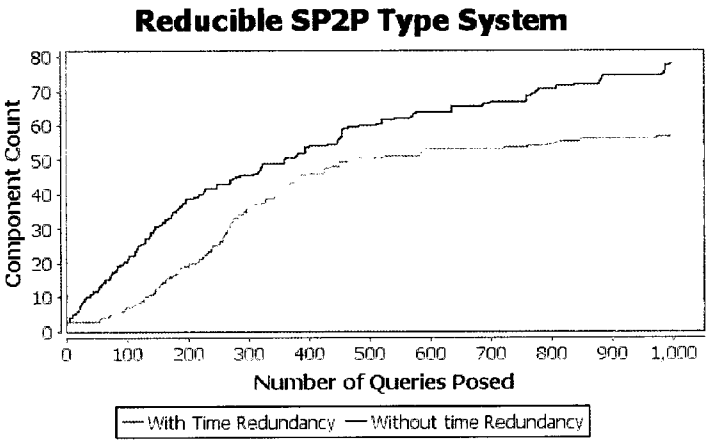

Figure 5.53: Reducible SP2P system with and without fault-tolerance capability

Furthermore, the test results show that the Reducible SP2P system is more vulnerable to network disconnection problems than Irreducible SP2P systems. This is due to the fact that not dropping query concepts during query forward could give a better chance for correct answers to be retrieved.

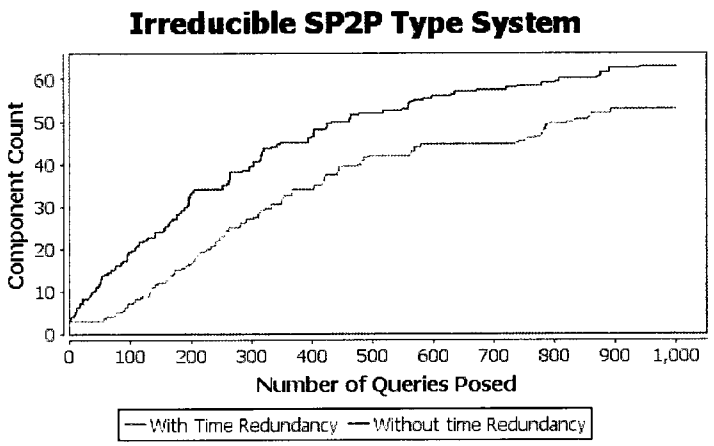

Figure 5.54: Irreducible systemSP2P with and without fault-tolerance capability

5.5.3.5.2 Delay time between query resubmits In this section, the effect of the *wait* function, the length of time peers wait before resubmitting queries, is examined. Two sets of tests are performed for Reducible(e.g. Chatty Web) system, and Irreducible (e.g. Piazza) SP2P systems. In the first set of tests, the value of Δ_1 is constant and is set to 1000 ms, and the value of Δ_2 is set to four different values: 250, 750, 1500, and 2000 ms. In the second set of tests, the value of Δ_2 is constant and set to 1000 ms, and the value of Δ_1 is set to four different values: 250, 750, 1500, and 2000 ms. Other parameters such as ontology update time t and ontology update rate u which are TRQA specific parameters are set to 100% update rate, and continuous ontology update respectively. Table 5.14 presents parameter values for both sets of tests. For each test, a network of 50 nodes is used, 1000 queries are executed in each simulation run, and the simulation run is repeated twenty five times; the median value of runs are reported as test results.

	Δ_1	Δ_2	SP2P System
First test	1000	250, 750, 1500, 2000	Reducible (Chatty Web, P2PSLN)
	1000	250, 750, 1500, 2000	Irreducible (Piazza)
	Δ_1	Δ_2	SP2P System
Second Test	250, 750, 1500, 2000	1000	Reducible (Chatty Web, P2PSLN)
	250, 750, 1500, 2000	1000	Irreducible (Piazza)

Table 5.14: The Δ values used for testing SP2P systems

Figures 5.55, 5.56, 5.57, and 5.58 present test results. The test results indicate that even though the differences between tested scenarios remain very slim, i.e., the number of network components varies from 20 to 25 components, the TRQA algorithm is effective on lowering the effect of transient faults for a wide range of Δ_1 and Δ_2 values.

It is evident from the results that on the occasion of transient faults, a quick recovery action, low value of Δ_2, gives a high chance for incorrect query results to be adjusted, and this observation is correct for both Reducible and Irreducible SP2P systems. Figures 5.55 and 5.56, indicate that a slightly better result is obtained when the value of Δ_2 keeps close and is less than the value of Δ_1 (Δ_1=1000, and Δ_2=750 ms) than the case where the value of Δ_2 is $>$ than Δ_1.

Figures 5.57 and 5.58 indicate that the above observations also hold when Δ_2 becomes

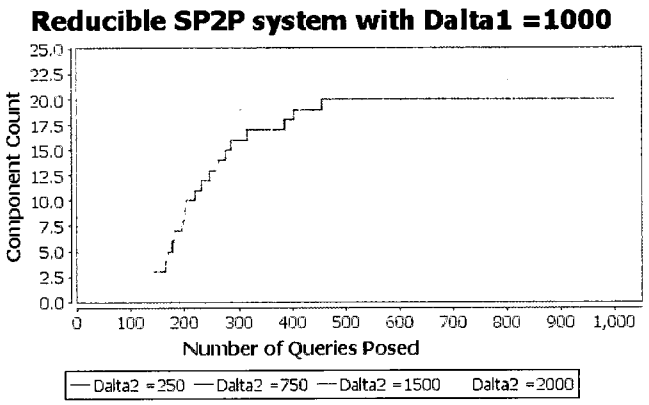

Figure 5.55: Test results for the *Reducible* SP2P system simulation when $\Delta_1 = 1000$ and Δ_2 takes different values

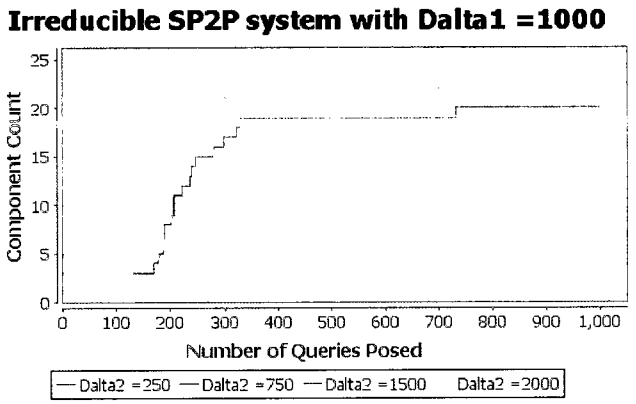

Figure 5.56: Test results for the *Irreducible* SP2P system simulation when $\Delta_1 = 1000$ and Δ_2 takes different values

constant and Δ_1 takes different values. The fact that peers update their ontology continuously, and all network peers update their ontology has an influence on the results. These issue will become clearer once we test the ontology update rate and the ontology update time parameters in the following sections.

One last observation which is common to all conducted tests in this section is that higher values of Δ_1 and Δ_2 mean that peers' reaction toward faults start late. This conduct is reflected in the test results causing result curves to start with different initial stability periods.

