*REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE*

*MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE*

*LA RECHERCHE SCIENTIFIQUE*

*UNIVERSITE MOLOUD MAMMERI DE TIZI-OUZOU*

*Faculté de Génie Electrique et d'Informatique*

*Département d'Informatique*

# Mémoire

## *de fin d'études*

**En vue de l'obtention d'un diplôme en Master 2 en informatique**

**Option : Conduite des projets informatique**

*Thème :*

# *Interopérabilité entre les ontologies dans un réseau Peer to Peer(P2P)*

**Proposé et Dirigé par :**

**Mr M.KERBICHE**

**Réalisé par :**

**Melle : Latoui Linda**

**Jury composé de :**

**Président :**

**Examinateurs :……**

**……..**

**Promotion : 2011/2012**

# Remerciements

**Je tiens à remercier vivement mon promoteur Mr M.KERBICHE pour m'avoir proposé ce sujet, pour la qualité de son encadrement, et son suivi toute la durée du projet.**

**Je remercie chaleureusement les membres de jury pour l'honneur qu'ils me font en acceptant de juger ce mémoire de fin d'études.**

**En fin je remercie toutes les personnes ayant contribué de prés ou de lion au bon accomplissement de ce travail.**

# Dédicaes

Je dédie ce travail à mon très cher mari yazid, à mes très chers parents, à mes frères Farid et Belkacem, à ma chère sœur Nadia et son mari Djilali, à toute ma belle famille et à toutes mes amies.

# *Sommaire*

# *Sommaire*

**Chapitre II : Les technique de clustering des réseaux Peer to Peer et l'alignement des ontologies**

## *Liste des tableaux :*



## *Liste des Figures :*

# *Introduction générale*

## Introduction générale :

Les applications Peer-to-Peer (P2P), deviennent très populaires à partir du début des années 2000. Actuellement, une part importante du trafic internet est due aux applications P2P.

Dans ces réseaux, un pair est à la fois un client et serveur **[Guy Pujolle,2004]**. Il n'y a donc plus de serveurs centraux qui regroupent l'ensemble des informations, l'information est distribuée sur l'ensemble des machines connectées à travers le réseau. Parmi les nombreuses applications P2P, on compte les applications de web sémantique, les messageries instantanées, les outils de recherche, etc. La classe la plus connue d'application P2P est représentée par des logiciels de partage de fichiers musicaux tels que Napster,Groove, Magi (architecture avec index centralisé),Gnutella, Freenet (architecture décentralisée).

Dans le contexte du Web Sémantique, on utilise des ontologies pour faire communiquer des applications. Une définition consensuelle, précise et complète des ontologies dans le contexte du Web sémantique n'existe pas encore **[Bach, 2004].** Cependant, Gruber en 1993 proposait la définition la plus citée : " une ontologie est une spécification explicite d'une conceptualisation" **[Gruber, 1993]**. La conceptualisation est un processus utilisant les produits de l'analyse pour produire les spécifications d'une solution d'un système. Cette conceptualisation est représentée dans une forme concrète où les concepts, les relations ainsi que les contraintes sont explicitement définis dans un formalisme de représentation.

Partant du fait que plusieurs connaissances peuvent prendre des représentations différentes et les applications doivent collaborer pour produire une activité visible par l'utilisateur, il est alors nécessaire de disposer de mécanismes permettant de faire des liens entre les connaissances exprimées dans chacune des ontologies.

Notre contexte, prend en considération un ensemble d'ontologies réparties dans un réseau P2P. Vu l'absence d'une ontologie globale pour faire collaborer ces

ontologies, nous essayons de mettre l'accent sur la définition des similarités entre ces ontologies, qui seront exploités dans une démarche de Clusterisation des réseaux P2P avec l'utilisation de patterns dans des topologies virtuelles pour garantir au mieux l'interopérabilité, les performances des applications et le passage à l'échelle.

En effet, des travaux précédents ont essayé de résoudre cette problématique en proposant des techniques s'appuyant seulement sur l'aspect géographique des pairs ou sur le contenu des ontologies (alignement, distances sémantique, etc.) [**Jérôme Euzenat et al, 2004**]. Généralement les techniques qui s'appuient sur l'aspect géographique assurent le passage à l'échelle, mais soufrent de l'incohérence des clusters, ce qui est le contraire dans les techniques qui se basent sur le contenu des ontologies.

Vu que ces deux techniques sont en conflit l'une par rapport à l'autre, il est nécessaire de définir une technique qui agrège au maximum les avantages de chacune en éliminant leurs inconvénients.

Notre solution définit les distances entre les catégories de concepts des ontologies pour assurer la cohésion à l'intérieur des clusters et simplifier le processus d'assurance de l'interopérabilité car les ontologies intermédiaires pour répondre à des requêtes sont éliminées. Les techniques de Clustering des réseaux P2P sont utilisées pour la formation des Clusters et améliorer ainsi le passage à l'échelle. Finalement certaines configurations de pairs sont utilisées pour assurer l'interopérabilité à l'intérieur et entre clusters.

Dans notre expérimentation, nous avons utilisé l'environnement de développement Net Beans pour assimiler les ontologies et le processus de Clustering. Les résultats de cette expérimentation montrent la possibilité de créer des clusters fortement cohésifs et faiblement liés, ce qui valide notre approche sur des cas réels.
Ce sujet, présente certaines difficultés car il est d'actualité et demande la maitrise de plusieurs domaines : P2P, ontologies, clustering, alignement des ontologies, recherche d'information et programmation.

Ce travail comporte trois chapitres brièvement décrit comme suit :

- **Chapitre I** s'intitule : ‹‹ **Généralité sur les réseaux Peer to Peer (P2P)** ››, présente quelques notions de base concernant les réseaux Peer to Peer(P2P) leurs définitions, principe, les modèles existants, les applications, et la liste des logiciels utilisé dans les réseaux P2P.
- **Chapitre II** s'intitule **:** ‹‹**les technique de clustering des réseaux Peer to Peer et l'alignement des ontologies** ››présente les notions de base des ontologies(définition, composants,languages) et l'état de l'art sur le Clustering des réseaux P2P et les techniques utilisées ainsi présente l'état de l'art de l'Alignement des ontologies et les techniques de distance utilisées.
- **Chapitre III s'intitule :** ‹‹**Positionnement et Solution** ››présente notre solution.

# *Chapitre I :*

*Généralité sur les systèmes Peer to Peer (P2P)*

**I.1 Introduction** :

L'internet est composé d'immenses ressources qui sont potentiellement utilisables mais souvent mal exploitées. Ces ressources sont de trois types: bande-passante, espace de stockage et capacité de traitement. C'est parce que le modèle client-serveur classique ne parvient pas à tirer le maximum des ressources du réseau que le modèle P2P est né.

**I.2 Définition [1] :**

Une application est dite Pair à Pair (P2P Peer To Peer) si elle met en relation des programmes de même nature sans intermédiaire. La grande caractéristique des applications Peer To Peer est que chaque participant (peer ou pair) joue à la fois le rôle de client et de serveur. Il va donc à la fois proposer et consommer des « services ».

**I.3 Principe : [1]**

Le peer-to-peer ("égal à égal", ou point à point) est un modèle d'architecture de système d'information dont le principal objectif est d'optimiser au maximum l'utilisation d'au moins un des trois types de ressources existantes par rapport à son équivalent en modèle client-serveur.

Le principe est de connecter directement deux machines du réseau dès qu'elles ont des informations communes qui les intéressent. Autrement dit cette connexion permet une relation d'égal à égal entre les deux postes. Il n'est donc plus nécessaire de passer par un serveur central pour dialoguer avec une machine distante.

**I.4 Les modèles existants : [1]**

Il existe deux modèles d'architecture P2P : le modèle centralisé et décentralisé. On se propose ici d'étudier ces modèles comparativement à celui du client-serveur .

- **L'architecture client-serveur** :

En client-serveur, tous les échanges entre clients passent forcément par le serveur. Autrement dit, les seules ressources publiées et consommées sont celles situées sur le serveur.


**Figure I.1 : Architecture client/serveur**

On peut citer l'exemple du service de messagerie mail (modèle client-serveur) lors de l'envoi d'un mail, ce dernier sera obligatoirement acheminé vers le serveur mail de l'expéditeur, puis vers celui du destinataire pour qu'enfin ce dernier puisse réceptionner le mail à sa reconnexion au serveur.

- **L'architecture P2P centralisée :**

P2P centralisé, un serveur de noms centralise l'entrée en communication et la coordination des pairs. Le reste des échanges s'effectue ensuite par connexion directe

entre les noeuds concernés. Globalement, la majorité des échanges va se faire par connexion directe.

L'application de partage de fichiers musicaux Napster est certainement l'exemple le plus connu réalisant ce type de modèle. Elle fonctionne de la manière suivante : l'utilisateur, lorsqu'il se connecte, expédie dans le même temps, les fichiers MP3 qu'il possède. Ainsi, à tout moment, les serveurs Napster savent quels sont les fichiers actuellement disponibles sur le réseau et sur quelles machines. Ensuite, à tout moment, un utilisateur de Napster peut contacter le serveur central pour demander une musique.

Ce serveur consulte juste une base de données qui contient l'index (fichier MP3-adresses des machines le possédant).Le transfert de fichiers va ensuite se faire par connexion directe avec une des machines possédant le fichier. Ici, on voit que l'utilisation de la bande-passante et de l'espace de stockage est optimisée.

On pourrait aussi citer l'exemple de Groove qui offre un service collaboratif d'édition de fichiers avec un système de groupe de travail. Ici, le rôle des serveurs centraux est aussi la coordination des pairs sur les différentes copies d'un document.

