International Journal of Soft Computing 2 (1): 48-60, 2007
© Medwell Journals, 2007

Building OWL Ontology to Enhance the Peer Profile in Platforms Integration Case

Younes Djaghloul and Zizette Boufaida
Lire laboratory, University Mentouri of Constantine, 25000, Algeria

Abstract: In this study, we present our approach to integrate the peer-to-peer platforms. Our approach is based on a semantic peer profile and the roaming service. The semantic profile is considered as a formal ontology that puts a clear separation between the platform specific characteristics and the logical ones. In order to enable the navigation of the peer across platforms, a roaming service is added. The latter uses the proposed ontology and deals with the platform specifications to ensure unique vision of the peer. The ontology is created according to a specific development process that produces a validate OWL document; so, the structure of the system is considered as a web resource.

Key words: OWL ontology, peer to peer, platforms integration

INTRODUCTION

Peer to peer is becoming one of the most important areas in computer science, especially in the Internet topic. This is due to the nature of P2P model that have several advantages as self-organization, load balancing, adaptation and fault tolerance. The P2P system is characterized by the autonomy of each peer and a high degree of decentralization. Each peer, acts as client and as server in the same time, so it demands and provides services without using the same typology of a client/Server model.

In P2P systems, peers (nodes) are working to achieve specific needs. The needs can be classified into several areas as: file sharing (music or other), distributed computing, distributed storage, communication…

In practice, one can classify P2P systems in two main architectures: unstructured and structured approaches. Unstructured P2P approach are loosely controlled, there is not special peer to control the execution of query or to maintain a global repository over others peers. This permits to have a high dynamic behavior of the system. So, if a peer joins or leaves the overly network, there are no added tasks to do.

All peers have equal role. Gnutella[1-3], KaZaa and Freenet belong to this type.

In the study of the structured P2P system, the Distributed Hash Tables play a key role to determine exactly the location of the peer and its resources. Many systems are proposed as: Chord[4,5], CAN[1]. In these systems, a hash function is used to obtain a valid key.

On the other hand, the semantic web is considered as the new vision to the web that try to give semantic to the web resources. Since it is based on a logical foundation, the web semantic and its language OWL provide a good solution to semantic problems and ontology presentation.

One remarks that several P2P systems are proposed; each system is based on a specific platform and appropriate technique. In these systems, a peer is presented with a specific structure and it is influenced by the technical aspect of the node. In our study, we aim to provide a new capability to peer that is the roaming service. As in mobile telephony, the peer can navigate across P2P systems and it maintains a unique profile to enhance the result's quality. The unified profile is assured by our proposed ontology. Special wrappers permit the mapping task between the global profile and the specific ones. In addition, the use of OWL language to describe the structure of the P2P system permits to export the P2P system itself as web semantic resource. So, the structure can be used by others with strength semantic.

PEER TO PEER SYSTEMS AND WEB SEMANTIC

The peer to peer paradigm allows to give the capabilities of client server paradigm to each node of the network

Corresponding Author: Younes Djaghloul, Lire laboratory, University Mentouri of Constantine, 25000, Algeria

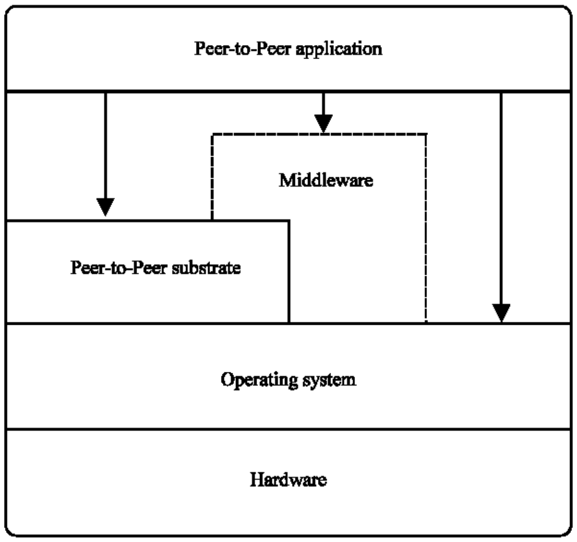

Fig. 1: A simplified model of peer application.

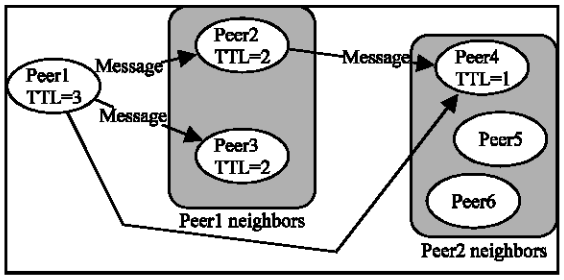

Fig. 2: Message sending in unstructured P2P system.

In this study, one will give an overview of the Peer-To-Peer paradigm and the web semantic technology.

Peer-To-Peer characteristics: The P2P is characterized by:

- Autonomy of peers. Nodes participating in a P2P system operate autonomic entities. Each peer can decide to leave the system, forward or performs a query.
- Ad hoc nature. There is no control of any node. This characteristic may poses a big problem in the organization of the whole system. So, a mechanism is incorporated to ensure a self-organization.
- Fault tolerance. In P2P, the system do not crashes if a number of peers cannot provide services (ex. leave the system).

Peer-To-Peer substrates: As shown in Fig. 1, the substrate is a key component in any peer architecture. It provides a mechanism for managing peers and resources.

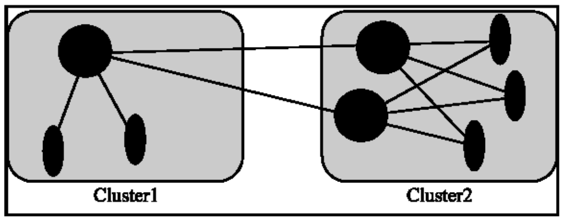

Fig. 3: A supper-peer approach.

For peers, it permits joining and leaving services. For resources, it permits the storing and the locating of every resource in the node.

P2P systems are divides according to their P2P substrate mechanisms. Therefore, one can divide the P2P substrates into two mains categories: unstructured P2P and structured ones.