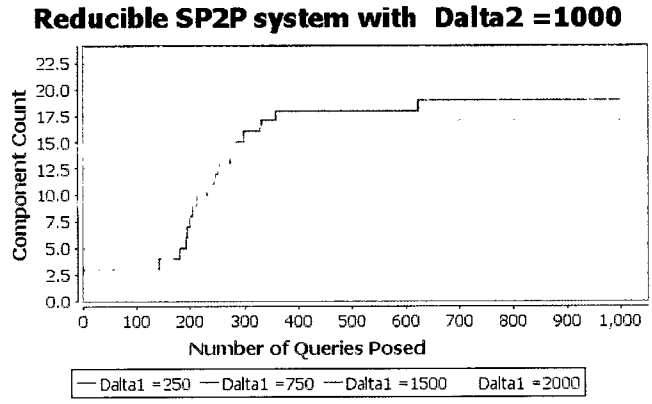

Figure 5.57: Test results for the *Reducible* SP2P system simulation when $\Delta_2 = 1000$ and Δ_1 takes different values

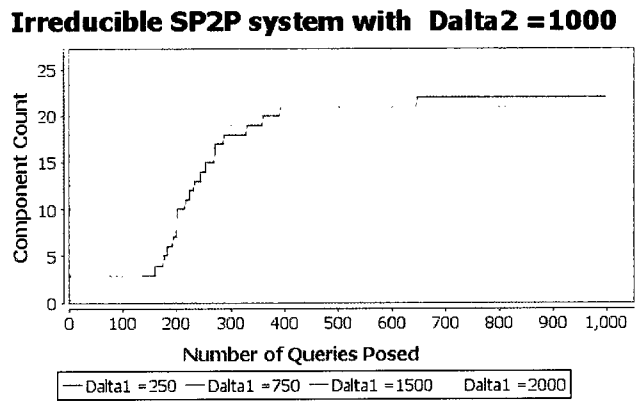

Figure 5.58: Test results for the *Irreducible* SP2P system simulation when $\Delta_2 = 1000$ and Δ_1 takes different values

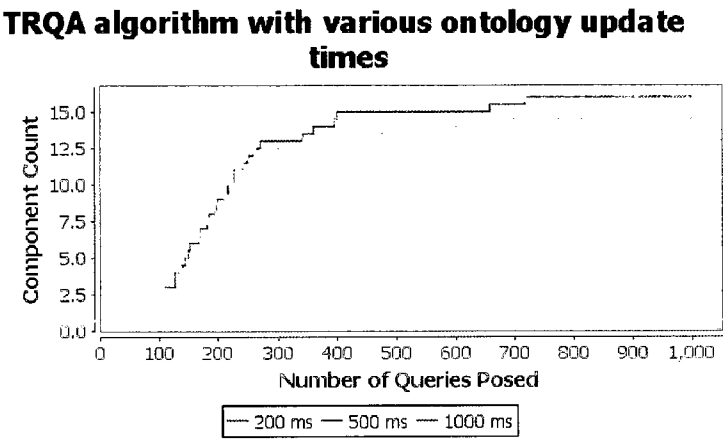

Figure 5.59: Ontology update time's effect on the TRQA algorithm

5.5.3.5.3 Ontology update times In this section we study the effect of the ontology update time t on the TRQA algorithm behavior. Each peer manages two different versions of the same domain ontology O_1 and O_2. Peers are assigned one of the ontology every t milliseconds (ms), where t is a simulation parameter initialized on the simulation start up. Table 5.15 lists the values of variable t along with some other parameters which are specific to the TRQA algorithm and are used for these testings. For each test, a network of 50 nodes are used, 1000 queries are executed in each simulation run, and the simulation run is repeated thirty times; the median value of runs is reported as test results.

Test results depicted in the Figure 5.59 indicate that the variation in t value has some limited effect on the outcome of results. The limitation of the influence is evident by the fact that the network component counts vary between 14 to 18 components only. The test outcomes also show that when the value of t is low ($t = 200$), the network connectivity degree is higher than the case when ontologies change less frequently (t is high, e.g. 1000). This result is due to the fact that frequent ontology changes offer an opportunity for fault adjustments, even better than the reduction in faults that might occur as a result of low ontology changes.

Δ_1	Δ_2	ontology update time (ms)	Ontology update rate
500	300	200, 500, 1000	100%

Table 5.15: Ontology update time testing parameters

5.5.3.5.4 Ontology update rate In this section we determine the effect that an ontology update rate u could have on the behavior of the TRQA algorithm. By u we refer to the percentage of network peers that change ontologies after ontology update time t elapses during simulation runs. For example, $u = 20$ means that 20% of total peers in a network will alternate their ontologies between O_1 and O_2 every t ms. Table 5.16 lists the values of variables used in studying the effect of ontology update rate. For each test, a network of 50 nodes is used, 1000 queries are executed in each simulation run, and the simulation run is repeated twenty times; the median value of runs is reported as test results.

Test results are shown in Figures 5.60, 5.61, 5.62 and 5.63. The results indicate that the effect of ontology update rates on the behavior of the TRQA algorithm is limited. The limitation of the u influence is evident by the fact that the network computer counts range from 17 to 20 components for a wide spectrum of updates: 10%, 25%, 50%, and 100%.

Furthermore, test results indicate that lowest number of network components is achieved when 50% of peers update their ontologies. The slightly better performance of 50% update rate could be due to the fact that this rate results in the required number of ontology changes needed for query concept adjustments. On the other hand, too low/high update rates could lower the chances for queries to adjust themselves causing a lower performance than the cases when update rate is 50%.

Δ_1	Δ_2	t	u
1000	300	300	10%, 25%, 50%, 100%

Table 5.16: Ontology update rate testing parameters

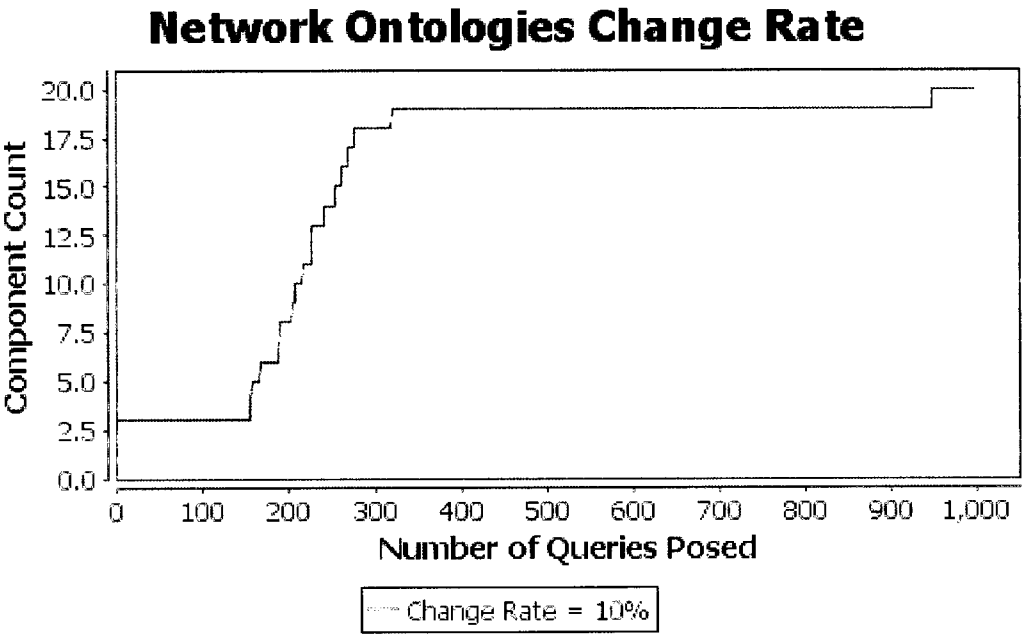

Figure 5.60: Ontology update rate's effect on the TRQA algorithm, change rate =10%