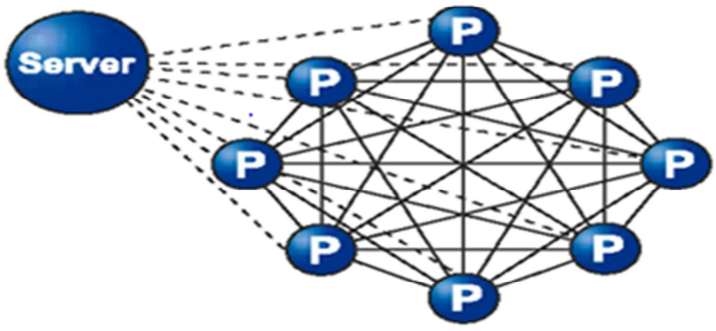


**Figure I.2 : Architecture P2P centralisée**

- **L'architecture P2P décentralisée** :

Le modèle décentralisé ou "pur P2P", ne comporte pas de serveur de noms. Les noeuds doivent donc assurer la connexion des nouveaux noeuds. Cela peut se faire par différents mécanismes:

- On connait au moins un pair toujours présent (mécanisme à point de rendez-vous)
- broadcasting local ou multicast sur adresse déterminée
- invitation d'un pair déjà connectée
- autres...

Après cette étape de connexion, le fonctionnement est exactement le même que dans le modèle centralisé. Chaque pair joue à tour de rôle les fonctions de serveur ou de client. En tant que serveur ils publient des ressources et répondent ou orientent les recherches des autres pairs. En tant que client, elles consomment des ressources.

Un des exemples le plus souvent cité est Gnutella, application de partage de fichiers. Gnutella fonctionne sur ce modèle P2P décentralisé. La découverte fonctionne avec un mécanisme à point de rendez-vous La transmission de requête se fait selon un mécanisme de proche en proche, avec la notion de voisin virtuel (algorithme de vague). Chaque pair a ainsi une vision locale du réseau. Le problème de boucle infinie est réglé grâce à l'utilisation d'un champ TTL (Time to Live) décrémenté à chaque passage dans un nœud.

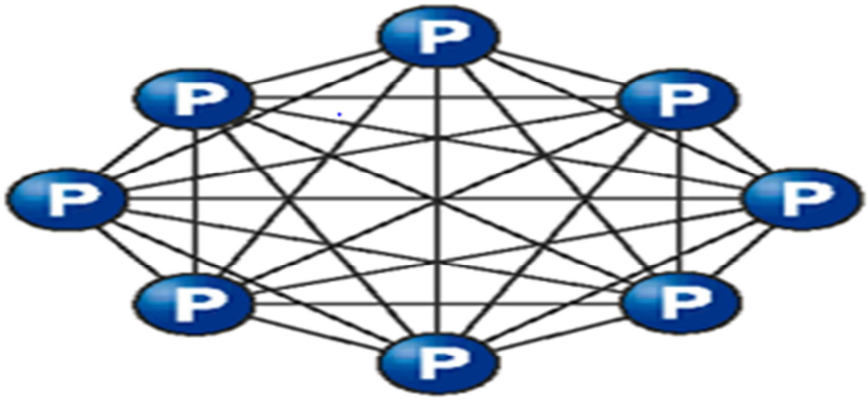


**Figure I.3 :Architecture P2P décentralisée**

## I.5 Applications dans le modèle P2P : [2]

- Applications distribuées (Alternative au RPC)
- Répartition de stockage

  - Multimédia, …
- Répartition de puissance de calcul
- Applications de collaboration
- Messageries instantanées
- Indexation et moteurs de recherche
- Jeux en réseau

  -exemple Xtank, Tron,
- …

## I.6 Plate-forme de P2P : [2]

- P2P et Web Services
  - Recherche de services UDDI.
  - Communautés de consomateurs.

- Microsoft
    - NET My Services (ex HailStorm)
    - Embedded eMbedded Visual Tools + ViaXML
- Sun
    - SunOne + JXTA
- JINI
    - Entre des devices géographiquement proches.

**I.7 Liste de logiciels P2P** : **[3]**

- **BitTorrent :**

Conçu en 2001 par Bram Cohen, est un protocole de transfert de données pair-à-pair (P2P). Celui-ci, réalisé en langage Python, est actuellement maintenu par la société BitTorrent Inc crée par Bram Cohen en 2002. Selon une étude réalisée en 2007 par l'entreprise ipoque, il utiliserait actuellement plus du tiers de la bande passante totale d'Internet. Sa popularité grandissante est à mettre au profit de sa facilité d'utilisation et de son importante vitesse de transfert.

- **Napster :**
    - Permet la référence de façon centralisée des fichiers de musiques, et propose à leurs auteurs de les échanger directement entre eux.
- **Gnutella et ses différentes implémentations** :
    - permettent l'échange totalement décentralisé de fichiers divers.
- **Freenet permet :**
    - échange et recherche de fichiers ou documents
    - garantit l'anonymat
    - Migration des fichiers vers des centres de gravité de forte utilisation

- **eDonkey2000** :

est un logiciel de partage de fichiers utilisant des techniques de P2P ,Il était développé par Meta Machine et utilise le protocole de transfert de fichiers multisource. Le client eDonkey peut travailler à la fois avec le réseau eDonkey et le réseau Overnet.

- **GNUnet** :

Est un réseau P2P informatique garantissant l'anonymat, n'utilisant aucun service centralisé ou autre service de confiance. Une première implémentation au-dessus de la couche réseau permet un partage de fichier anonyme et résistant à la censure. GNUnet utilise un système simple basé sur un système économique d'allocation des ressources. Les nœuds GNUnet prennent en compte les demandes des autres nœuds avec un respect des ressources disponibles ; les nœuds contribuant au réseau sont récompensés par un service de meilleure qualité. Il fait partie du projet GNU, GNUnet propose une option "F2F topology" pour construire des réseaux de type ami-à-ami.

**I.8 Avantages et inconvénients P2P : [3]**

- ✓ **Avantages :**

**- Rapidité d'exécution:** le P2P confère une rapidité d'exécution accrue notamment grâce à l'extension des ressources mises à disposition (exemple du SETI), et à la réduction des temps de transits (connexion directe).

-**Utilisation de la bande passante:** Le P2P exploite mieux la bande passante. En effet, il évite la formation de centres névralgiques de passage. On pourrait d'ailleurs se demander si ces connexions ne pourraient pas être très vite encombrées, avec ce mode de fonctionnement sur des connections souvent à faible débit, mais les chiffres

montrent que les consommateurs vont de plus en plus vers les connexions à haut débit.

**-Tolérance aux pannes**: En client-serveur, la chute des serveurs entraine irrémédiablement la panne du service. En P2P, l'instabilité du réseau est compensée par la redondance des informations.

**- Coût:** Le P2P réduit les coûts. La maintenance est effectuée et l'équipement est financé, par l'utilisateur.

**-Capacité d'extensibilité**: l'extension de ressources se fait de manière dynamique. On augmente le nombre de ressources, en augmentant le nombre d'utilisateurs.

**- Ethique**: Nul n'est censé ignorer les déboires judiciaires de Napster...on se heurte ici aux problèmes légaux (copyright), moraux (pornographie infantile) etc...

✓ **Inconvénients**: **[2]**

- Utilisation des ressources inutilisée : CPU, Stockage, …
- QoS :
  -Ligne peu fiable, débit peu élévé…
- Sécurité
  - Crackers
  - Virus
  -Distributed Deny of Service (DDoS)
  - Confidentialité
  -Authentification
- Contenu trompeur
  - Consistance
  - Contradiction

- Loi : *Wild Wild Web*

  - Droit d'auteurs

  -Contenu immoral

  - Pornographie infantile, …

- Régulation / Répression

  - Application difficile des lois.

**I.9 Conclusion :**

Les systèmes pairs à pair (P2P) constituent une plateforme récente pour exécuter des applications reparties dans des environnements à grande échelle. Dans ce type de systèmes, les nœuds peuvent se comporter à la fois comme client et serveur.

Avec les réseaux peer-to-peer, les ordinateurs partagent les données et les ressources, en communiquant directement entre eux sans utiliser un serveur central. Les entreprises et les organisations nécessitent de mettre leurs concepts dans des ontologies qui sont logées dans un système P2P.

Pour cela dans le chapitre suivant on va étudier les technique de clustering de système peer to peer et l'alignement des ontologies.

# *Chapitre II :*

*Les technique de clustering des réseaux Peer to Peer et l'alignement des ontologies*

## II.1. Introduction :

Les applications Peer-to-Peer, aussi appelées P2P, deviennent très populaires à partir du début des années 2000. Des pointes de plus de 50% du trafic internet ont été observées pour ce type d'application, qui correspond à des trafics allant directement entre machines et terminales.

Dans cette configuration, une machine terminale est à la fois un client et un serveur Il n'y a donc plus de serveurs centraux qui regroupent l'ensemble des informations, l'information est distribuée sur l'ensemble des machines connectées au système distribué. Parmi les nombreuses applications P2P, on compte le web sémantique, les messageries instantanées, les outils de recherche...etc.

En ce qui concerne les informations, et plus particulièrement leur représentation et leur sémantique, l'ontologie est apparue comme un moyen de structuration formelle, contribuant à faciliter leur compréhension. Grace à elle la construction d'un web sémantique est vue comme un rêve devenu réalité.