Unstructured P2P substrates are loosly-controled. There is not a special peer to organize or to maintain the system, the system is organized as follow.

Two peers that maintain a connection are called neighbors. A peer can have more than one neighbor, the number of these neighbors is called outdegrees. In an unstructured P2P system, messages (queries or results) are only routed along an open connection. Therefore, if there is not an open connection between two peers. The message must use a path and pass along other peers that are considered as bridge. The length of path is called hop.

At the peer level, if a user sends a query, the peer becomes a source that sends the query to all its neighbors. Other techniques are proposed to reduce the number of the sent message. They consist to send to a set of neighbors rather than to all ones. Each message is accompanied with a time to leave (TTL) values that specify the number of hop. This number decrease on each peer and the sending ends when the TTL =0.

In Fig. 2, Peer 1 has two neighbors peers (Peer2, peer3). Peer1 send a message to its neighbors, the latter decrease the TTL value, peer2 do the same task Fig. 2.

In the follow, we present some important systems in the unstructured peer category.

Gnutella[1] is considered as one of the most widely deployed file sharing system. The Gnutella peer is called servant (server/client). As presented before, the peer uses flooding technique to propagate a message, the flooding is controlled by a TTL value whose default value is 7.

Gnutella system provides a high degree of reliability. However, since the peers have equal responsibility, there is no distinguishing between the peers according to their bandwidth or performance. In addition, the use of flooding technique with all the neighbors consumes the bandwidth without guaranty the quality of results.

In order to solve this problem, solutions are proposed. In[6], algorithms to reduce the bandwidth consummation by sending messages to a set of neighbors are proposed. An other solution consists of the introduction of the Super-Peer approach[7].

In a super-peer system[7], the network is divided into groups, each one contains one special peer called a super-peer. All the peers in the group are connected to the super-peer and the latter is connected to other supper peers of other groups. In the terminology of the supper-peer approach, the group (supper-peer and its peers) is called cluster. The main problem in the supper peer approach is the risk for the dysfunction of the supper peer. If the latter is disconnected, the entire cluster is disconnected. In order to resolve the problem a supper-peer redundancy is proposed. Fig. 3 shows two clusters, the first with one supper peer and the second with two supper-peers. The connection between the two clusters is assured by the supper peers.

In *Structured P2P substrates* the placement of the resource and the peer is tightly associated to the network structure. The substrate component uses a Distributed Hash Table (DHT) to locate the peers and the resources. The DHT technique guarantees the locating the file or the peer if they are connected.

DHTs organize the system in different manners. One can uses different geometrical shapes as tree, ring, hypercub. The DHTs are strong technique to lookup for peers resources. However, it suffers of some problems like

- DHTs support an exact match query, so, a little change in one characteristics of the resource causes the fail of the lookup.
- The peers can join or leave much time in a few times, this needs more operations for the management of the DHT.

Many works belong to this category; in (Content Addressable network) CAN the network is considered as virtual d-dimensional Cartesian coordinate. CAN uses dynamic partitions, each ones contains a set of peer. Chord associates to nodes an m-bit identifier using a special hash function as SHA-1. The identifiers values are ordered in a circle manner. Each node maintains a local routed table called the finger table. The system uses the IP addresses and port number to determine the physical location of the node.

Semantic web: The semantic web is a new technology that intends to make web resources more readily from users and machines. The semantic markup extensions permit to obtain this goal. These markup extensions are considered as meta-data annotations to describe their contents.

In the area of the web semantic, ontologies play a key role to describe information sources and permit an efficient share of meta-data. This need prompts the development of several languages as OIL, DAML+OIL and OWL[8]. The OWL language uses the capabilities of RDFs language as: declaration of classes, organization of classes in subsumption hierarchy. Moreover, OWL gives other extensions for RDFs as: putting that a property is transitive, symmetric, or it is an inverse of another property.

Theses relations permit to obtain a clear and formal semantic of the proposed ontology.

In our work, we aim to use the web semantic to export the P2P system as a web resource.

RELATED WORKS

Various research projects address the use of semantic in P2P system. Edutela[9] uses the sun JXTA platform to exchange learning resources. P-Grid[10] provides a strong self-organization service in a high-decentralized system. It is based on a virtual distributed search tree. SWAP[11] combines P2P and web semantic. RDFPeer[12] builds the Multi-Attribute Addressable Network (MAAN) that extends Chord.

When analyzing the P2P systems, one remarks that several architecture and technique are proposed to assure the lookup service and the organization task. For the lookup service, techniques as DHT (in structured P2P) provide identifications for the peer and the resources. However, this identification differs according to the used system; one may have the same peer with the same resources but the identification changes.

So, A peer p1 can be identified in P-Grid with ID = “pgrids://01001101” and the same peer can be reference in JXTA with ID = "jxta:uuid-150325033CD144F82DED74E". Furthermore, a peer must belong to a unique P2P system in order to be referenced. So the peer cannot navigate between P2P systems because of the characteristics of each system. In addition, each system uses a specific platform and peers are relied to the specific characteristic of its platform.

Furthermore, one can present other problems and limitation of the actual architectures, as:

- The absence of a clear and formal presentation of the Peer structure.
- The impossibility to navigate between different Peer platforms.
- The heterogeneity of techniques used in the different platforms.

- The loose of the historical activity of peer between the different sessions of the user. The historic permits to improve the quality of the future navigations based on the old experiences.

In order to enhance the capability of a P2P system and to resolve theses problems, we propose a new P2P design that has as objectives:

- The proposition of a formal peer profile.
- The separation between the technical aspect of a peer and the logical ones.
- The proposition of a unique identification of peer unless they use different platforms.
- The possibility of the peer to navigate between P2P systems.
- The use of the ontology paradigm (based on Description logic and OWL DL language). This permits to export peer structure as web semantic resource, in order to unify the notation and enhance the interoperability between the different systems.

This study augments our previous work[13]. It enhances the proposed ontology and describes the roaming service components.