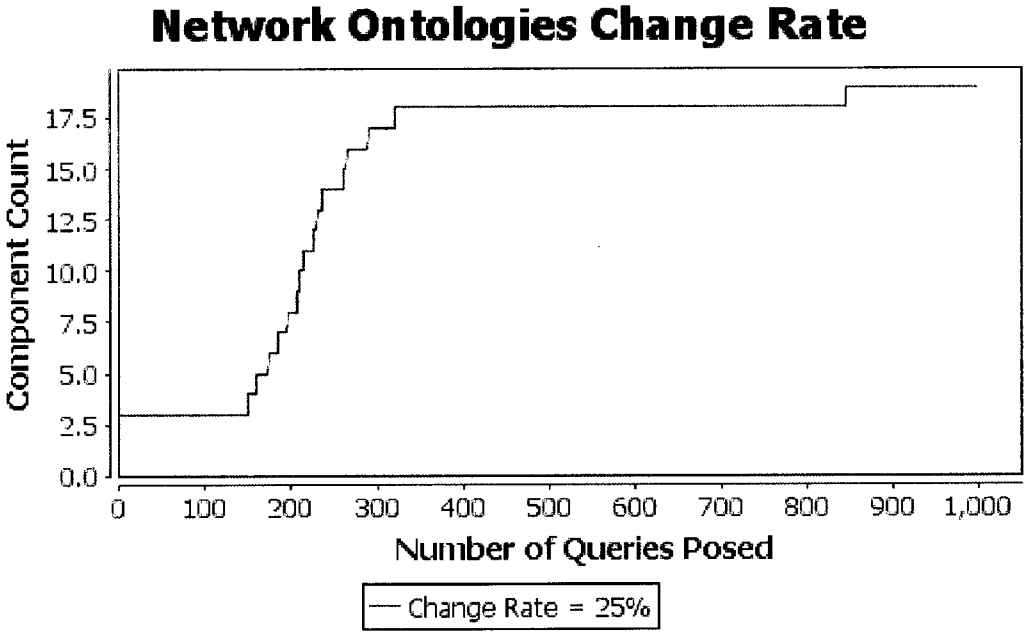

Figure 5.61: Ontology update rate's effect on the TRQA algorithm, change rate =25%

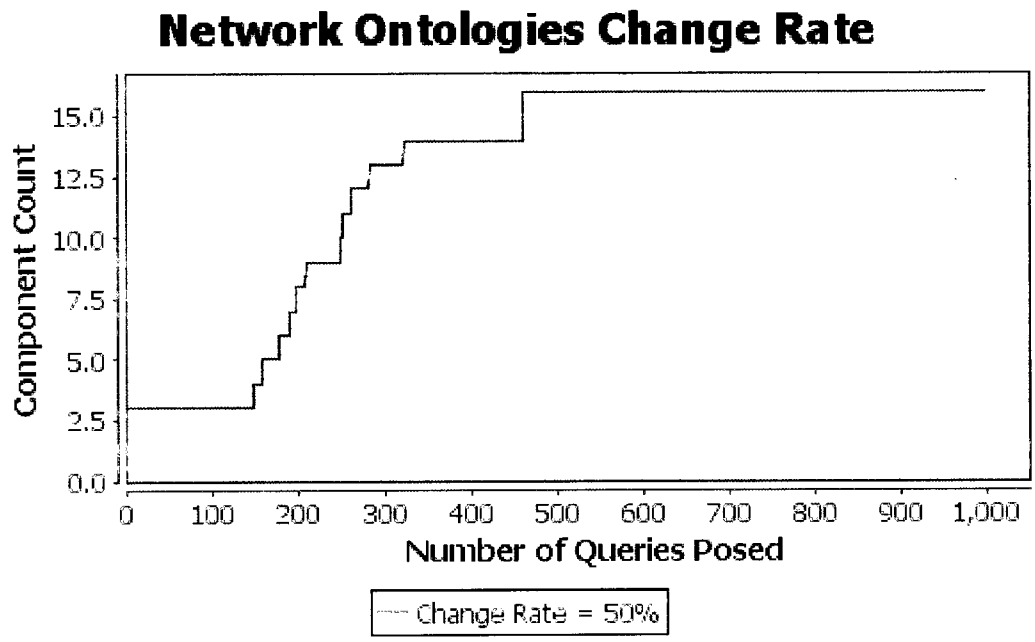

Figure 5.62: Ontology update rate's effect on the TRQA algorithm, change rate =50%

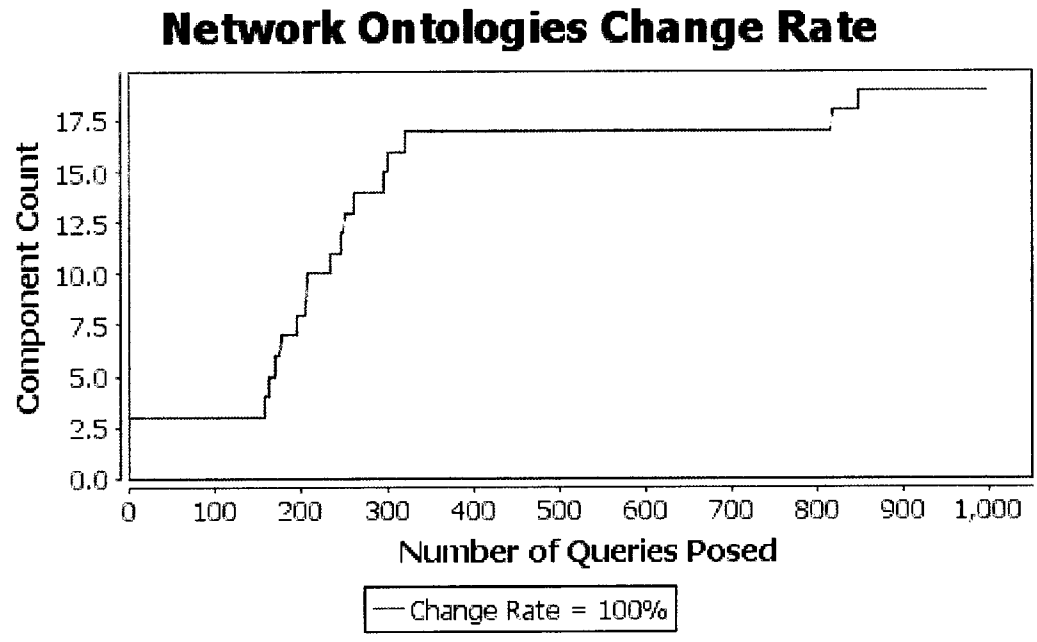

Figure 5.63: Ontology update rate's effect on the TRQA algorithm, change rate =100%

5.5.3.5.5 Ontology update time and rate In this section we combine the test parameters used in sections 5.5.3.5.3 and 5.5.3.5.4 to further study the effect of these parameters on the behavior of the TRQA algorithm. Four new set of tests are carried out for this purpose and test parameters and their values are provided in Table 5.17. For each test, a network of 50 nodes is used, 1000 queries are executed in each simulation run, and the simulation run is repeated twenty times; the median value of runs is reported as test results.

Figures 5.64, 5.65, 5.66 and 5.67 depict test results, and the following observation could be made regarding the results:

1) both ontology update time and ontology update rate parameters have effects on the TRQA algorithm behavior.

2) algorithm's worst behavior occurs when peers update their ontology frequently $t = 0$, and with highest update rate u=100% (see Figure 5.64).

3) The TRQA algorithm's best performance comes from moderate rate and time ontology updates. That is when u=50% and t=1000 or 1500, t $\geq \Delta_1$ (see figures 5.66 and 5.67).

These observations are consistent with the test results in sections 5.5.3.5.3 and 5.5.3.5.4. They again reinforce the fact that moderate ontology rate and time updates provide an opportunity for incorrect answers to be readjusted, hence, produce better results than extremely high/low ontology update rates and time.