## II.2 Les ontologies :

### II.2.1 Définition d'une ontologie :

La définition de l'ontologie dans le domaine informatique a été présentée **[Neches, 1991]:** comme suit: «une ontologie définit les termes et les relations de base comportant le vocabulaire d'un domaine aussi bien que les règles pour combiner des termes et les relations afin de définir des extensions du vocabulaire ».

### II.2.2 Composants d'une ontologie :

Les connaissances traduites par une ontologie sont à véhiculer à l'aide des éléments Suivants **[Gomez-Perez, 1999]** :

- **Les concepts :** les concepts doivent être compris dans un sens très large, ils peuvent être classifiés selon plusieurs dimensions :
  - **niveau d'abstraction** : les concepts représentent les objets abstraits ou concrets du monde réel.
  - **Atomicité :** les concepts représentent les objets élémentaires ou composites du

monde réel.

- **niveau de réalité :** les concepts représentent les objets réels ou fictifs du monde réel.

Un concept pourrait être aussi la description d'une tache, d'une fonction, d'une action, d'une stratégie, d'un processus de raisonnement…etc.

- **Les relations :** les relations traduisent les associations (pertinentes) existant entre les concepts du domaine. Ces relations incluent les associations suivantes:
  - **Sous-classe-de :** les classes sont organisées par des relations taxonomiques selon un ordre de généralisation ou de spécialisation;
  - **Partie de :** une classe peut définir une partie d'une autre classe par une relation d'agrégation ou de composition ;
  - **Instance-de :** on peut relier une instance avec sa classe d'origine,…etc.
- **Les fonctions :** les fonctions sont un cas spécial de relations dans lesquelles le nième élément de la relation est unique pour les n-1 éléments précédents.
- **Les axiomes :** les axiomes constituent des assertions, acceptées comme vraies.
- **Les instances :** les instances représentent les éléments extensionnels spécifiques au domaine du problème modélisé.

**II.2.3 Langage d'ontologie :**

Les langages d'ontologie sont des langages formels permettant de représenter une ontologie. Beaucoup de ces langages se bornent à permettre l'expression d'ensemble de concepts et leurs relations conceptuelles. Ils suffisent à construire des ontologies légères ayant des propriétés minimales nécessaires à la représentation et l'appréhension d'un domaine. Cependant, de plus en plus, il s'avère nécessaire de pouvoir ajouter aux ontologies légères des axiomes et des restrictions clarifiants le sens. Pour construire des ontologies lourdes qui modélisent un domaine de façon plus profonde.

Il existe plusieurs langages informatiques spécialisés dans la création et la manipulation des ontologies. Parmi ces langages nous citons :

- **RDF, RDF(S) :** RDF « Resource Description Framework » un langage XML permettant de décrire des métadonnées et facilitant leur traitement **[Xavier Lacot, 2005].** Il est basé sur la notion de triplet (sujet, prédicat, objet) pour représenter les ressources (objets réels, concepts..) et pour que les informations présentes dans les graphes, sous la forme d'un réseau sémantique, soient utilisables par une machine. Cependant, RDF n'a pas de mécanismes pour définir des relations entre des propriétés et entre des ressources, c'est le rôle de RDF(S). Ce dernier définit des classes et des propriétés utilisables pour décrire des classes, des propriétés et d'autres ressources.

**-OWL (Web Ontology Langage) :** OWL est, tout comme RDF, un langage XML profitant de l'universalité syntaxique de XML. Fondé sur la syntaxe de RDF/XML, OWL offre un moyen d'écrire des ontologies web. OWL se différencie du couple RDF car il intègre, en plus, des outils de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, OWL se compose de trois sous-langages d'expressivité croissante : OWL Lite, OWL DL et OWL Full. etc **[Xavier Lacot, 2005].**

### II.3. Etat de l'art sur le Clustering des réseaux P2P:

#### II.3.1 Définition de clustering :

Le clustering est le partitionnement des données en cluster, tels que les données d'un même cluster soient similaires et que les données des deux clusters distincts soient différentes. **[4]**

-**Cluster :** un cluster est un groupe de tables stockées ensemble parce qu'elles ont en commun certaines colonnes et sont souvent utilisées en même temps **[Marlene Theriaut et al, 2002]**. Un cluster sert à améliorer les performances, l'interopérabilité et le passage à l'échelle.

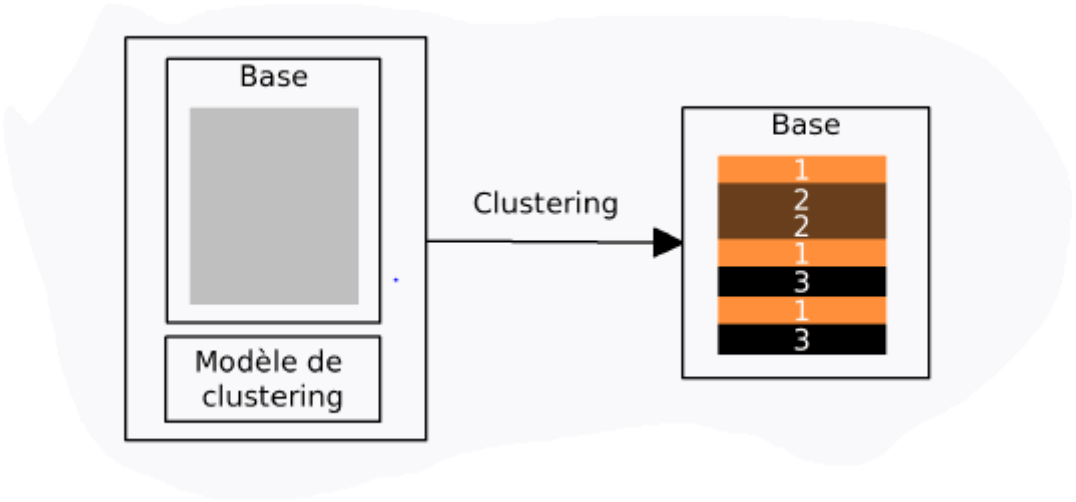


**Figure II.1 : Clustering [4]**

- **Quelques exemples de clustering : [5]**

1. Identification des clients qui ont des profils d'achat similaires.
2. Identification des thèmes dans une collection des donnés (application en recherche d'information).
3. Classification de machines pour facilité la maintenance/dépannage.

- **Qu'est ce qu'un bon clustering? [6] :**

- Une bonne méthode va produire des clusters dont les éléments ont

  -Une forte similarité intra-groupe.

  -Une faible similarité inter-groupe.

- La qualité d'un clustering dépend de la mesure de similarité.
- La qualité d'une méthode peut aussi être mesurée par sa capacité a trouver quelques ou tous les motifs intéressant.

**II.3.2. But de Clustering :[7]**

Le clustering (regroupement) des documents vise à mettre les documents similaires ensemble. En ce faisant, on veut atteindre un des buts suivants:

- Le nombre de clusters, par rapport au nombre de documents, est beaucoup plus petit. Ainsi, on peut accélérer le processus de recherche.

- Si on document est pertinent à une requête, alors les documents similaires ont plus de chance à être pertinents aussi. Ainsi, les clustering peut être aussi vu comme un

moyen d'expansion.

- Finalement, les réponses du système peut être regroupées, plutôt qu'être mises dans une liste individuellement. L'avantage de cette présentation de résultats est que l'utilisateur peut avoir une idée globale des résultats que les systèmes a trouvés assez rapidement.

Avec le progrès rapide sur les matériels d'informatique, le premier avantage semble beaucoup moins important maintenant. Les deux autres restent toujours d'actualité.

### II.3.3. Méthodes de clustering : [7]

- Les méthodes de clustering peuvent être de deux sortes:
  - Hiérarchique
  - non-hiérarchique

Le premier type d'algorithme essaie de créer une hiérarchie des clusters, les documents les plus similaires sont regroupés dans des clusters aux plus bas niveaux, tandis que les documents moins similaires sont regroupés dans des clusters aux plus hauts niveaux.

Selon comment la hiérarchie est créée, ce type d'algorithmes peut encore se diviser en deux: divisif ou agglomératif. En partition, on tente de diviser un grand cluster en 2 plus petits (approche descendante). En regroupement, on tente de regrouper 2 clusters en un plus grand (approche ascendante).

Le deuxième type d'algorithmes ne crée pas une hiérarchie. Les clusters sont au même niveau.

Parmi les algorithmes souvent utilisés, il y a:

**k-means:**

1. Choisir k germes (seeds)
2. attacher les points aux clusters dont le centroïde est le plus proche
3. re-calculer le centroïde
4. répéter jusqu'à ce que les centroïdes se stabilisent.

- **Avantages et faiblisses de k-means** :

- **Avantages :[6]**

- Sa simplicité conceptuelle.

-Sa rapidité et ses faibles exigences en taille mémoire.

-Termine souvent sur un optimum local. L'optimum global peut être atteint en utilisant des techniques telles que les algorithmes génétiques.

- **Faiblisses : [6]**

-Utilisable seulement lorsque la moyenne est définie. Que faire dans le cas des données nominales ?

-Besoin de spécifier K a l'avance.

-Ne gère pas le bruit et les exceptions.

-Ne trouve que des clusters de forme convexe.

**k-Nearest-Neighbors (kNN): [7]**

Dans cet algorithme, un point est d'abord associé à k points voisins les plus proches. Le cluster auquel est associé le plus grand nombre de voisins est le cluster choisi.