Overview of the proposed approach : Our study is based on a clear separation between the technical aspect and the logical one of the peer. In order to ensure this independency, each peer has a unique identification that can be mapped into specific one, according to the platform or the used system. In order to provide a unique view of the peer a clear description of the peer structure is needed. The description must be independent from the technical aspect of the used platform. In this study; the ontology and the OWL language provide a good solution to express the semantic of the peer structure. The OWL language proposes some semantic relations that are needed in our work as restriction, transitive, symmetric, transitive...

However, the ontology is not enough to ensure the navigation and the persistence of the peer profile. Because of its nature that is an XML document without any active capability. Hence, In order to use this ontology, we add a new layer with specific components. The latter are used to manage the ontology and to deal with the technical aspect of the chosen platform.

The proposed architecture contains tow layers: (1) the Platform layer contains the technical aspect of the platform (2) the Roaming Layer proposes the logical view of the peer and contains the components needed to assure this activity.

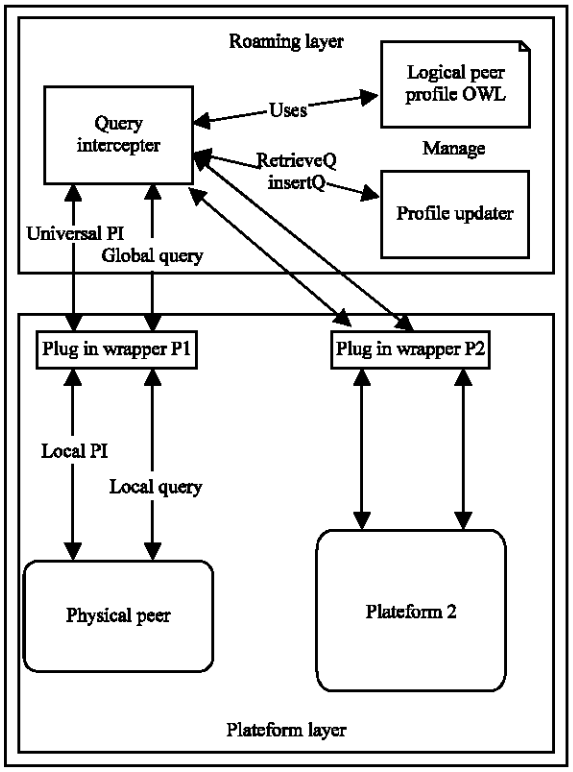

Fig. 4: Global architecture of the roaming service

Figure 4, shows the global architecture of our approach.

Peer roaming Layer provides new components that permit:

- To navigate easily across a P2P system like in the mobile telephony. Therefore, the peer can participate to several P2P systems and it keeps a unique identification in its navigation.
- To export the meta-data to several systems at the same time.
- To manage the logical peer profile.

The roaming service is used within other components (cf Fig. 1).

The peer's structure in the platform layer is called "physical". However, it is called "logical" in the roaming one.

The roaming layer provides the following services:

- Interception of the Join Query. This query is sent by other peers to join each platform.
- Interception of the AddRessource query. The latter is done by the user in order to publish some resources to share.

- Establishing a connection between the logical peer and the physical one. This is done by mapping of the logical Peer ID and the logical query with the local ones.
- Storing the profile of the peer as OWL document. This permits to use it with other platforms or with the same platform in different sessions. The OWL document is considered as logical profile and permits to maintain the history of the peer.

For each platform, we associate a special wrapper. The latter uses a correspondence table to map from the logical to the physical representation of the peer.

The platform layer contains the specific techniques to manage the peer system. The characteristics of each platform are conserved so long as possible.

The proposed ontology: In order to build ontology, one must use an appropriate methodology or processes. In our study, we use an Ontology Development Process for the Semantic Web[14]. In this study, we follow five phases that are; needs specification, conceptualization, formalization, codification and finally verification. Fig 5 shows the different steps of the process.

In the following, we will apply the process to develop our ontology.

The Conceptualization step consists of identifying concepts and theirs relationships. In this step, we should create the following representations:

- The glossary of terms that contains all the usable terms in the domain.
- The concepts taxonomies that permit to organize the domain concepts and to propose suitable hierarchies of the domain knowledge.
- The binary relation diagram that establishes a relation between two concepts.
- The concept dictionary that contains all the concepts of the domain.
- A table of binary relations that presents all the binary relations between concepts. In this table, one must specify the name of the relation, the name of the source, the target concepts, the source and the target cardinality and the inverse role.

The conceptualization step: In the following, we will present the different representations of these tables. We begin with the glossary of terms.

Peer community: permits to organize the peers into groups of interest. The main goal of this organization is to facilitate the search process and to enhance the quality of the results.

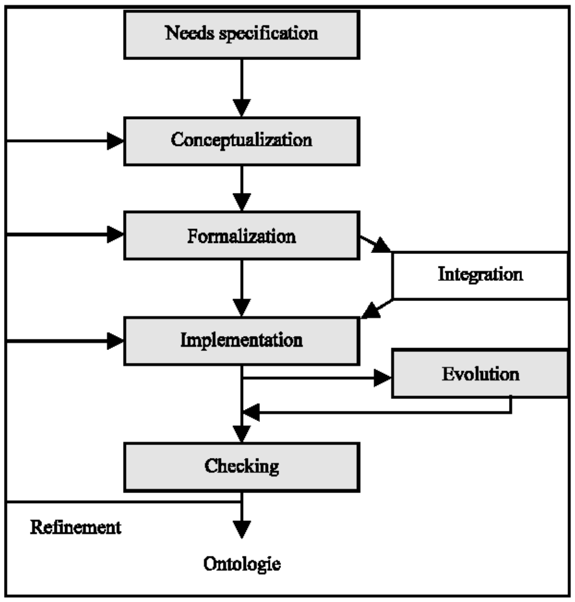

Fig. 5: Ontology development process.

Peer com ID: represents an identification of the community.

Peer com designation: Is the designation and the description that helps the user in his choice.

Peer com domain: indicates the domain and the global interest of the Peer community. It plays an important role to facilitate the search process. For example, if the user query is to find some document on semantic and the interest of this user is computer science, the finale results will be different with the some query on other community with different Peer Com Domain as "philosophy" Fig. 6.