Δ_1	Δ_2	u	t
1000	300	100%, 50%, 33%, 25%, 20%	0
			500
			1000
			1500

Table 5.17: Ontology update rate and time testing parameters

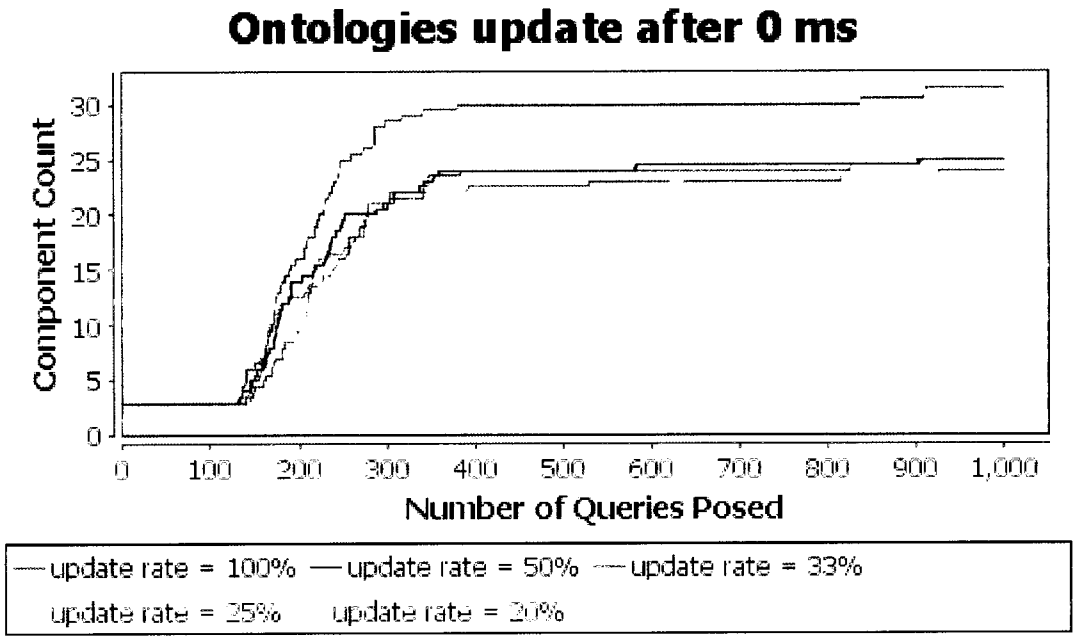

Figure 5.64: Ontology update rate's effect on the TRQA algorithm when update time =0

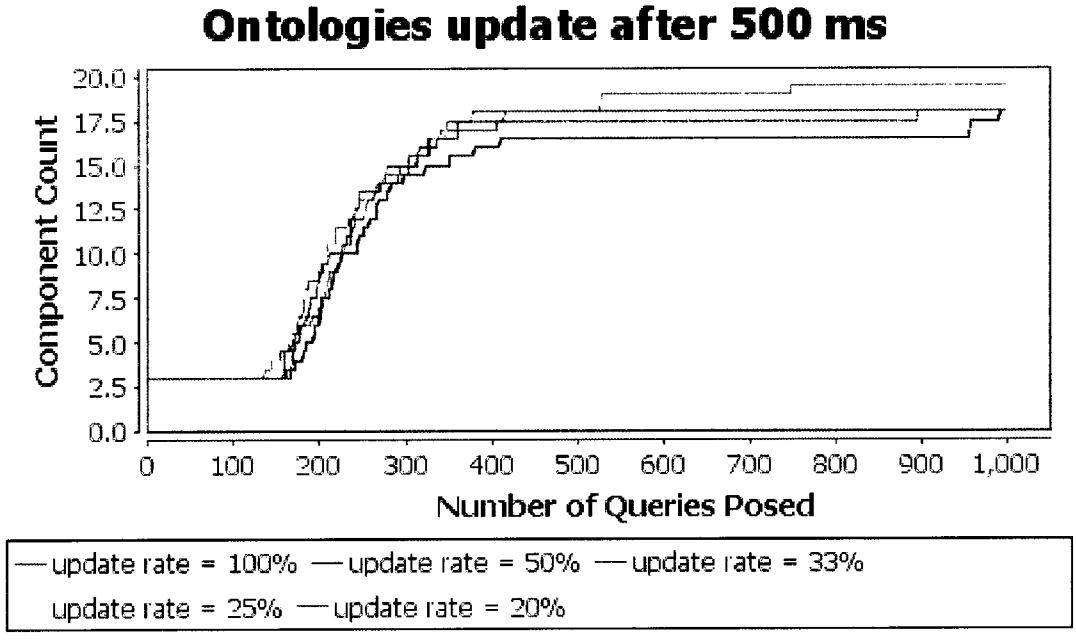

Figure 5.65: Ontology update rate's effect on the TRQA algorithm when update time =500

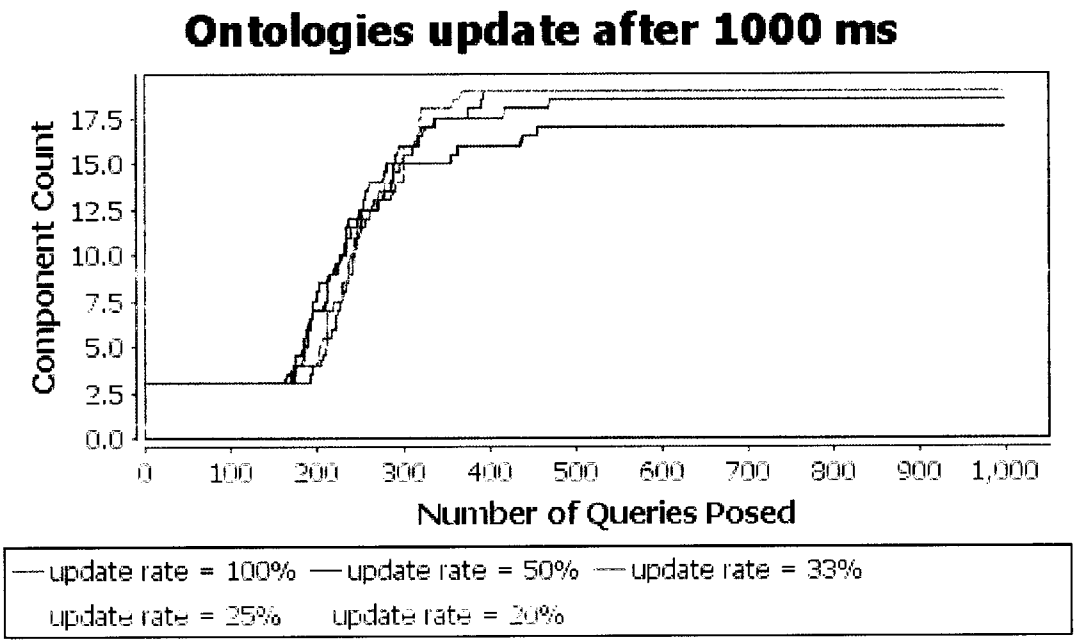

Figure 5.66: Ontology update rate's effect on the TRQA algorithm when update time =1000