**Agglomératif:**

1. calculer la similarité entre les clusters (au début, chaque document forme un cluster tout seul)
2. regrouper les 2 clusters les plus similaires
3. répéter 1-2 jusqu'à obtenir un seul cluster

bisection k-means:

1. déterminer un cluster pour couper en 2
2. couper ce cluster en deux clusters
3. répéter l'étape 2 ITER fois et prendre les clusters coupés dont la similarité globale au sein de ces clusters est la plus élevée
4. répéter 1-3 jusqu'à obtenir k clusters.

- **Caractéristiques des méthodes de clustering :[6]**
  - ✓ Capacité à gérer les différents types d'attributs.
  - ✓ Découverte des clusters avec des formes arbitraires.
  - ✓ Capacité à gérer le bruit et les exceptions.
  - ✓ Indifférent à l'ordre des données en entrée.
  - ✓ Nombre des dimensions.
  - ✓ Incorporation des contraintes par l'utilisateur.
  - ✓ Interprétabilité et utilisabilité.

### II.3.4 Techniques de Clustering des systèmes P2P :

Depuis la popularité des systèmes P2P en début des années 2000, de nombreux travaux visaient à les clusteriser et cela principalement pour assurer les performances et le passage à l'échelle. Plusieurs travaux ont proposé des protocoles d'organisation en cluster des réseaux P2P : Connectivity-based Distributed node Clustering (CDC), SCM-based Distributed Clustering (SDC)**[Y.Li et al, 2006]**,Clustering in P2P exchanges **[S*tevens Le-Blond et al*, 2005]** et les techniques de Clustering basées sur le nombre de sauts. Ces protocoles sont focalisés sur la topologie du réseau, et sa maintenance face à la volatilité des pairs.

**Formalisation du réseau :**

Un réseau de P2P est assimilé à un graphe G= (V, E) non pondéré et non orienté, où V = {V1, V2,…, Vm} est l'ensemble des noeuds du réseau et E est l'ensemble des arêtes. Soit C= {C0, C1, C2,…Cn} un ensemble de Clusters donnés qui correspondent au graphe G. Chaque cluster Ci est un ensemble non vide de nœuds et soit Vi un nœud de V, nous avons la notation suivante :

- Degré (Vi) : l'ensemble des voisins de Vi.
- Clust (Vi) : l'ensemble des noeuds dans le même cluster que Vi.
- FalsePos (Vi) : l'ensemble de non voisins (les noeuds à atteindre de Vi par un certain nombre de sauts supérieur à une valeur fixée) de Vi dans le même cluster que Vi.
- FalseNeg (Vi) : l'ensemble de voisins de Vi mais pas dans le même cluster que Vi.

### II.3.4.1 Clustering basé sur le degré des pairs :

SCM (Scalable Coverage Measure) **[S.Dongen, 1998]** est proposé pour évaluer la précision d'un réseau de clusters. Son idée principale c'est qu'un Clustering optimal d'un graphe doit minimiser le nombre d'arêtes entre les clusters et le nombre des nœuds voisins à l'intérieur de chaque cluster.

La SCM (Scale Coverage Measure) d'un noeud est donnée par la formule suivante :

**SCM(Vi)=1-[(|FalsePos(Vi)|+|FalseNeg(Vi))/|Degré(Vi)|UClust(Vi)|]**Et celle du graphe G est définie comme la moyenne des SCM de tous les noeuds.

**A. Le protocole SDC (SCM-based Distributed Clustering):**

Ce Protocole est définie dans **[Y. Li et al, 2006]**, au départ, chaque noeud Vo est considéré comme orphelin (il forme un cluster qui ne contient que lui-même) avec son identifiant qui est considéré comme identifiant du cluster et la taille du cluster est égale à 1. Pour le calcul dans SCM, chaque noeud Vo maintient deux variables a0 et b0 tel que : a0=|Degré(V0)|+|Clust(V0)| et b0=|FalsePos(V0,C)|+|FalseNeg(V0,C)|. Initialement a0=b0=|Degré(V0)|, on utilisant l'équation 1, SCM=1-(b0/a0).

Quand on exécute le protocole SDC, tous les noeuds commencent par l'échange de messages avec leurs voisins pour former un réseau provisoirement clusterisé qui n'est généralement pas optimisé. La procédure de Clustrering sera exécutée quand le noeud V0 tente de rejoindre un nouveau cluster, cette procédure peut être décrite comme suit :

Premièrement, le noeud V0 cherche les Clusters de son voisinage (clust_Prob message) et les informe de son intention à les rejoindre. Deuxièmement, chaque noeud Vj dans un cluster candidat Ci calcule le gain ΔSCM(Vj) assumant que le noeud V0 va rejoindre le cluster Ci. Ce calcule requière seulement l'information sur le voisinage de V0.

Après la réception des messages reply de tous les noeuds de son cluster et ceux de Ci, le noeud V0 calcule le gain global ΔSCM(G) assumant qu'il quitte son cluster original et rejoint le cluster Ci. Si ΔSCM(G) > 0, alors V0 doit rejoindre le

Cluster Ci (en respectant le diamètre). Il se peut qu'il aura plusieurs clusters candidats pour lesquels ΔSCM(G) est positif, dans ce cas le noeud V0 doit rejoindre celui dont ΔSCM est maximal. Une fois où le noeud V0 détermine le cluster à rejoindre, un message cluster_update est envoyé vers son cluster original et aussi vers son nouveau cluster, cela permet de faire une mise à jour de l'information de l'ancien et nouveau cluster.

Cette technique permet de former des clusters en se basant uniquement sur le voisinage et le non voisinages des noeuds. Selon les expérimentations elle permet de former des clusters qui contiennent que des noeuds voisins. Pour notre cas, cette technique semble lente à s'achever et construit des clusters non cohésifs ce qui complique l'interopérabilité et baisse les performances des requêtes.

**B. Connectivity-based Distributed node Clustering (CDC):**

Dans cette technique **[L. Ramaswamy et al, 2003]**, le Clustering se base sur les poids des messages envoyés à partir des noeuds Originators (noeuds qui débutent l'exécution de l'algorithme CDC) du réseau. Les messages sont Pondérés(initialement basé sur le nombre de voisins de l'originator, MWeight =1/degrée(Vi)), diffusés par inondation, et finalement le processus de décision affecte un noeud Vi au cluster contenant un Originator offrant le poids maximal.

Chaque noeud Vi est associé à un élément de calcul autonome CEi (utile pour le calcul des poids des messages). Un message Mj est caractérisé par les attributs suivant : Originator ID (OID) pour identifier l'originator, Message ID (MID) pour identifier le message, Message Weight (MWeight) c'est le poids du message, Source ID (SourceID) indique le noeud le plus récemment visité et Time to Live (TTL) indique la durée de vie du message (nombre maximal de saut dans notre cas). On représente les originators par l'ensemble O = {O1, O2,…, On}.

**Algorithme CDC :**

L'algorithme CDC (figure 1) commence par l'initialisation des messages à partir d'un ensemble d'originators. Chaque originator envoie ses messages qu'il initialise à tous ses voisins. Les champs SourceID, MID et OID décrivent le message lui même et ils sont simples à initialiser, par contre le champ MWeight est initialiser par un originator Ol par la valeur de la fraction de 1 sur le degré Ol.

**Figure 1: Algorithme exécuté par un noeud Originator Ol :**

Créer un nouveau message Msg

Msg.OID← Ol, Msg.Mweight ← 1/Degrée(Ol).

Msg.SourceID← Ol, Msg.TTL← initialeTTL

Msg.MID←Current System Time {une unique valeur}

**Pour** chaque noeud Vi€Nbr(Ol) **faire**

Envoyer le message à Vi.

***Fin pour.***

Chaque noeud Vi (figure 2) maintient un ensemble de valeurs, représentées sous forme de TotalWeignt(Vi,Ol). Cette valeur indique la somme des poids de tous les messages prévenants de l'originator Ol et reçus par le noeud Vi. A la réception du message Msg de l'originator Ol, le noeud Vi ajoute le poids de ce message à la valeur de TotalWeignt(Vi,Ol) . Puis le noeud Vi vérifie si le TTL du message est supérieur à 0. Si s'est le cas, le noeud Vi envoie le message à tous ses voisins. Avant la diffusion du message, le noeud Vi met à jour le poids du message pour avoir la valeur Msg.MWeight/degréeVi et décrémente de 1 la valeur du TTL. Chaque noeud peut recevoir plusieurs messages à partir de différents originators. Lorsque tous les TTL sont égaux à 0, le noeud Vi rejoint le cluster mené par l'originator dont la valeur de TotalWeignt est maximale. Si toutes les valeurs de TotalWeignt sont inférieures à un seuil prédéfini, le noeud Vi reste isolé.

Il reste maintenant à choisir dans le réseau, les noeuds qui jouent le rôle des originators, afin d'éviter les noeuds isolés et assurer un bon Clustering. Pour cela, les originators doivent être réparties uniformément sur le réseau et le poids total des

messages initiés par chaque originator doit être supérieur à n'importe quel poids total associé à un autre noeud de son voisinage. Malheureusement, la dernière propriété n'est pas satisfaisante, car elle demande l'exécution de l'algorithme CDC. Pour palier à ce problème, un noeud Vl se prend comme originator si la somme des fractions de  sur la multiplication de son degré et le degré de ses voisins (Nbr(Vl) ) est maximale(TwoHopProb).