Peer cluster: is another solution to organize the peer system, but not with the some goal of the PeerCommunity. Indeed, if the PeerCommunity aims to organize the peer system according to their interests, the main idea with the PeerCluster is to split the peer community into small groups. This division will reduce the number of query between the peers. Therefore, as presented in Fig. 3, only the SupperPeer sends and receives query between clusters. The query of the other peers of the some cluster must be pass by the Supper Peer.

Number of redundancy: Determines the number of supper peer in the cluster that ensures the redundancy if the supper peer is not available.

Cluster up bandwith: And ClusterDownBandwit determines the total bandwidth of the cluster in the uploading and downloading activities Fig. 7.

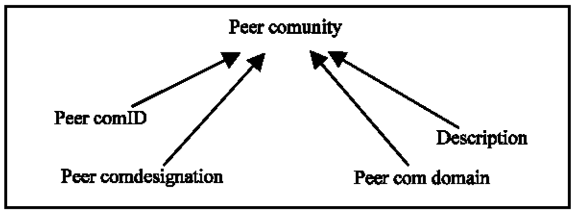

Fig. 6: The peer community hierarchy

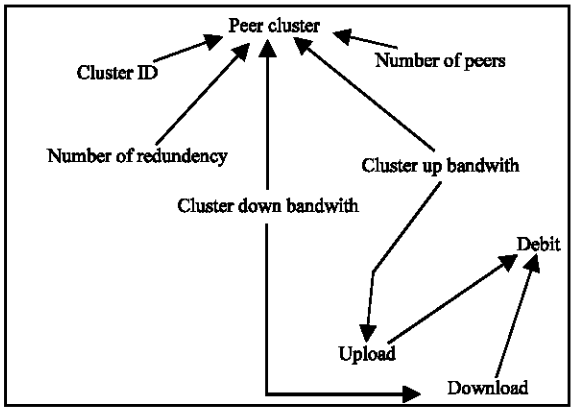

Fig. 7: The peer cluster hierarchy

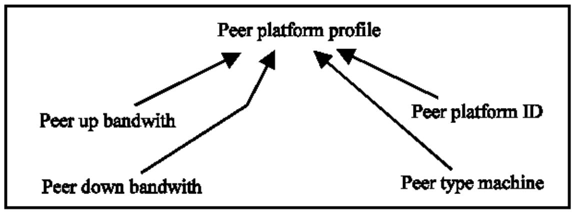

Fig. 8: The peer platform profile hierarchy

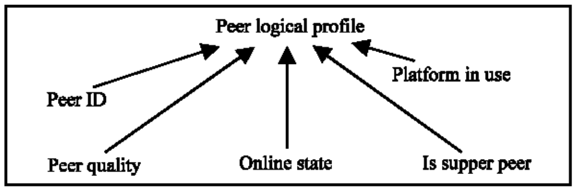

Fig. 9: The peer logical profile hierarchy

Peer profile: Represents a set of information about the profile of the peer

Peer logical profile: Represents a set of information about the logical aspects of the peer

The online state: Indicates if the peer is on line or not

Platform in use: Indicates the platform that is in use by the peer in the current session. According to its value, the roaming service can establish a correct connection with the wrapper in order to accomplish the different mapping between the roaming layer and the platform one.

Is supper peer: Indicates if the current peer is considered as a supper one in its cluster or no. If it is the study, it will be used as a proxy and all the other peers in the some cluster will pass their queries by it.

Peer ID: Each peer has a unique PeerID, The latter guarantees the identity of the peer between its different session in the some platform or in different ones.

In order to ensure that, we propose the use of a centralize mechanism as in mobile telephony. So, each user obtains his PeerID after a special query in which a special peer verifies the value of the PeerID. A user can request a special value of its PeerID if it is available.

Peer quality: Is used as information to classify the results of a query. In a resource sharing system, or in a simple search engine over a peer-to-peer system, one can have several results for the some query. In order to help the user to choose from these results, the latter are classified according to "PeerQuality". The value of the PeerQuality is calculated according to a special model of trust.

Peer platform profile: Represents a set of information about the profile of the peer in the platform in use. Unlike the "PeerLogicalProfile", this profile changes according to the platform in use and the current session.

Peer platform ID: Represents the code of the platform in use. We propose the use of the name of the platform in addition of the version. Ex JXTA 1.0, P-Grid...

Peer up bandwith: And PeerDownBandwithIndicates respectively the upload and the download capability of the peer. This is determined by the device used and the network configuration and restriction. Ex, in current Internet connections with ADSL support, one can have until 24 Mb in download and more than 8Mb for upload. The values are more interest in a LAN with a Gigabit connection.

Peer type machine: Represents the type of the machine used by the user. The type my be a PC, mobile phone, Pocket Pc or any other types.

Peer query: Represents the query sent by the Peer in its activities.

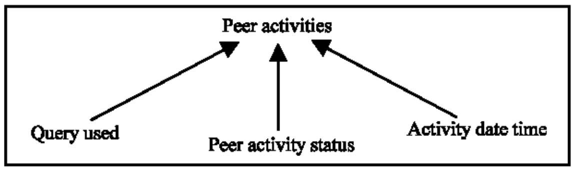

Fig. 10: The peer activities hierarchy

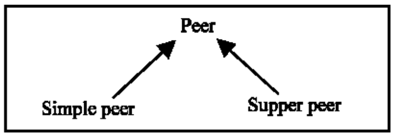

Fig. 11: The peer hierarchy

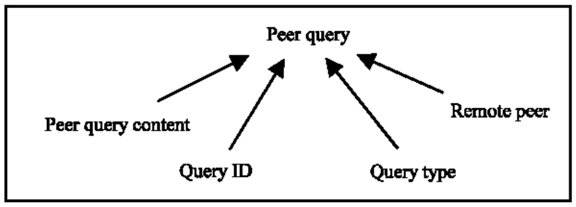

Fig. 12: The PeerQuery Hierarchy

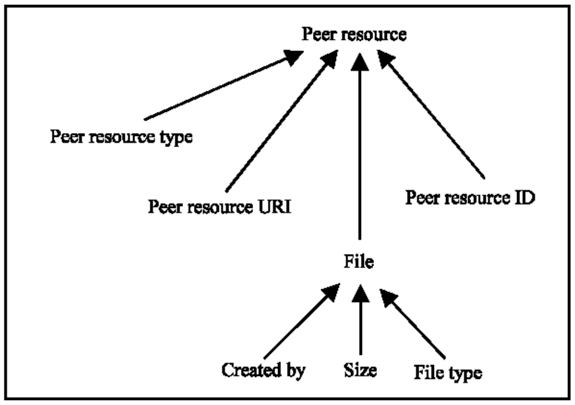

Fig. 13: The peer resource hierarchy

Query ID: Each query is identified by a Query ID.