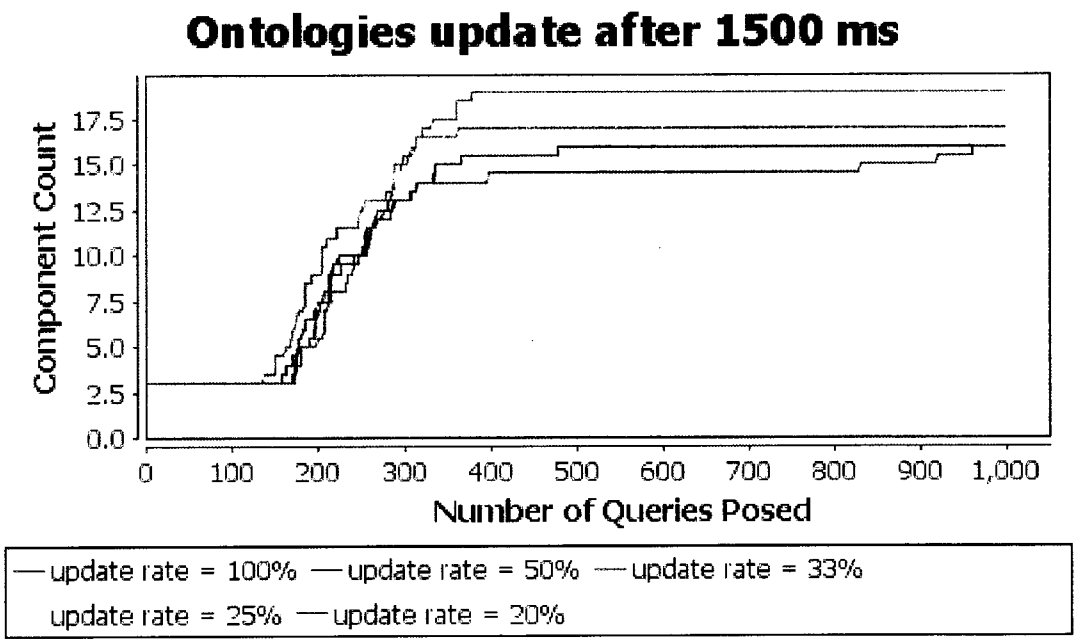

Figure 5.67: Ontology update rate's effect on the TRQA algorithm when update time =1500

5.5.3.6 Message complexity of TRQA algorithm

In this section we determine the message complexity (MC) of the TRQA algorithm. We quantify the complexity of the additional messages used by TRQA algorithm for maintaining relationships among SP2P system peers.

The message complexity of the TRQA is same as the message complexity of the GQA algorithm.

This is due to the following facts:

1) Both TRQA and GQA algorithms use similar approaches (additional queries) to maintain the relationships among peers in SP2P systems. See section 5.5.1 for the description of the similarities between the two algorithms.

2) The parameters that could be used to determine the message complexity of the TRQA algorithm are similar to the ones used by the GQA algorithm, and they are k, and TTL. The k parameter defines the query resubmission number, and the TTL parameter specifies the length of the path a query traverses during query forwarding. The traversal paths, named coverage $\vec{c}$, is defined in section 5.5.2.5.

We assume the k value to be low, e.g. 2 to 5. A low value assumption for k is reasonable because a larger value causes SP2P systems to spend a large amount of computational time on the query re-try, jeopardizing the usability of the entire system. Thus, the message complexity of the TRQA algorithm remains $\leq K\vec{C}$, and this gives us the $O(L)$ message complexity where L is the total number of links in the network.

5.5.3.7 Summary and conclusion

In this section, the time redundancy query answer (TRQA) algorithm, its steps and procedures along with the psuedocode, have been described. The algorithm was used to detect transient semantic mapping faults and, hence, maintain the semantic relationships among peers in the SP2P systems. The message complexity of the TRQA algorithm, and the differences and the similarities between the TRQA's fault-tolerant approach and the GQA's fault-tolerant approach were highlighted as well. A set of experimental results were carried out to determine the effectiveness of the

Δ_1	Δ_2	u	t	Δ_1 relevant to Δ_2	t relevant to Δ_2
1000	200	50%	200	$\Delta_1 < \Delta_2$	$t \geq \Delta_2$

Table 5.18: Parameters' values for the TRQA algorithm best result

algorithm and to study the impact of the algorithm's parameters on the algorithm's behavior.

Based on the test results we have arrived to the following conclusions: (1) The TRQA algorithm preserves the semantic relationships among peers in the SP2P systems. The network connectivity for the Reducible and Irreducible SP2P systems were improved by rates 20% and 27% respectively. (2) The low value of the wait function, Δ_2, gives a high chance for incorrect query results to be adjusted, and best results were obtained when the value of Δ_2 kept close to and less than the value of Δ_1. (3) High values of Δ_1 and Δ_2 prevent peers from reacting to incorrect answers on time. (4) The low value of t (e.g. $t = 200$), leads to a higher network connectivity degree than the case when ontologies change less frequently (e.g $t = 1000$). (5) As for the ontology update rate, the highest network connectivity is achieved when 50% of peers update their ontologies. Table 5.18 shows the parameters' values where the TRQA algorithm have the best result.

Furthermore, the ability to simulate the P2PSLN network has demonstrated that our reference model-based simulation can be used to imitate a wide range of existing SP2P systems.

Chapter 6

Conclusion and Future Work

This study identified that the diversity in implementation and architecture of SP2P systems cause ambiguity and incompatibility in defining domain abstracts and concepts. Progress in this area is hampered by a lack of commonality between these approaches, which makes their comparison and translation into practical implementations difficult. To address this shortcoming, a reference model was developed. The potential contributions of the reference model to the advancement of the current SP2P systems various areas.

An SP2P simulation framework, SP2P:sim, was derived from the reference model. The simulation framework is generic which enabled instantiating different existing SP2P systems, for example, Chatty Web, Piazza, and P2PSLN systems for studying various aspects of these systems.

Simulations built based on the SP2P:sim framework were used for studying various aspects of the existing SP2P systems. Using the generic framework and simulation models, we examined the effect of architecture changes, introducing different component implementations, on the existing systems and compared existing SP2P systems to one another. Furthermore, the simulations were used for studying the lack of a fault-tolerant problem (i.e reliability problem) of the current SP2P systems. To solve the reliability problem, the study developed two different solutions. The first solution, Generous Query Answerer algorithm (GQA), was inspired by the generosity tit-for-tat algorithm developed for game theory. The second solution, Time Redundant Query Answerer Algorithm (TRQA), was based on the time redundancy technique used in

the fault-tolerance discipline. Test results generated from running SP2P simulation with GQA, TRQA, and various system parameters provide system designers with valuable knowledge about SP2P system characteristics and behaviors. This information could in turn contribute in improving the current practice of SP2P systems.

Using simulation models, we were able to examine the impact of our developed algorithms on the reliability method of ChattyWeb [3], Piazza [49] and P2PSLN [48]. In Sections 5.5.2.3.3 and 5.5.2.3.4, we demonstrated that the reliability of Chatty Web could be improved by tolerating non-permanent faults. In Section 5.5.2.3.4, we identified how Piazza could benefit from incorporating the GQA algorithm. Using the TRQA algorithm (section 5.5.3), the promise of checking on recurrent incorrect query answers from semantically relevant peers made by P2PSLN [48] is fulfilled and the feasibility of building SP2P systems with potential for high reliability is demonstrated.

6.1 Future work

In this section, we list some of future research work.

1) Validating the reference model with a new set of SP2P systems for discovering new constructs or improving the currently identified SP2P constructs is important for future work. The reference model and its validation reveal the fact that different SP2P systems possess different features. Combining prominent features from different SP2P systems to come up with new systems is yet another viable future research direction.

2) The described SP2P:sim simulation could be improved in several ways. These include, improving query construction, using more involved local mapping procedure, and code modularity. Currently queries in the SP2P:sim are merely ontology concepts. This could be improved by using an actual query language for constructing queries. An example of such query language would be SPRQL[1]. Similarly, the SP2P:sim mapping component could be improved by incorporating a more involved mapping procedure, plugging existing mapping tools, into the simulation for mapping between query concept and peers' local ontologies, rather than

[1]http://www.w3.org/TR/rdf-sparql-query/

checking whether or not a query concept exist in the peer's local ontologies. The simulation code could be re-written to enhance the code modularity. This will enable reusing commercial off-the-shelf (COTS) components such as mapping and routing components. We believe that overcoming the identified limitations of the SP2P:sim brings the simulation closer to the actual systems. Thus, getting over these limitations is important for future research.