**TwoHopProb(Vl) =ΣVi∈ Nbr(Vl) 1/ (Degré(Vl)*Degré(Vi))**

***Figure 2: Algorithme exécuté par un noeud Vi à la réception de message***

{Vérifier si on a reçu le message à partir de Msg.OID}

***Si*** on a reçu un message de Msg.OID avant ***alors***

{vérifier si le dernier MsgId(Ol) = =Msg.MID}

***If*** LastMsgId(Ol) = = Msg.messageID ***then***

TotalWeignt(Vi,Ol) ← TotalWeight(Vi,Ol) + Msg.Mweight

***Else***

TotalWeightt(Vi,Ol)←TotalWeight(Vi,Ol)+Msg.Mweight

LastMsgId(Ol) ← Msg.messageId

***Fin si***

***Si non*** {c'et le premier message de Ol}

TotalWeight(Vi,Ol)←Msg.MWeight

LastMsgId(Ol) ← Msg.MId

***Fin si***

***Si*** TotalWeight(Vi,Ol)> MaxWeight ***alors***

MaxWeight ←TotalWeight(Vi,Ol)

MaxWeightId←Msg.OID

**Fin si**.

***Si*** Msg.TTL >0 et (Msg.Mweight/Degree(Vi)) >MinWeight ***alors***

{Créer un nouveau Message NewMsg}

NewMsg.OID←Msg.OID,

NewMsg.SourceID← Vi

NewMsg .Mweight ←Msg.Mweight/Degree(Vi)

NewMsg.TTL←Msg.TTL-1

NewMsg.MID←Msg.MID

***Pour chaque noeud Vj€ Nbr(Ol) faire***

Envoyer le message NewMsg à Vi

***Fin pour.***

Attendre que tous les messages soient reçus.

***Si*** MaxWeight >WeightThreshold ***alors***

Rejoindre le cluster mené par MaxWeightId

***Si non*** rester isoler

***Fin si.***

L'expérimentation montre que la moyenne de SCM est optimal (égale à 1) lorsque le graphe contient seulement des nœuds complètement connexes, si non la valeur est strictement inférieur à 1. Les résultats de l'expérimentation sont comparés principalement aux méthodes de Clustering centrales tel que ''MCL Clustering'' et ''Distributed KPath Clustering schemes''. En sélectionnant les originators d'une façon aléatoire et sans respecter le deuxième principe, la précision du Clustering est approximativement inférieure à **Centralized Clustering et mieux que Distributed KPath.** Lorsqu'on respecte les deux principes de l'algorithme, on remarque qu'il a des résultats mieux que les deux autres algorithmes. Comme critiques, on remarque que l'algorithme n'informe pas quel cluster à rejoindre quand il y en a plusieurs valeurs max, présente certains problèmes de performances et d'interopérabilité des requêtes causés par l'incohérence des Clusters notamment lorsque le TTL prend des grands valeurs. Finalement, il demande maitrise de la valeur de TTL pour éviter la surcharge du réseau et le temps du processus de Clustering. Comme avantage, elle présente l'aspect décentralisé et le passage à l'échelle.

**C. Les techniques de Clustering basées sur le nombre de sauts (cas réseau sans fil)**

Dans ce cas, le Clustering est utilisé comme solution pour combler aux limites du passage à l'échelle des protocoles de routage.

**C.1. Les algorithmes de Clustrering à 1 saut LCA et HCC**

Ces deux algorithmes ont la même heuristique, mais une métrique différente **LCA (Linked Cluster Algorithm)** utilise l'identifiant des noeuds et **HCC (High Connectivity Clustering)** utilise le degré des noeuds. Un noeud possède les trois statuts suivants : Cluster-head (un noeud chef du cluster), Passerelle, noeud ordinaire (état par défaut). Le noeud Vo, se déclare Cluster Head, s'il constate qu'il possédait la métrique la plus forte. Dans le cas où cela se passe dans un noeud voisin, il s'y attache et se déclare noeud ordinaire. Si plusieurs noeuds de ses voisins se sont déclarés cluster-heads, V0 se déclare passerelle.

Ces deux algorithmes engendrent des Clusters recouvrant et rendent la maintenance coûteuse à cause des réactions en chaine. Pour résoudre le premier problème et réduire le deuxième, les algorithmes 3HBCA (3 Hops Between Adjacent Clusters Algorithm.) sont proposés. Cette amélioration est assurée, en empêchant deux cluster-head de s'éloigner à une distance inférieure ou égale à 3. Des solutions proposées plus récemment, utilisent un nombre de saut K> 1 pour assurer le passage à l'échelle (pas beaucoup de clusters). Même si que ces algorithmes assurent le passage à l'échelle, ils demeurent toujours victimes des problèmes précédents.

**C.2. Les algorithmes de Clustering à k sauts Max Min d clusters**

L'objectif essentiel de ces algorithmes, s'est de résoudre le problème de construction des clusters à l'aide d'un algorithme np-complet. Un certain nombre de sous objectif sont encore à mentionner : les noeuds doivent exécuter l'algorithme d'une façon asynchrone, limiter le nombre de messages entre les noeuds, avoir un algorithme qui utilise peu de données, minimiser le nombre de clusters-head en fonction de la distance de d, la distribution de la gestion des clusters doit être faite sur tous les noeuds du réseau (pour avoir l'équitabilité).

Un noeud V0 utilise les max et les min de ses d voisins pour se déclarer chef de Cluster.

**Chaque noeud V0:**

- ✓ Collecte l'identifiant de tous ses d voisins a au plus d-sauts de lui et maintient le maximum des identifiants de ses d voisins (Max_idV0).
- ✓ Diffuse Max_idV0 aux d sauts. (et collecte ceux envoyés par ses d-voisins)
- ✓ Maintient le minimum de ces Max_idV0 (Min_idV0).
- ✓ Diffuse Min_idV0 aux d sauts. (et collecte ceux envoyés par ses dvoisins).

**Choix du cluster-head:**

- ✓ Si V0 voit son identifiant parmi les MIN_ID de ses voisins, il se déclare chef.
- ✓ Sinon, si V0 a vu l'identifiant d'un ou plusieurs noeuds a la fois parmi les MAX_ID et les MIN_ID de ses voisins, le noeud de plus faible ID est élu chef du cluster.
- ✓ Sinon, Vo élit le plus fort identifiant dans son d-voisinage.

Cet algorithme présent l'avantage de fournir un processus de Clustering non recouvrant a d-sauts en utilisant les informations de ses d-voisins. Malgré cela il utilise beaucoup de messages avec une latence importante ce qui peut diminuer leur performance. Comme il se base sur l'identifiant, les chefs restent chefs longtemps (dans le cas où le réseau est stable) mais le processus de Clustering se déclenche à chaque fois qu'un noeud se déplace ce qui provoque la naissance de nouveaux clusters plutôt que d'adaptation des Anciens. Dans le cas de notre contexte, cette approche peut causer des clusters non cohésifs ce qui va augmenter les problèmes d'interopérabilité et de performance des requêtes.

### II.3.4.2 Clustering basé sur la sémantique :

Les Semantic Overlay Networks (SON) **[Arturo Crespo, al, 2002]** consistent à former des clusters dont les noeuds sont sémantiquement proches tout en assurant un degré élevé d'autonomie et le passage à l'échelle.

Parmi les challenges à surmonter pour respecter l'idée principale de la technique de SON on trouve : les difficultés de classification des requêtes, le niveau de granularité de la classification, l'appartenance ou non des noeuds des SONs et le choix des SONs à utiliser pour répondre à des requêtes.

**A. Présentation de Semantic Overlay Networks :**

Soit D l'ensemble des documents à étudier. Chaque noeud Vi∈V maintient un ensemble de documents di (un document particulier peut être stocké dans plusieurs noeuds). Un lien est un triplet (Vi,Vj,l) où Vi et Vj sont les noeuds connectés et l une chaine de caractères. On appelle l'ensemble de liens qui ont le même l, un Overlay Network. Tous les liens sont bidirectionnelles, autrement dit (Vi,Vj,l) est identique à(Vj,Vi,l).

**B. Classification Hierarchique :**

Il existe plusieurs possibilités à utiliser pour avoir une classification hiérarchique (classification par Type/Sous Type, par intervalle de temps, décennie):

**Exemple :** la classification hiérarchique des fichiers Musicaux peut être faite selon les possibilités suivantes : Classification selon le style (rock, jazz, etc.) et le sous style (soft, dance, etc.), Classification par décennie: chansons éditer entre 1990 et 2000 et Classification par sonnerité:(Warm, Exciting,..Sweet) **[Arturo Crespo, al, …]**.