Peer query content: Represents the syntax of the query that includes the requested resources.

Query type: Indicates the type of query can be a request for resource, insert, connect…

Peer ressource: Represents the resource that the peer hopes to share.

Peer resource ID: Each resource has a unique ID

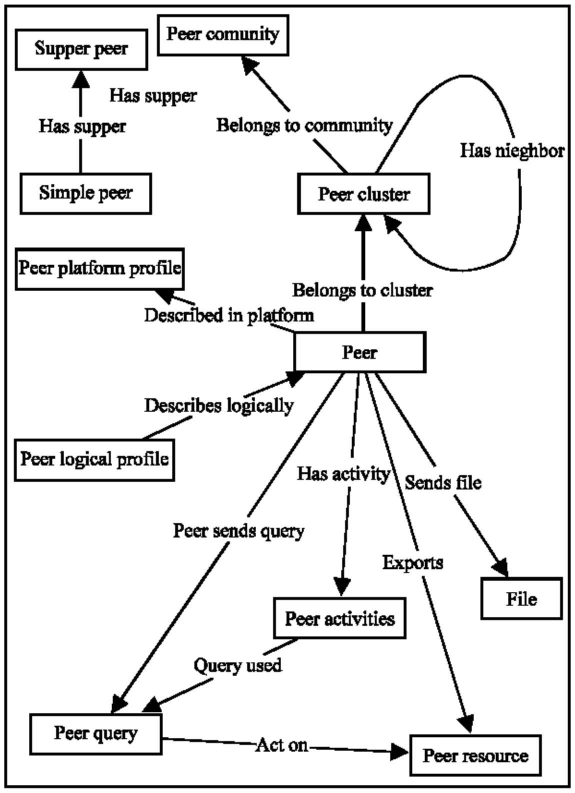

Fig. 14: The Binary relations diagram

Peer resource type: The type of the resource can be a simple file or records of Databases.

Peer resource URI: The Unique resource Locator of each resource permits to access directly the resource in the peer.

Peer logaical profile: Th peer logaical profile hirarcy shown in Fig. 9.

Peer activities: Permits to save the historical activity of the peer.

Activity date time: Indicates the time and the date of the activity Fig. 10.

Activity peer resource: Indicates the resource used in this activity.

File is a resource: In the peer–to-peer sharing system, the main resource is a file.

Created by: Indicates the creator of the file Size The size of the file

File type: The type of the file, It is determined by its extension. "PDF, DOC, AVI, MP3…

Up load: Indicate in Mb, the capacity of uploading.

Download: Indicates in Mb, the capacity downloading.

Debit: Represents the uploading and the downloading capacities.

Supper peer: is a peer, with the capacity of receiving the query from the simple peers and it deals with the others supperPeers.

Simple peer: is a peer that sends the query to the supper one Fig. 11 and 14.

On each attribute, we can define a restriction on its type or the values that can obtain. The restrictions are part of the OWL language and are used according to its syntax.

Ex: the concept "PeerCluster" contains an attribute "Number of redundancy". The value of the latter is between 0 and 3. So, by using

<owl:Restriction>

<owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >3</owl:maxCardinality>

<owl:onProperty> <owl:DatatypeProperty rdf:ID="NumberOfRedundancy"/></owl:onProperty>

</owl:Restriction>

An other example of a restriction is the fact that we indicate that some values are allowed for an attribute. The concept "Peer Platform Profile" contains an attribute "Peer Type Machine". We determine that a set of values are allowed (PC; PocketPc…). With OWL; we use

"<owl:one of>.

According to[14], a binary relation Table that contains the relations between the concepts is created. It permits to indicate the Source Cardinalities (SC), the Target Cardinalities (TC), the name of the relations and the inverse ones. In addition, we create an attributes table to describe each attribute, logical axioms table and instances table. Table 2 represents the relation table.

The formalization step: In this step of the ontology building process, we present the inclusion of concepts, the definition of roles and the ontology formalism according to SHIQ DL.

In Fig. 16, we present for each role its definition and its inverse role.

Table 1: Represents the concept dictionary

Concept name	Attributes
Peer Community	-Peer com ID
	-Peer comdesignation
	-Peer com domaine
	-description
Peer cluster	-Cluster ID
	-number of Peers
	-number of redundancy
	-Cluster up bandwith
	-Cluster down bandwith
Peer profile	-Peer logical profile
	-Peer platforme profile
Peer logical profile	-Peer ID
	-online state
	-Platform in use
	-Peer quality
	-issupperpeer
Peer platform profile	-Peer platformID
	-Peer up bandwith
	-Peer down bandwith
	-Peer type machine
Peer resource	-Peer resourceid
	-Peer resource type
	-Peer resourceuri
Peer query	-Peer query content
	-Query id
	-Query type
Peer activities	-Query used
	-Activity datetime
	-Peer activitystatus
Debit	-Upload
	-Download
Peer	-Simple Peer
	-Supper Peer
File	-Created by
	-Size
	-File type