3) While we had implemented and tested the majority voting query answer (MVQA) algorithm, test results were not reported in this dissertation. This is because reported tests in sections 5.5.2 and 5.5.3 were sufficient for validating the research hypotheses. Nevertheless, complete testing of MVQA algorithm and reporting test results are important for future works.

4) The developed reference model is an essential step toward building a comprehensive API for SP2P systems. We consider building such an API to be important for future work.

5) We believe that studying the impact of permanent and non-permanent semantic mapping in research areas (such as semantic negotiation [39], semantic Web Services [108] and emergent semantics [3, 2, 60, 102] to be a desirable research direction.

Bibliography

[1] K. Aberer, L. O. Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, and S. Haridi. The essence of p2p: a reference architecture for overlay networks. In *Proceedings of the Fifth IEEE International Conference on P2P Computing*, pages 11-20, 2005.

[2] K. Aberer, T. Catarci, P. Cudré-Mauroux, T. Dillon, S. Grimm, M.-S. Hacid, A. Illarramendi, M. Jarrar, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, A. M. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, R. Studer, and O. De Troyer. Emergent semantics systems. In *Proceedings of the International Conference on Semantics of a Networked World (ICSNW)*, pages 14-43, 2004.

[3] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. Start making sense: the chatty web approach for global semantic agreements. *Journal of Web Semantics*, 1(1): 89-114, 2003.

[4] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. Van Pelt. GridVine: building internet-scale semantic overlay networks. In *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, pages 107-121, 2004.

[5] P. Adjiman, P. Chatalic, F. Goasdoué, M.-c. Rousset, and L. Simon. SomeWhere in the Semantic Web. In *Proceedings of the Third International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR06)*, pages 1-16, 2005.

[6] Y. An, A. Borgida, and J. Mylopulos. Discovery and maintaining semantic mappings between XML schemas and ontologies. *Journal of Computing Science and Engineering*, 2(1): 44-73, 2008.

[7] L. Anghel, D. Alexandrescu, and M. Nicolaidis. Evaluation of a soft error tolerance technique based on time and/or space redundancy. In *Proceedings of the Thirteenth Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 237-242, 2000.

[8] G. Antoniou and F. v. Harmelen. *A Semantic Web Primer.* The MIT Press, Cambridge, Massachusetts, 2004.

[9] R. Axelrod. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration.* Princeton University Press, Princeton, N.J. 1997.

[10] E. Bahceci, O. Soysal, and E. Sahin. A review: pattern formation, and adaptation in multi-robot systems. Technical Report Number CMU-RI-TR-03-43, Robotics Institute, Carnegie Mellon University, 2003.

[11] T. Berners-Lee and M. Fischetti. *Weaving The Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor.* Collins publishing, New York, 2000.

[12] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: a vision. In *Proceedings of Fifth International Workshop on the Web and Databases (WebDB)*, pages 88-94, 2002.

[13] D. Bianchini, V. De Antonellis, M. Melchiori, and D. Salvi. Peer-to-peer semantic-based web service discovery: state of the art. Technical Report, Dipartimento di Elettronica per l'Automazione Universit di Brescia, 2006.

[14] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni. Discriminating fault rate and persistency to improve fault treatment. In *Proceedings of Twenty-seventh Annual International Symposium on Fault-tolerant Computing (FTCS-27)*, pages 354-62, 1997.

[15] A. Bondavalli, F. D. Giandomenico, and F. Grandoni. Threshold-based mechanisms to discriminate transient from intermittent faults. *IEEE Transactions on Computers*, 49(3): 230-45, 2000.

[16] M. Bonifacio, P. Bouquet, G. Mameli, and M. Nori. Peer-mediated distributed knowledge management. In *Proceedings of International Symposium on Agent-Mediated Knowledge Management (AMKM-2003)*, pages 31-47, 2004.

[17] M. Bonifacio, P. Bouquet, G. Mameli, M. Nori. KEx: a peer-to-peer solution for distributed knowledge management. In *Proceedings of the Fourth International Conference on Practical Aspects of Knowledge Management*, pages 490-500, 2002.

[18] P. Bouquet, A. Donk, L. Scrafini, and S. Zanobini. ConTeXtualized Local Ontology Specification Via CTXML. Technical Report Number 0303-11, Istituto Trentino di Cultura, 2003.

[19] P. Bouquet, F. Giunchiglia, F. v. Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: contextualizing ontologies. In *Proceedings of the Second International Semantic Web Conference*, pages 164-179, 2003.

[20] S. Castano, A. Ferrara, and S. Montanelli. H-Match: an algorithm for dynamically matching ontologies in peer-based systems. In *Proceedings of the First VLDB International Workshop on Semantic Web and Databases (SWDB)*, pages 231-250, 2003.

[21] S. Castano and S. Montanelli. Enforcing a semantic routing mechanism based on peer context matching. In *Proceedings of the Second International Workshop on Contexts and Ontologies: Theory, Practice and Applications*, at *seventeenth European Conference on Artificial Intelligence (ECAI)*, pages 10-15, 2006.

[22] N. Choi, I. Song, H. Han. A survey on ontology mapping. *SIGMOD Record*, 35(3):34-41, 2006.

[23] D. Colazzo1, and C. Sartiani. Mapping maintenance in XML p2p databases. In *Proceedings of the Tenth International Symposium on Database Programming Languages (DBPL 2005)*, pages 7489, 2005.

[24] P. Cudré-Mauroux, K. Aberer, and A. Feher. Probabilistic message passing in peer data management systems. In *Proceedings of the Twenty-Second International Conference Data Engineering (ICDE '06)*, pages 41-41, 2006.

[25] F. Dabek, E. Brunskill, M. F. Kaashoek, and D. Karger. Building peer-to-peer systems with chord, a distributed lookup service. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HOTOS-VIII)*, pages 81-86, 2001.

[26] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica Towards a common API for structured peer-to-peer overlays. In *Proceedings of the Peer-to-Peer systems II, Second International Workshop (IPTPS 2003)*, pages 33-44, 2003.

[27] A. De Moor. Ontology-guided meaning negotiation in communities of practice. In *Proceedings of Workshop on the Design For Large-Scale Digital Communities at the Second International Conference on Communities and Technologies*, pages 21-28, 2005.

[28] A. De Moore, P. De Leenheer, and M. Meersman. DOGMA-MESS: A meaning evolution support system for inter organizational ontology Engineering. In *Proceedings of fortieth International Conference on conceptual structures*, pages 189-203, 2006.

[29] A. Doan and A. Halevy. Semantic integration research in the database community: a brief survey. *AI Magazine*, 26(1):83-94 2005.

[30] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *VLDB Journal*, 12(4):303-19, 2003.

[31] C. Doulkeridis, A. Vlachou, K. Nørvåg, Y. Kotidis, and M. Vazirgiannis. Efficient search based on content similarity over self-organizing p2p networks. *Peer-to-Peer Networking and Applications Journal*, 3(1):67-79, 2010.

[32] E. Dupont, M. Nicolaidis, and P. Rohr. Embedded robustness IPs for transient-error-free ICs. *IEEE Design & Test of Computers*, 19(3):56-70, 2002.

[33] M. Ehrig. *Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond)*. Springer publishing, New York, 2007.

[34] P. Fergus, A. Mingkhwan, M. Merabti, and M. Hanneghan. Distributed emergent semantics in p2p networks. In *Proceedings of the Second International Conference on Information and Knowledge Sharing (IASTED)*, pages 75-82, 2003.

[35] P. Fettke and P. Loos (eds.). *Reference Modeling for Business Systems Analysis*. Idea Group Publication, Hershey, PA, 2007.

[36] P. Fettke and P. Loos. Using Reference Models for Business Engineering - State-of-the-Art and Future Developments. In *Proceedings of the Innovations in Information Technology International Conference*, pages 1-5, 2006.