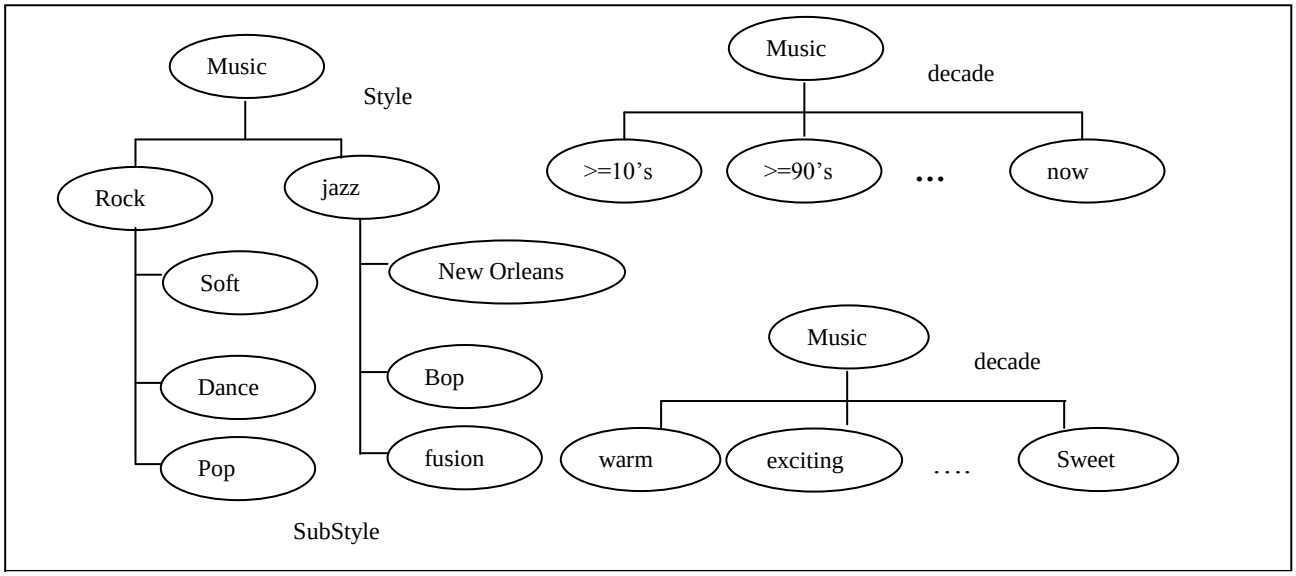


**Figure II.2: classification hiérarchique**

Les fichiers et requêtes sont classifiés en un ou plusieurs concepts dans la hiérarchie (processus pas forcément exacte). On appelle **''bucket de X''** l'ensemble de fichiers qu'on classifie dans le même concept X. cette classification induit à la création des Sématiques Overlay Network X **SONx [Arturo Crespo, al, …].**

Il existe principalement 2 stratégies qui permettent de placer les nœuds dans les SONs. La première est la stratégie **very conservative**, elle place un nœud dans un SONx s'il possède au moins un document classés dans le concept x. la second, est la **Stratégie less conservative**, elle place un nœud dans SONx s'il possède un nombre significatif de documents classés dans le concept x. contrairement à la première, cette stratégie réduit le nombre de nœuds dans le SON, ce qui augmente la rapidité d'exécution et comme aussi elle réduit le coût des SONs du moment où on n'a pas beaucoup de connexions entre les SONs et les nœuds. Comme inconvénient elle souffre du problème de silence des résultats des requêtes.

Le processus de construction et d'utilisation des SON évalue on premier lieu plusieurs classifications hiérarchique potentielles puis il choisit la meilleur. Celle-ci est enregistrée sur tous (ou certain) les nœuds dans le système et elle sera utiliser pour définir les SONs. Un nœud exécute un classificateur de documents basé sur la hiérarchie défini. Après, le classificateur de nœuds assigne le nœud à des SONs spécifiques (par exemple en utilisant la stratégie conservative). Le nœud rejoint chaque SON par une découverte de nœuds qui appartiennent à ses SONs et cela peut être fait soit par l'envoi de message comme le système de Gnutella ou par l'utilisation d'un fichier centrale. Quand un nœud reçoit une requête, il la classifie et l'envoie vers les SONs concernés, après que la requête soit envoyée à des SONs les nœuds concernés essayent de trouver un matching en utilisant des mécanismes appropriés (comme Gnutella Floding ou Super Peers).

Les algorithmes de classification hiérarchique, veillent à assurer les caractéristiques suivantes pour assurer une bonne classification hiérarchique :

- ✓ Produit des buckets dont les documents appartiennent à un petit nombre de nœuds
- ✓ Les nœuds ont des documents liés à un petit nombre de buckets
- ✓ Simplifier et assurer la validité de l'algorithme de classification hiérarchique.

**C. classification des documents et de requêtes :**

Pour les documents, la classification demande de les mettre qu'avec les groupes adéquats et avec des coûts de gestion faible, par contre pour les requêtes, la classification demande de respecter un flux régulier et de veiller à optimiser les requêtes, car elles sont nombreuses.

Sous ces conditions, le classificateur de documents peut utiliser un Algorithme très précis (mais lent) qui peut traiter en lots un grand nombre de documents, tandis que, le classificateur de requêtes peut être implémenté par un algorithme rapide mais qui peut être imprécise.

**D. Stratégie Layered SONs :**

Dans la stratégie Layered SONs, on fixe d'abord le seuil (un pourcentage ou un nombre fixe) à dépasser pour qu'un noeud puisse rejoindre les SONs. Puis pour déterminer quels sont les SONs à rejoindre, le noeud procède comme suit: Premièrement il considère les catégories feuilles dans la hiérarchie, puis il rejoint les SONs de celles qui dépassent le seuil. Deuxièmement, pour les catégories restantes (celles qui n'ont pas rejoint les SONs), le noeud considère le niveau supérieur de la hiérarchie puis il fait la combinaison des descendants restants pour décider s'il rejoint les SONs correspondants à ce niveau. Cette étape se répète jusqu'à ce qu'il n'ya plus de catégorie feuilles restantes ou jusqu'à atteindre la racine.

Pour les requêtes, on procède d'abord à leur traitement pour avoir l'arbre de classification. En utilisant les feuilles de cet arbre, on les envoie vers le SON (ou les SONs) qui a des concepts commun avec elles. Le processus continu de monter

progressivement dans la hiérarchie jusqu'à ce qu'on ait suffisamment de résultats.

Selon les expériences proposées, SON (en particulier Layered SON respecte plus le principe de départ) offre un moyen d'amélioration de l'efficacité des systèmes P2P avec des coûts de développement en plus. Cette technique suppose la connaissance des ontologies (au moins le domaine) a priori pour proposer une technique de hiérarchisation meilleur qui est vital pour assurer un bon Clustering. En plus que ca, l'algorithme de classification (hiérarchisation) est légèrement simple à trouver, si on est dans un seul domaine (Médecine, Enseignement, E-Commerce, etc.) par contre si plusieurs domaines se présentent en même temps, on est dans une situation de demander est ce que on doit proposer autant d'algorithmes que de domaines ou un seul algorithme pour tous les domaines (cas difficile, exemple : Médecine, Musique, e-commerce). Une autre remarque c'est que le passage à l'échelle n'est pas garantie du moment où tous les noeuds qui respectent les conditions d'inclusions au SON doivent se retrouver ensemble. En plus pour une raison ou une autre les documents peuvent contenir des données appartenant à plusieurs SONs qui vont contrarier même le principe de départ. Finalement, dans le cas d'existence de plusieurs sous type, cet algorithme soufre de performance et beaucoup plus celle des requêtes.

**II.4 Etat de l'art de l'Alignement des ontologies** :

**II.4.1 Définition de l'alignement d'ontologie :**

L'alignement de deux ontologies revient à trouver une correspondance entre leurs entités qui sont sémantiquement similaires (Ehrig *et al.* 2004b). D'une façon formelle, l'alignement est défini par la fonction *map* comme suit :

$$map : O \longrightarrow O' \; tel \; que \; map(e_1) = e_1' \quad \text{si } sim(e_1, e_1') > t,$$

Où *O* et *O* 'sont les deux ontologies à aligner, t désigne un seuil minimal de similarité appartenant à l'intervalle [0,1], $e1 \in O$ et $e1 \in O'$. *e*1 et *e*2 représentent les entités au niveau des deux ontologies. Le seuil t indique le niveau minimum pour que deux

entités soient similaires.

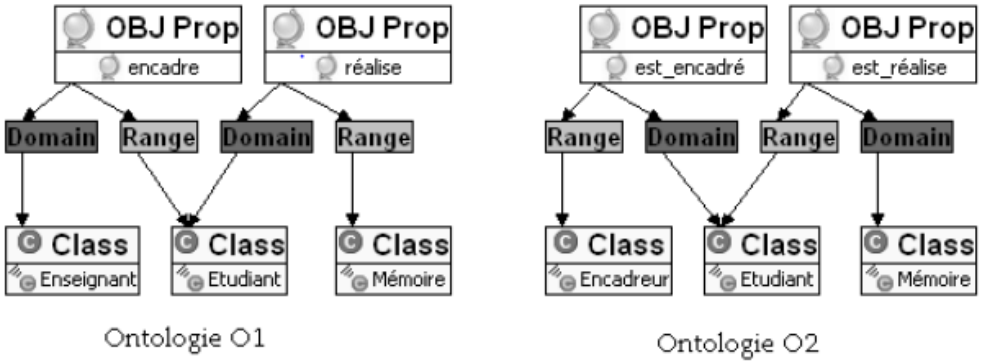


**Figure II.3 : Exemples de connaissances de deux ontologies représentées sous forme de graphes**

La figure II.3 présente deux exemples de représentation de connaissances de deux ontologies. La première ontologie, *O*1, indique qu'un enseignant encadre un étudiant qui réalise son mémoire. La seconde ontologie, *O*2, indique qu'un mémoire est réalisé par un étudiant, qui est encadré par un enseignant. L'alignement des deux ontologies *O*1 et *O*2 revient à déterminer la correspondance entre les différentes entités ontologiques par catégorie.

Le tableau II.1 permet de donner les différentes entités à comparer au niveau des deux ontologies. Toutes les méthodes d'alignement déterminent des correspondances entre les entités ontologiques en utilisant des mesures de similarité (Zghal *et al.*, 2007b).

| **Entité de l'ontologie** $O1$ | **Entité de l'ontologie** $O2$ |
|---|---|
| Encadre | Est_Encadrer, Est_réaliser |
| Réalise | Est_Encadrer, Est_réaliser |
| Enseignant | Encadreur, Étudiant, Mémoire |
| Étudiant | Encadreur, Étudiant, Mémoire |
| Mémoire | Encadreur, Étudiant, Mémoire |

**Tableau II.1 Entités à comparer au niveau des deux ontologies à aligner**

### II.4.2 Objectif de l'alignement des ontologies : [8]

L'objectif du processus d'alignement est de gérer le plus automatiquement possible, des appariements sur des ontologies, consiste à trouver des correspondances entre les connaissances spécifiées dans les deux ontologies, de manière à pouvoir les exploiter conjointement dans le même système (Euzenat et al., 2004). En pratique, il

s'agit d'identifier des concepts (ou des relations) de la première ontologie avec des concepts (ou des relations) de la seconde.