Table 2: Relation Table

Retaliation name	Concept source	SC	Target concept	TC	Inverse relation
Belong to community	Peer cluster	11	Peer community	0N	Containes cluster
Has supper	SimplePeer	11	SupperPeer	1.N	HasChild
Belongs to cluster	Peer	01	Peer cluster	0N	Contains peer
Supper peer in cluster	Peer cluster	1N	Supper peer	11	Is supper peer in cluster
Has nieghbor	Peer cluster	0N	Peer cluster	0N	Has neighbor
Describes logically	Peer logical profile	11	Peer	11	Described logically
Described in platform	Peer	1N	Peer platform profile	1N	Describes peer platform
Has activity	Peer	1N	Peer activity	1,1	Is done by
Peer sends query	Peer	1N	Peer query	11	Query sended from
Peer receives query	Peer	1N	PeerQuery	1N	Query received by
Act on	Peer query	0N	Peer resource	0N	Is performed
File sended by	File	0N	Peer	0N	Sends file
Exports	Peer	0N	Peer ressource	1N	Exported
Query used	Peer activities	11	Peer query	1N	DoneIn zctivity

In addition to the inverse relation, other semantic relations can be used to elaborate the ontology, as Transitivity and symmetric. So, the role named

Peer community ⊆ T	**Peer profile ⊆ T**
Peer cluster ⊆ T	**Supper peer ⊆ Peer**
Peer query ⊆ T	**Peer resource ⊆ T**
Peer down bandwith ⊆ download	**File ⊆**
Peer up bandwith ⊆ upload	**Peer resource**
Peer platform profile ⊆ T	**Peer activity ⊆ T**
Peer logical profile ⊆ peer profile	
Peer plat form profile ⊆ peer profile	
.....	

Fig. 15: Concepts inclusions

Belong to community (Peer cluster, Peer community)
Has supper (Simple peer, Supper peer)
Belongs to cluster (Peer, Peer cluster)
Has nieghbor (Peer cluster, Peer cluster)
Describes logically (Peer logical profile, Peer)
Query sended from (Peer query, Peer)
Peer receives query (Peer, Peer query)
Query used (Peer activities, Peer query)
Described in platform (Peer, Peer platform profile)
Has activity (Peer, Peer activity)
Act on (Peer query, Peer ressource)
File sended by (File, Peer)
Belong to community ¬ Containes cluster
Has supper ¬ Has child
Belongs to cluster ¬ Contains peer
Has nieghbor ¬ Has neighbor
Describes logically ¬ Described logically
Query sended from ¬ Peer sends query
Peer receives query ¬ Query received by
Query used ¬ Done in activity
Described in platform ¬ Describes peer platform
Has activity ¬ Is done by
Act on ¬ Is performed
File sended by ¬ Sends sile

Fig. 16: Roles definitions

"Has Neighbor" is considered as symmetric one (If P1 "Hasneighbor" P2 then P2 "Hasneighbour" P1),

The role named "SendsQuery" is considered as transitive one (If P1 "SendsQuery" P2 and P2 "SendsQuery" P3 then P1 "SendsQuery" P3").

Also, the "HasNeighbor" is a transitive role, because (If P1 "Hasneighbor" P2 and P2 "Hasneighbor" P3 then P1 "Hasneighbor" P3").

The OWL language provides theses possibilities to represent the semantic of the relations.

- To indicate that the relation "ContainsPeer" is the inverse of "BelongsToCluster" we use : "<owl:inverseOf rdf:resource="#BelongsToCluster"/>".
- To indicate the relation "HasNieghbor" is symmetric, we use: <owl:ObjectProperty rdf:ID="HasNeighbor"> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/><rdfs:range rdf:resource="#PeerCluster"/> <rdfs:domain rdf: resource="#PeerCluster"/>

- To indicate that the "HasNighbor" is transitive we use the following OWL syntax: <owl:ObjectProperty rdf:ID="HasNeighbor"> <rdf:type df:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/><rdfs:range rdf: resource="#PeerCluster"/><rdfs:domain rdf:resource ="#PeerCluster"/><owl:inverseOf rdf:resource="# HasNeighbor"/></owl:ObjectProperty>

Ontology representation in description logic is the most important representation in the ontology building process. The following is the ontology representation in SHIQ description logic.

Debit:= Upload ∪ downLoad

Download:= ¬ Upload
PeerCommunity:= (∃ PeerComID.String)∩(∃ Peer Comdesignation.String)∩(∃ PeerComDomaine.String)∩ (∃ Description.String) ∩ (≤ 1 ContainsCluster PeerCluster)

Peer cluster:= (∃ClusterID.String) ∩ (∃ NumberOfPeers. Integer) ∩(∃NumberOfRedundancy.Integer) ∩ (∃ClusterUpBandwith.Double)∩ (∃ClusterDownBandwith. Double)∩ (∃ BelongsToCommunity PeerCommunity) ∩(≤ 1 ContainsPeer Peer) ∩(≤0 HasNeighbor PeerCluster) ∩(=3 SupperPeerInCluster SupperPeer)

Peer profile:= PeerLogicalProfile ∪ PeerPlatformProfile

Peer logical profile:= ¬ PeerPlatformProfile

Peer logical profile: =PeerProfile ∩ (∃ PeerID.String) ∩ (∃ OnlineState.Boolean)∩(∃ Platform in use.String)∩(∃ PeerQuality.Integer)∩(∃ IsSuperPeer.Boolean)∩(∃ Describes Logically Peer)

Peer platform profile: = PeerProfile ∩ (∃ PeerPlatformID. String) ∩ (∃PeerUpBandwith.Double) ∩ (∃ Peer DawnBandwith.Double) ∩ (∃ PeerTypeMachine.String) ∩ (<1 DescribesPeerPlatform Peer)

Peer ressource: = (∃ PeerRessourceID.String) ∩ (∃ PeerRessourceType.String) ∩ (∃ PeerRessourcesURI. String) ∩ (<1 IsPerformed PeerQuery)

Peer query:= (∃ QueryID.String)∩(∃ PeerQueryContent. String (<1 QuerySendedFrom Peer) ∩(<1 QeuryReceived By Peer) ∩(∃ DoneInActivity PeerActivities)

Peer activity: = (∃ Activity Date Time) ∩ (∃ Peer Activity Statues. String) ∩ (∃ Query Used Peer Query)

File: = Peer Ressource ∩ (∃ Created By. String) ∩ (∃ Size. Double) ∩ (∃ File Type. String) ∩ (<0 File Sended By Peer)

Simple peer: = Peer ∩(∃ Has Supper Supper File)

Supper peer: = Peer ∩(∃ Has Child Simple Peer)

Peer: = Simple Peer ∪ SupperPeer ∩ (∃ Described Logically Peer Logical Profile) ∩ (<1 Described In Platform Peer Platform Profile)∩(∃ Belongs To Cluster Peer Cluster)

The codification step: In this step, we create the OWL document based on the previous representation. We use the well known Ontology Editor named "Protege2000 Version 3.1.1" in order to produce the OWL document. In Fig. 17, we present the concept Hierarchy as created in the editor.