[37] E. Franconi, G. Kuper, Gabriel, A. Lopatenko, and I. Zaihrayeu. Queries and updates in the coDB peer to peer database system. In *Proceedings of the Thirstiest International Conference on Very Large Databases (VLDB04)*, pages 1277-1280, 2004.

[38] A. Gal. Semantic interoperability in Information services: experiencing with CoopWARE. *SIGMOD Record*, 28(1):68-75, 1999.

[39] S. Garruzzo and D. Rosaci. Ontology enrichment in multi agents systems through semantic negotiation. In *Proceedings of the OTM Confederated International Conference: CoopIS, DOA, ODBASE, GADA, and IS*, pages 391-398, 2007.

[40] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence Archive*, 127(2):221-259, 2001.

[41] R. J. Glushko and T. McGrath. *Document Engineering: Analyzing and Designing Documents for Business Informatics & Web services.* The MIT Press, Cambridge, Massachusetts, 2005.

[42] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering: with Examples From the Areas of Knowledge Management, E-commerce and the Semantic Web.* Springer publishing, New York, 2003.

[43] T. R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 601-602, 1991.

[44] N. Guarino. Formal ontology and information systems. In *Proceedings of the First Formal Ontology in Information Systems*, pages 3-18, 1998.

[45] C. Gutierrez, C. Hurtado, and A. Vaisman. Temporal RDF. In *Proceedings of the Second European Semantic Web Conference (ESWC)*, pages 93-107, 2005.

[46] P. Haase, J. Broekstra, M. Ehrig, M. Menken, P. Mika, M. Olko, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster – A Semantics-based bibliographic peer-to-peer System. In *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, pages 122-136, 2004.

[47] P. Haase, R. Siebes, and F. van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. In *M. Bouzeghoub, C. A. Goble, V. Kashyap, and Stefano Spaccapietra (Eds.)* LNCS 3226, pages 108-125, 2004.

[48] Z. Hai, L. Jie, L. Feng, X. Sun, and C. He. Query routing in a peer-to-peer semantic link network. *Computational Intelligence*, 21(2):197-216, 2005.

[49] A. Halevy, Z. Ives, P. Mork, and I. Tatarinov. Piazza: mediation and integration infrastructure for semantic web data. In *Proceedings of the Twelfth International World-Wide Web Conference*, pages 556-567, 2003.

[50] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the Nineteenth International Conference on Data Engineering*, pages 505-516, 2003.

[51] P. Hammerstein and E. H. Hagen. Robustness: a key to evolutionary design. *Biological Theory*, 1(1):9093, 2006.

[52] C. Hurtado and A. Vaisman. Reasoning with temporal constraints in RDF. *Principles and Practice of Semantic Web Reasoning: Fourth International Workshop, (PPSWR 2006), Revised Selected Papers (LNCS. 4187)*, pages 164-78, 2006.

[53] H. L. Johnson, K. B. Cohen, and L. Hunter. A Fault model for ontology mapping, alignment, and linking systems. In *online Proceedings of Pacific Symposium on Biocomputing (PSB) 2007.* available at http://psb.stanford.edu/psb-online/Proceedings/psb07/, accessed July 15, 2010.

[54] S. Joseph. Neurogrid: Semantically Routing Queries in peer-to-peer Networks. In *Web Engineering and Peer-to-Peer Computing*, E. Gregori, L. Cherkasova, G. Cugola, F. Panzieri, and G. P. Picco (Eds.), LNCS 2376, pages 202-214, 2002.

[55] N. Kandasam, J. P. Haes, and B. T. Murray. Tolerating transient faults in statically scheduled safety-critical embedded systems. In *Proceedings of the Eighteenth IEEE Symposium on Reliable Distributed Systems*, pages 212-221, 1999.

[56] K. Kang. *Feature-Oriented Domain Analysis.* Technical Report Number CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.

[57] A. Kementsietsidis, M. Arenas, and R. J. Miller. Managing data mappings in the hyperion project. In *Proceedings of the Nineteenth International Conference on Data Engineering (ICDE)*, pages 732-734, 2003.

[58] M. Klein, A. Kiryakov, D. Ognyanov, and D. Fensel. Finding and characterizing changes in ontologies In *Proceedings of the Twenty-First International Conference on Conceptual Modeling*, pages 79-89, 2002.

[59] L.W. Lacy. *OWL: Representing Information Using the Web Ontology Language.* Trafford publishing, Victoria, BC, 2005.

[60] K. Larry, J. Hanjo, and K. Wooju. Emergent semantics in knowledge sifter: an evolutionary search agent based on semantic web services. *Journal on Data Semantics*, VI, pages 187-209, 2006.

[61] K. Laskey, F. McCabe, J. Estefan, and M. McRae. Reference model for service oriented architecture. url: http://docs.oasis-open.org/soa-rm/v1.0. accessed July 12, 2010.

[62] L. Liu, J. Xu, D. Russell, and N. Antonopoulos. Self-organization of autonomous peers with human strategies. In *Proceedings of the Third International Conference on Internet and Web Applications and Services (ICIW 2008)*, pages 348-357, 2008.

[63] D. Llic and D. Trobitsyna. Formal development of software for tolerating transient faults. In *Proceedings of the Eleventh Pacific Rim International Symposium on Dependable Computing*, pages 140-150, 2005.

[64] A. Löser, S. Staab, and C. Tempich. Semantic social overlay networks. *IEEE Journal on Selected Areas in Communications*, 25(1): 5-14, 2007.

[65] M. R. Lyu. *Software Fault Tolerance.* Wiley Publishing, Chichester, New York 1995.

[66] Y. Ma, B. Jin, Y. Li and K. Wu. A timing analysis model for ontology evolutions based on distributed environments. In *Proceedings of the Eleventh Pacific-Asia Conference (PAKDD)*, pages 183-192, 2007.

[67] A.-R. Mawlood-Yunis, M. Weiss, and N. Santoro. Reference model for semantic peer-to-peer networks. In *Proceedings of the Fourth International MCETECH Conference on E-Technologies*, pages 313-394, 2009.

[68] A.-R. Mawlood-Yunis. Reliable peer-to-peer semantic knowledge sharing system. In *Proceedings of Third International Workshop on Reliability in Decentralized Distributed Systems(RDDS)*, pages 894-903, 2008.

[69] A.-R. Mawlood-Yunis. Fault-tolerant semantic mappings among heterogeneous and distributed local ontologies. In *Proceedings of the Second International Workshop on Ontologies and Information Systems for the Semantic Web (ONISW)*, pages 31-38, 2008.

[70] A.-R. Mawlood-Yunis, M. Weiss, and N. Santoro. Fault-Tolerant Emergent Semantics in p2p Networks. In *Cardoso, J., and Lytras, M. (eds.), Semantic Web Engineering in the Knowledge Society*, pages 161-187, IGI Global, 2008.

[71] A-R. Mawlood-Yunis, M. Weiss, and N. Santoro. Fault classification in p2p semantic mapping. In *Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa)* at *International Conference on Artificial Intelligence (IJCAI)*, pages 103-111, 2007.

[72] A-R. Mawlood-Yunis, M. Weiss, and N. Santoro. Issues for robust consensus building in p2p networks. In *International Workshop on Ontology Content and Evaluation in Enterprise (OnToContent)*, pages 1020-1028, 2006.

[73] R. McCann, B. AlShebli, Q. Le, H. Nguyen, L. Vu, and A. Doan. Mapping maintenance for data integration systems. In *Proceedings of the Thirty-First International Conference on VLDB*, pages 1018-1029, 2005

[74] P. McDougall. The power of peer-to-peer. P. McDougall. In *Information week*, url: http://www.informationweek.com/801/peer.htm, accessed July 12, 2010.