Dans les deux cas, la connexité des deux domaines de connaissance modélisés par les ontologies est requise, sans quoi aucun lien ne peut être établi entre concepts (Kefi et al.,2006). De plus, les formalismes de représentation d'ontologie utilisés doivent être au moins compatibles, ainsi que les paradigmes conceptuels (Furst,2004).

Les méthodes appliquées pour repérer les similarités entre concepts et/ou relations sont (Euzenat et al.,2004) : -les méthodes terminologiques qui comparent les **labels** désignant deux concepts ou deux relations;-les méthodes qui comparent les **propriétés internes** des concepts et relations(attributs des concepts, portée d'une relation, etc..); -les méthodes qui comparent les **propriétés externes** des concepts et relations (subsomptions, relations entre concepts, etc.); -les méthodes qui comparent les **extensions** des concepts et relations; -les méthodes qui comparent la **sémantique** des concepts et relations. Ces méthodes peuvent bien entendu être combinées entre elles. Elles peuvent parfois recourir à des ressources extérieures aux ontologies à aligner.

### II.4.3 Classification des méthodes d'alignement : [8]

Dans la littérature, la classification la plus citée dans les travaux qui traitent le problème de matching est celle proposée par *Rahm* et *Bernstien* . Elle distingue deux approches de matching de schéma de base de données : l'approche individuelle, et l'approche combinée :

a) **L'approche individuelle :** Le système d'alignement utilise un seul algorithme pour exécuter le matching. Les algorithmes sont classés selon les critères suivants :

- **Instances vs. Schéma** : On distingue deux approches, la première est basée sur les instances de données, et la deuxième est basée sur les schémas de données qui incluent souvent les propriétés des éléments de schéma, telles que le nom, le type de données, etc.

- **Elément vs. Structure** : On distingue deux niveaux de granularité de matching ; élément et structure. Dans le niveau élément le matching est réalisé sur les entités des schémas qui sont analysées d'une manière isolées. En revanche, dans le niveau structure le matching est réalisé sur les relations qui relient les entités entre elles.

**- Langage vs. Contrainte :** on distingue les matchers utilisant des approches linguistiques et celles utilisant des contraintes comme les types de données, les clés des relations.

**- Cardinalité** : La cardinalité de matching peut être classée en deux catégories. La première catégorie est le matching direct qui correspond à une cardinalité de matching(1:1). La seconde catégorie est le matching indirect qui couvre les cardinalités de type(1:n), (n:1) et (n:m) qui signifie qu'il peut y avoir n éléments d'un schéma et m éléments d'un autre schéma qui se correspondent.

**- Informations auxiliaires :** plusieurs matchers dépendent non seulement des schémas d'entrée, S1 et S2, de matching mais aussi des informations auxiliaires comme les thésaurus et les dictionnaires (synonymes, hyperonymie, …..).

b) **L'approche combinée :** Le système d'alignement utilise plusieurs algorithmes pour exécuter le matching selon deux approches :

**- L'approche hybride :** consiste en une utilisation séquentielle et multicritère des algorithmes pour exécuter le matching.

**- L'approche composite :** les algorithmes sont utilisés de façon indépendante. Les résultats fournis par ces algorithmes sont combinés a fin d'obtenir le résultat final de matching.

La classification la plus récente est proposée par *Shvaiko* et *Euzenat* où les auteurs se sont intéressés aux approches de matching de schémas/ontologies. La classification a été vue d'une façon descendante en se focalisant sur la granularité des informations d'entrées (niveau élément ou niveau structure). Dans ces niveaux on distingue les techniques syntaxiques, sémantiques et externes. La classification a été lue aussi d'une façon ascendante en se focalisant sur le type des informations d'entrées. Dans le premier niveau on distingue les approches terminologiques, structurelles, extensionnelles et sémantiques. Celles-ci exigent un raisonnement sémantique. Dans le niveau suivant on distingue les techniques basées sur les chaines de caractères ou linguistiques dont les approches terminologiques, et les approches structurelles qui sont divisées en deux techniques : celles qui considèrent la structure interne des entités (nom, type..) et celles qui considèrent les relations entre les entités. Dans les systèmes d'alignement, on peut utiliser des mesures simples ou combinées pour calculer la similarité entre les entités. Ces mesures peuvent être classées selon la nature des entités alignées : des termes, des structures, des instances, ou des modèles théoriques.

### II.4.4 Techniques de distance et d'alignement entre Ontologies :

Il existe plusieurs méthodes d'alignement d'ontologies présentées dans **[Euzenat, 2008]**. Ces méthodes exploitent différents formats de représentation d'ontologies (OWL-DL, OWL-Lite, OWL-Full). L'alignement d'ontologies repose sur le calcul des mesures de similarité. On essaie de décrire dans ce qui suit les différentes mesures de similarité (lexicale, linguistique, structurelle, et sémantique) qu'on peut utiliser pour trouver des alignements entre ontologies.

**Propriétés algébriques des distances**

Une dissimilarité est une fonction réelle positive $\delta$ de deux ontologies qui doit être d'autant plus élevée que les ontologies diffèrent.

**Définition 1(Dissimilarité) :** Soit un ensemble O d'ontologies, une dissimilarité $\delta : O \times O' \rightarrow R$ est une fonction qui associe une valeur réelle à un couple d'ontologies telle que :

$\forall o, o' \in O, \delta(o,o') \geq 0$…….....(positivité)

$\forall o \in O, \delta(o,o)=0$ .....(minimalité)

$\forall o, o' \in O, \delta(o,o')=\delta(o',o)$.....(symétrie)

**Définition 2 (Distance) [Euzenat, 2008] :** Une distance (ou métrique) $\delta : O \times O' \rightarrow R$ est une fonction de dissimilarité définie et satisfaisant l'inégalité triangulaire :

$\forall o, o' \in O, \delta(o, o')=0$ si et seulement si $o=o'$ ..... (Définition)

$\forall o, o', o'' \in O, \delta(o, o') + \delta(o',o'') \geq \delta(o, o'')$……. (inégalité triangulaire)

La sémantique entre ontologie est nécessaire du fait quelles utilisent pas les mêmes mots, même langage, etc. Les opérateurs qu'on peut proposer sont: équivalence (=) et subsomption ($\subseteq$, $\supseteq$).

**Définition 3 (Mesure normalisée) [Euzenat, 2008] :** Une mesure est dite normalisée si elle prend ses valeurs dans l'intervalle réel unitaire [0 1]. Une version normalisée d'une mesure $\delta$ sera notée par $\overline{\delta}$.

**Définition 4 (Alignement simple) [Euzenat, 2008] :** Soient deux ontologies O et O', un alignement simple est un ensemble de correspondances <e, e', r>, telles que :

- $e \in N(o)$ et $e' \in N(o')$ : sont des entités des ensembles nommées des ontologies.
- $r \in \{=, \subseteq, \supseteq\}$.

**II.4.4.1. Distances dans l'espace des ontologies :**

**A. Distances lexicales :**

Une distance entre ontologies peut être calculée à partir des étiquettes apparaissant dans les deux ontologies en utilisant une mesure telle que la distance de Hamming.

**Definition 5 (Dissimilarité de Hamming sur les noms de classe) [Euzenat, 2008] :** Soient o et o' deux ontologies et L(·) une fonction retournant les noms des entités dans une ontologie, la distance de Hamming sur les noms de classe est caractérisée par :

$$\delta hdcn(o,o')=1-[|L(o) \cap L(o')| / |L(o) \cup L(o')|]$$

Cette mesure est très dépendante des langages utilisés. Elle est relativement rapide et donne des bons résultats lorsque les ontologies sont fidèles à la représentation lexicale.

Il est possible d'utiliser des mesures telles que TFIDF (*term frequencyinverse document frequency*) pour mesurer combien une ontologie est pertinente visà-vis d'une autre. L'idée de TF (Term Frequency) est que, plus un terme est fréquent dans une collection plus il est important dans la description du document. Soit le document dj et le terme ti, alors la fréquence de ce terme dans le document est : $tf_{i,j}=n_{i,j}/\Sigma_k n_{k,j}$. Où $n_{i,j}$ est le nombre d'occurrences du terme ti dans dj. Le dénominateur est le nombre d'occurrences de tous les termes dans le document dj.

Par contre l'idée de IDF (Inverse Document Frequency) est que plus qu'un terme est fréquent dans une collection moins il est important dans la description de ce document. Elle consiste à calculer le logarithme de l'inverse de la proportion de documents du corpus qui contiennent le terme : $idf_i=\log(|D| / |\{dj:ti\epsilon dj\}|)$ Où : $|D|$ c'est le nombre total de documents dans le corpus, $|\{dj:ti\epsilon dj\}|$ est le nombre de documents où le terme tj apparait (c'est-à-dire $n_{i,j}\neq 0$)

**B. Mesures structurelles :**

Elle permet de trouver une distance entre ontologies à partir d'une distance entre concepts. Parmi les mesures disponibles pour passer d'une distance entre concepts à une distance entre ontologies on trouve les mesures de lien, la distance de Hausdorff ou des mesures reposant sur un couplage **[Euzenat, 2008]**,Match-Based Similarity . Dans cette partie on ne présente que la mesure **Match-Based Similarity** .