After creation of concepts hierarchy, we create the properties of each concept. Fig 18 shows the roles and the attributes of the "PeerPlatformProfile" concepts.

In the following, we present a part of the ontology coded in the OWL language.

```
<?xml version="1.0"?><rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns="http://www.owl-ontologies.com/unnamed.owl#"
xml:base="http://www.owl-ontologies.com/unnamed.
owl"> <owl:Ontology rdf:about=""/> <owl:Class rdf:ID
="SimplePeer"> <rdfs:subClassOf> <owl:Class
rdf:ID="Peer"/> </rdfs:subClassOf>
<rdfs:subClassOf> <owl:Restriction>
<owl:cardinality rdf:datatype="http://www.w3.org/
2001/XMLSchema#int" >1</owl:cardinality>
<owl:onProperty> <owl:ObjectProperty
rdf:ID="HasSupper"/> </owl:onProperty>
</owl:Restriction> </rdfs:subClassOf>
</owl:Class> <owl:Class rdf:ID="PeerPlatformProfile">
<rdfs:subClassOf> <owl:Class rdf:ID="PeerProfile"/>
 </rdfs:subClassOf> </owl:Class> <owl:Class
rdf:ID="PeerLogicalProfile"> <rdfs:subClassOf>
<owl:Class rdf:about="#PeerProfile"/>
</rdfs:subClassOf> </owl:Class> <owl:Class
rdf:ID="Upload"> <rdfs:subClassOf> <owl:Class
rdf:ID="Debit"/> </rdfs:subClassOf>
<owl:disjointWith> <owl:Class rdf:ID="Download"/>
 </owl:disjointWith> </owl:Class> <owl:Class
rdf:ID="PeerCommunity"> <rdfs:subClassOf>
```

Fig .17 Concept hierarchy in Protégé 2000

Fig. 18 "Peer Platform Profile" properties

```
<owl:Restriction> <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
 >1</owl:minCardinality> <owl:onProperty>
<owl:InverseFunctionalProperty rdf:ID="ContainsPeer"/>
 </owl:onProperty> </owl:Restriction>
</rdfs:subClassOf> <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>…
 …
```

THE ROAMING COMPONENTS

The proposed ontology permits to have a semantic profile for the peer. But, in order to manage this profile, we must create the appropriate components. Thus, three components are proposed; the interceptor, the profile updater and the wrapper.

The interceptor: The main role of this component is to capture the activities of peer in the used platform. This is done by gathering information about the traffic network done by the peer and sent by the wrapper. The most important activities that are captured are: Connect, AddRessource, Disconnect, SendsRessource, Receives Ressource. After capturing the activities, it sends information to the profile manager. Figure 19 represents the UML class diagram of Interceptor.

In the following, we give a description of the class members.

The attributes of the interceptor class are:

- ProfileLocation contains the URI of the profile as stored in the peer machine.
- PlatformInUse saves the Platform that is used by the peer in the current session.
- UniversalPID represents the Peer Identification. Its value is extracted from the Profile.
- Profile Manager Port is the IP port that is used to establish a connection with the profile manager. We can choose any value not in use by other applications. In our study, the default value is 2302.

The methods of the class are:

- ActivateRoaming( ) permits to activate the Roaming service. It sends the platform characteristics to the profile manager and sends the value of the PeerID to the wrapper. The WrapperPort argument represents the IP port of the wrapper that is used to capture the activation request sent by the wrapper.
- InterceptQuery( ) is used to intercept the query sent by the wrapper.
- SendUP( ) is the method used bay the activate Roaming one, in order to send the PeerID to the wrapper.
- Connect ( ) deals with the Profile Manager. Its role is to inform it that the intercepted query is a connect one.

Fig. 19: UML class diagram for Interceptor.

Fig. 20: UML class diagram for Profile manager

- AddRessource( ) deals with the Profile manager. It informs it that intercepted query is AddRessource( ) one.
- The same thing of the following methods; Disconnect, Send Ressource( ), Receive Ressource( ).

The profile manager: The main role of the Profile manager is to update the Peer profile according to queries sent by the Interceptor. The latter do not have a direct access to the Peer profile. The separation between the two components is motivated by the need of modularity in our approach. So, any component can be extended without influencing the other ones.

Figure 20 represents the UML diagram class for the Profile Manager.

In the following, we give a description of the class members.

The attribute of the ProfileManager class are:

- ProfileLocation contains the URI of the profile as stored in the machine.

- CurrentComunity determines the community used by the peer.
- InterceptorPort is the IP port that used to establish a connection with the Interceptor. The default value is 1978.

The methods of profile manager are:

- IsSupper( ) determines if the peer is a supper one. This is done by getting the value from the attribute "IsSupperPeer" of the PeerLogicalProfile concept.
- GetCummunity( ) permits to get the name of the community.
- GetQuality( ) gives the quality of the peer. The quality is calculated by the CalculateQuality() equation
- GetProfile ( ) permits to return the whole peer profile (the PeerProfile.owl document). The latter will be used by other peer in order to establish specific connections.
- CalculateQuality( ) is the method in charge to calculate the quality of the current Peer. Many techniques can be used in order to estimate the value of the quality. In our study we have used a simplified equation:

$$PQ = \frac{\sum \text{SatisfiedQuery}}{\sum \text{RequestedQuery}} \qquad (1)$$

- AddRessource ( ) permits to add an individual in the ontology in order to determine that the resource can be exported by the peer.
- SetPeerType ( ) sets the peer type according to the argument value; 0 for SimplePeer, 1 for SupperPeer. The default value is 0.
- Set Platform Caracteristics ( ) updates the value of the concept "Peer Platform Ptrofile" with the current characteristics of the used platform.
- Send Resource ( ) adds an instance in the "Peer Activitie" concept with the information of the sent resource and the remote peer.
- Receive Ressource ( ) adds an instance in the "Peer Activitie" concept with the information of the received resource and the remote peer.
- Connect ( ) modifies the value of the attribute "On Line State" of "Peer Logical Profile" with true value.
- Disconnect ( ) modifies the value of the attribute "On Line State" of "Peer Logical Profile" with false value.