[75] D. L. McGuinees. Ontologies come of age. In *D. Fensel, J. Hendler, H. Lieberman and W. Wahlster (eds.), Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential*, The MIT Press, 2003.

[76] E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. 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, 2000.

[77] E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Imprecise answers in distributed environments: estimation of information loss for multi-ontology based query processing. *International Journal of Cooperative Information Systems*, 9(4):403-25, 2000.

[78] P. Mika. Ontologies are us: a unified model of social networks and semantics. In *Proceedings of the Fourth International Semantic Web Conference (ISWC 2005)*, pages 522-36, 2005.

[79] C. F. Naiman and A. M. Ouskel. A classification of semantic conflicts in heterogeneous database systems. *Journal of Organizational Computing*, 5(2):167-193, 1995.

[80] C. Namyounc, S. Il-Yeol, and H. Hyoil. A survey on ontology mapping. In *ACM SIGMOD*, 35(3):34-41, 2006.

[81] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, T. Risch. EDUTELLA: a P2P networking infrastructure based on RDF. In *Proceedings of the Eleventh International Conference on World Wide Web*, pages 604-615, 2002.

[82] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. PeerDB: a p2p-based system for distributed data sharing. In *Proceedings of the Nineteenth International Conference on Data Engineering*, pages 633-644, 2003.

[83] M. Nilsson. The Edutella p2p network - supporting democratic E-learning and communities of practice. In *R. McGreal (ed.) Online Education Using Learning Objects*, RoutledgeFalmer publishing, pages 244-254, 2002.

[84] M. J. North, N. T. Collier, and J. R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16(1):1-25, 2006.

[85] N. F. Noy and M. Klein. Ontology evolution: not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428-440, 2004.

[86] N. F. Noy and M. A. Musen. Ontology versioning in an ontology management framework. *IEEE Intelligent Systems*, 19(4):6-13, 2004.

[87] A. Oram (ed). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies.* O'Reilly and Associates Inc publishing, Cambridge, Massachusetts, 2001.

[88] A. M. Ouksel. Ontologies are not the panacea in data integration: a flexible coordinator to mediate context construction. *Distributed and Parallel Databases*, 15(1):7-35,1999.

[89] E. Papalilo, T. Friese, M. Smith, and B. Freisleben. Trust shaping: adapting trust establishment and management to application requirements in a service-oriented grid environment. In *Proceedings of the Fourth International Conference on Grid and Cooperative Computing (GCC)*, pages 47-58, 2005.

[90] D. K. Paradhan. *Fault-tolerant Computer System Design.* Prentice-Hall PTR publication, Upper Saddle River, N.J., 1996.

[91] M. Parashar, S. Member, and J. C. Browns. Conceptual and implementation models for the grid. *Proceedings of the IEEE Journal*, 93(3):653-668, 2005.

[92] M. Pizza, L . Strigini, A. Bondavalli, and F. D. Giandomenico. Optimal discrimination between transient and permanent faults. In *Third IEEE International High-Assurance Systems Engineering Symposium* , pages 214-23, 1998.

[93] L. L. Pullum. *Software Fault Tolerance Techniques and Implementation.* Artech House Publishing, 2001.

[94] S. Ram and J. Park. Semantic conflict resolution ontology (SCROL): an ontology for detecting and resolving data and schema-level semantic conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):189-202, 2004.

[95] J. Risson and T. Moors. *Survey of Research Towards Robust Peer-to-Peer Networks: Search Methods.* Technical Report Number UNSW-EEp2p-1-1, University of New South Wales, Sydney, Australia, 2004.

[96] J. F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383-393, 1995.

[97] M.-C. Rousset Small can be beautiful in the semantic web. In *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, pages 6-16, 2004.

[98] M.-C. Rousset, P. Chatalic, P. Adjiman, P. Chatalic, F. Goasdoue, and L. Simon. SomeWhere: A scalable p2p infrastructure for querying distributed ontologies. In *Proceedings of the Fifth International Conference on Ontologies Databases and Applications of Semantics*, pages 698-703, 2006.

[99] A. Rowstron, P. Druschel, Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the Middleware 2001: IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329-350, 2001.

[100] B. Sainty. Achieving greater cooperation in a noisy prisoner's dilemma: an experimental investigation. *Journal of Economic Behavior and Organization*, 39(4):421-435, 1999.

[101] C. Schmitz and A. Löser. How to model semantic peer-to-peer Overlays? In *Proceedings of the GI Jahrestagung Workshop (Informmatik 2006)*, pages 1-8, 2006.

[102] S. Staab. Emergent semantics. *IEEE Intelligent Systems*, 17(1):78-86, 2002.

[103] S. Staab. Social networks applied. *IEEE Intelligent Systems*, 20(1):80-93, 2005.

[104] S. Staab and S. Stuckenschmidt (eds.). *Semantic Web and Peer-to-Peer: Decentralized Management and Exchange of Knowledge and Information.* Springer publishing, New York 2006.

[105] L. M. Stephens, and M. N. Huhns. Consensus ontologies: reconciling the semantics of web pages and agents. *IEEE Internet Computing*, 5(5): 92-95, 2001.

[106] L. Stojanovic, A. Maedche, N. Stojanovic and, R. Studer. Ontology evolution as reconfiguration-design problem solving. In *Proceedings of the Second International Conference on Knowledge Capture*, pages 162-171, 2003

[107] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: principle and methods. *Data and Knowledge Engineering*, 25(1-2):161-197, 1998.

[108] R. Studer, S. Grimm, and A. Abecker (eds.). *Semantic Web Services.* Springer publishing, Heidelberg, Berlin, 2007.

[109] C. Tempich, H. S. Pinto, Y. Sure, and S. Staab. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (DILIGENT). In *Proceedings of the Second European Semantic Web Conference (ESWC)*, pages 241-56, 2005.

[110] C. Tempich, S. Staab, and A. Wranik. REMINDIN': semantic query routing in peer-to-peer networks based on social metaphors. In *Proceedings of the Thirteenth International Conference on the World Wide Web*, pages 640-649, 2004.

[111] H. Volker and S.-V. Katarina. Towards a reference model for grassroots enterprise mashup environments. In *Proceedings of Seventeenth European Conference on Information Systems (ECIS2009)*, 2009.

[112] S. Voulgaris, A. Kermarrec, L. Massoulié, and M. V. Steen. Exploring semantic proximity in peer-to-peer content search. In *Tenth International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, pages 238-243, 2004.

[113] A. B. Williams, A. Padmanabhan, and M. B. Blake. Experimentation with local consensus ontologies with implications for automated service composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):969-981, 2005.

[114] H. F. Witschel. Ranking information resources in peer-to-peer text retrieval: an experimental study. In *Proceedings of the Sixth Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR'08)*, pages 75-82, 2008.

[115] J. Wu and R. Axelrod. How to cope with noise in the iterated prisoner's dilemma. *Journal of Conflict Resolution*, 39(1):183-189, 1995.

[116] C. Yu and L. Popa. Semantic adaptation of schema mappings when schemas evolve. In *Proceedings of the Thirty-First International Conference on VLDB*, pages 1006-1017, 2005

[117] I. Zaihrayeu *Towards Peer-to-Peer Information Management Systems.* PhD Dissertation, International Doctorate School in Information and Communication Technologies, DIT - University of Trento, 2006.

[118] H. Zhu, S. E. Madnick, and M. D. Siegel. Effective data integration in the presence of temporal semantic conflicts. In *Proceedings of the Eleventh International Symposium on Temporal Representation and Reasoning*, pages 109-114, 2004.

[119] H. Zhuge, Jie Liu, L. Feng, and C. He. Semantic-based query routing and heterogeneous data integration in peer-to-peer semantic link networks. In *Proceedings of the First International IFIP Conference on Semantics of a Networked World (ICSNW 2004)*, pages 91-107, 2004.