The wrapper: The wrapper component has two main roles:

- Spying the activities of the peer. So, all the platform activities can be captured and sent to Interceptor component.

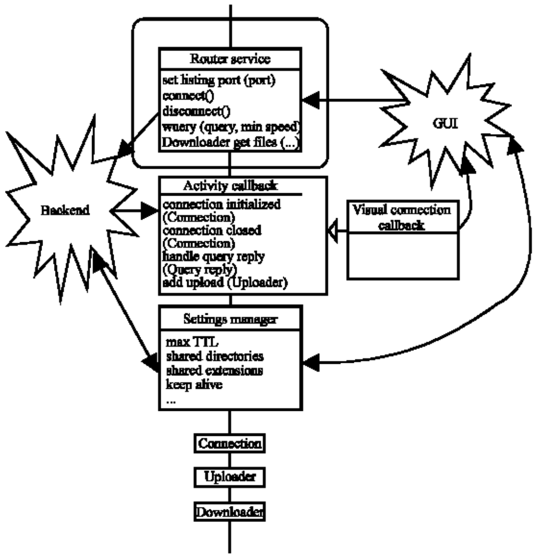

Fig. 21: A part of UML class diagram of LimeWire

- Dealing with the platform GUI (Graphic User Interface) in order to put more information about the peer, as the Peer ID, Peer Quality and Issupper Peer.

The spying technique consists of intercepting the data traffics on the used IP port. This is the solution if one can not use the source code.

In our study, we have used the PGrid and LimeWire platforms that are in open source. So, we can modify the code source to accomplish the spying requirement. In the following, we will explain the modifications done on the LimeWire application that is based on Gnutella platform. The LimeWire provides the UML diagram of the application. Therefore, the modification can be done first on the model, before implementing it. LimeWire separates between the GUI and the core of the activities (the Backend). Fig 21 shows a part of the UML class diagram of LimeWire. Our modification is applied on the Router Service class. The latter contains the methods that are necessary to the spy requirement as:

- Connect ()
- Disconnect ()
- Query (Query, MinSpeed)
- Downloader Getfile( )
- Uploader sendFile()

At the end of each method, we add the code that sends a description of the activity to the interceptor. This is done by using the IP port of the interceptor to send theses messages.

An other modification is to modify the GUI of the application, by adding a specific columns (PeerID, Peer Quality and IssuperPeer). We can see that the modifications are littlies. The techniques of routing and file transferring are kept. Only a little code is added and some modifications on the GUI of the application are made. The some operations are made on the PGrid Platform, so we have the same identification between the tow platforms

CONCLUSION

In this study, we presented our work to integrate the peer to peer platforms. Our approach allows the peer to navigate across different platforms and to maintain a history of the peer. This is assured by using a semantic profile and a roaming service. The semantic profile is created according to a clear ontology development process. Thus, the structure of the whole system is exported as a semantic web resource. The roaming service provides the capability to navigate across different platforms and permits to deal with the specific aspects of each platform by using unique peer identification. The main benefic of such ontology and service is to permit to other platforms or users to share a global structure and permits to locate a peer independently to a specific platform, especially in platform integration case.

REFERENCES

1. Gnutella home page. www.gnutella.com.
2. Kazaa home page. www.kazaa.com.
3. Ian, C., S.G. Miller, T.W. Hong, O. Sandberg and B. Wiley, 2002. Protecting Free Expression Online with Freenet. IEEE Internet Computing pp: 6.
4. Stoica, I., R. Morris, M. Kaashoek and H. Balakrishnan, 2001. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. of ACM SIGCOMM'01, San Diego, CA, USA.
5. Ratnasamy, S., P. Francis, M. Handley, R. Karp and S. Shenker, 2001. A Scalable Content-addressable Network. In Proc. of ACM SIGCOMM'01, San Diego, CA, USA.
6. Yang, B. and H. Garcia-Molina, 2002. Improving Efficiency of Peer-to-Peer Search. In Proc. of the 28th Intl. Conf. on Distributed Computing Systems.
7. Yang, B. and H. Garcia-Molina, 2002. Designing a super peer network.
8. Mike, D., D. Connolly, F.V. Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-schneider and L.A. Stein, 2003. Web ontology language (OWL) reference version 1.0. W3CWorking Draft. Available at http://www.w3. org/TR/2003/WD-owl-ref-20030331.
9. Nejdl, W., *et al.*, 2003. Super-peer-based routing and clustering strategies for rdf-based peer-topeer networks. In: Proceedings of the twelfth international World Wide Web Conference (WWW), Budapest, Hungary.
10. Aberer, K., P. Cudr´e-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva and R. Schmidt, 2003. P-grid: A self-organizing structured p2p system. ACM SIGMOD Record, pp: 32.
11. Ehrig, M., P. Haase, R. Siebes, S. Staab, H. Stuckenschmidt, R. Studer and C. Tempich, 2003. The SWAP Data and metadata model for semantics-based peer-to-peer systems. In Multiagent System Technologies (MATES).
12. Cai, M. and M. Frank, 2004. RDFPeers: A scalable distributed rdf repository based on a structured peer-to-peer network. In International World Wide Web Conference (WWW).
13. Younes D. and Z. Boufaida , 2005. Enhancing the Peer-To-Peer Architecture with a roaming service and OWL". Enformatika, Transactions on Engineering, Computing and Technology ISSN 1305-5313, pp: 37-43.
14. Hemam, M. and Z. Boufaida, 2004. An Ontology development Process for the Semantic Web. In ACIT'2004 N5, International Arab Conference on Information Technology, pp: 595-601.