**Mesure de la similarité structurelle « Match-Based Similarity »**

Dans ce cas les noeuds voisins adjacents de l'entité à aligner sont classés par catégories (utilisation **OL-graphs**) pour lesquelles on affecte des poids Mc. Pour effectuer ce calcule, la mesure de similarité ''Match-Based Similarity'' **[Touzani, 2005]** est utilisée.

**$MSim(C,C')=\Sigma(i,i')\epsilon(C,C')Sim(i,i') / Max(|C|, |C'|)$**

Où C, C′ représente deux ensembles de noeuds de même catégorie, sim(i, i′) représente la similarité linguistique des couples (i,i′). Finalement la similarité structurelle est calculée par :

Sims=ΣC,C′ϵVoisinage O,O′Π(C,C′)MSim(C,C′)

Pour des raisons de normalisation ΣΠ(C, C′)=1.

**C. Mesures sémantiques : [Euzenat, 2008]**

On définit ce qui peut caractériser une mesures sémantique par la notion de conséquence (|=) qui certainement une distance fondée sur l'interprétation des ontologies.

**Définition 9 (Distance sémantique) [Euzenat, 2008] :** Soit un ensemble d'ontologies O et une relation de conséquence |= pour la logique dans laquelle ces ontologies sont exprimées, une distance est sémantique si et seulement si

∀o, o′, o′′ ∈O, o|=o′ et o′|=o′′→δ (o, o′) < δ (o, o′′) et δ (o′, o′′)< δ (o, o′′)…….(compatibilité)

∀"o, o′ ∈O, o|=o′ et o′|=o, si et seulement si δ (o, o′)=0……....(définissabilité)

**Definition 10 (Distance sémantique idéale) [Euzenat, 2008] :**

Soient deux ontologies O et O′ et une fonction Cn retournant leurs ensembles de conséquences, la distance sémantique idéale est définie par :

**δis(o,o′)=1-[|Cn(o) ∩ Cn(o′)| / |Cn(o) ∪ Cn(o′)|]**

Les ensembles de conséquences sont habituellement infinis, ce qui rend cette mesure très difficile à calculer. Une alternative classique serait d'utiliser la réduction des ontologies à la place de la clôture. Mais dans le cas général, ces réductions ne sont pas uniques et leur taille peut être variable. Il est donc difficile d'utiliser des mesures fondées sur la cardinalité.

**D. Mesure linguistique**

La **distance de Jaro-Winkler** mesure la similarité entre deux chaînes de caractères. Il s'agit d'une variante proposée en 1999 par William E. Winkler, découlant de la **distance de Jaro.** La distance de Jaro entre chaînes *s*1 et *s*2 est définie par :

**Dj= 1/3(m/|S1| + m/|S2| + m-t/m)**

Où m est le nombre de caractères correspondant et t est le nombre de transpositions.

Deux caractères identiques de S1 et de S2 sont considérés comme correspondant si leur éloignement (différence entre leurs positions dans leurs chaînes respective) ne dépasse pas **:**

**(max (|S1|, |S2|)/2)-1**

La méthode introduite par Winkler utilise un coefficient de préfixe p qui favorise les chaines commençant par un préfixe de longueur l (avec l<=4). En considérant deux chaines S1 et S2, leur distance de Jaro-Winkler est : dw=dj+(lp(1-dj)) Où ; dj est la distance de jaro entre S1 et S2, l est la longueur du préfixe commun (maximum 4 caractères) et p (Winkler propose 0. 1) est un coefficient qui permet de favoriser les chaines avec un préfixe commun.

**II.5 Conclusion et critiques :**

Il n'y a pas de critères universels pour décider si une ontologie est proche ou éloignée d'une autre, mais les diverses mesures proposées peuvent être combinées pour avoir d'une façon réaliste une distance ou une similarité entre ontologies. Vue leurs influences sur la cohésion des Clusters, dans notre contexte, les méthodes de similarité entre ontologies semblent intéressantes et cela dans le chapitre prochain on va de donner notre positionnement et proposer notre solution.

# *Chapitre III :*

## *Positionnement et Solution*

### III.1 Introduction :

Dans un contexte où chaque intervenant du système P2P propose une ontologie pour définir les concepts qu'il manipule, la présence d'un nombre important d'ontologies et l'absence d'une ontologie globale causent par conséquence des problèmes l'interopérabilité, de performance des requêtes et de passage à l'échelle. Pour résoudre cette problématique, il est nécessaire de définir un moyen de communication efficace entre ces dernières. Pour cela, notre méthode utilise un mixage entre les techniques d'alignement des ontologies (lexicale, linguistique, structurel, et sémantique), les techniques de Clustering des systèmes P2P et quelques Design Patterns pour résoudre les problèmes précédents. Au cours de notre travaille, on doit prendre en considération l'aspect dynamique de ces systèmes (connexion et déconnexion fréquentes). Les résultats fournissent par l'expérimentation montrent que l'on peut améliorer d'une façon significative les performances des requêtes, l'interopérabilité des ontologies et le passage à l'échelle.

### III.2. Positionnement :

Du moment où les ontologies sont stockées au niveau de chaque pair, nous parlerons de pair quand il s'agira d'envoi de messages entre les pairs, et d'ontologie

quand il s'agira de similarité entre Ontologies.

Dans notre contexte, chaque intervenant du système P2P stocke une ontologie pour définir les concepts qu'il manipule. En présence d'un nombre important d'ontologies et l'absence d'une ontologie globale, il est difficile d'identifier les pairs qui répondent aux requêtes. Dans un système de P2P traditionnel, les messages circulent à travers tout le réseau pour répondre aux requêtes des utilisateurs,cette technique cause des problèmes d'interopérabilité et comme aussi elle provoque l'apaisement des performances des requêtes à cause de la surcharge du réseau. Pour palier à ces problèmes, il est nécessaire de définir un moyen de communication efficace entre ces dernières. Nous nous intéressons dans notre

contexte à la construction des Clusters de pairs à base de certaines techniques de Clustering des réseaux P2P et de similarité entre les ontologies.

Les liens entre clusters présentent des collaborations pour fournir des résultats observables par les utilisateurs. Malheureusement, cette collaboration se manifeste sous forme de baisse de performances et de problèmes d'interopérabilité.

Pour assurer l'interopérabilité et la performance des requêtes, il est intéressant de veiller à ne pas augmenter le nombre de ces liens et à avoir des clusters qui contiennent que des ontologies fortement proches d'un point de vue lexical, linguistique, structurel et sémantique.

### III.3 Algorithme de Clustering adapté aux ontologies :

#### III.3.1 Définition de similarité :

Pour permettre à notre système le passage à l'échelle, on essaie de fournir

pour chaque pair une vue qui contient un nombre limité de pairs, cette vue est limitée par la distance d qui est considérée comme le nombre de sauts qu'un message d'un pair doit effectuer pour arriver à l'un des derniers pairs inclus dans la vue.

Il est clair qu'un Clustering qui ne prend que les techniques de Clustering des réseaux P2P avec une valeur pour d, va causer les mêmes problèmes que nous avons vu dans la partie positionnement, et cela à l'intérieur des clusters si la valeur de d est grande (plusieurs ontologies, pas forcément proche dans un cluster) ou entre les Clusters si la valeur de d est petite (avoir un grand nombre de Clusters).

Comme aussi il est clair qu'un Clustering qui ne prend on considération que l'aspect lexicale, linguistique, structurelle et sémantique des ontologies peut induire à des problèmes de performances et des pairs isolés (surtout dans la logique des systèmes P2P qui favorise l'hétérogénéité des ontologies) ce qui ennuie le résultat des requêtes, care celle-ci s'exécute à travers les Clusters. Il est clair qu'une bonne solution doit respecter le principe suivant:

''**Un bon Clustering doit assurer un nombre de liens faible entre les Clusters, une forte ressemblance des ontologies qui forment les Clusters, et Chaque pair doit appartenir à**

**tous les Clusters dont il est question d'appartenir pour protéger la pertinence des requêtes**''.

Pour permettre à notre algorithme une meilleure performance, on utilise dans notre cas, la fonction de similarité qui est une fonction réelle positive de deux ontologies qui doit être d'autant plus élevée que les ontologies ressemblent.

Pour décider sur la ressemblance des ontologies, la comparaison doit prendre on considération des relations de chaque concept, car une forte ressemblance entre deux concepts n'implique pas forcément la ressemblance de leurs ontologies. Pour expliquer cette technique, on peut s'appuyer sur l'exemple suivant : soient Enseignant1 (NumE, NomE, AdressE, VolumeHoraireE) et Enseignant2 (NumE, NomE, AdressE, Nombre de communication E) sont fortement similaire, mais si Enseignant1 appartient au groupe (Vacance, Congé, Sanction) et Enseignant2 appartient au groupe (Chercheur, Séminaire, Papier Acceptés) alors les deux ontologies ne méritent pas d'être rassemblées dans le même cluster. Pour assurer cette technique, nous pouvons utiliser les OL-graph.

Pour la similarité lexicale, nous pouvons utiliser une fonction de similarité

déduite de la distance de Haming avec un dictionnaire de synonyme pour remédier au problème de langage utilisé **[Euzenat, 2008]**:

$$\mathbf{Sim(O,O')=|L(O) \cap L(O')| / |L(O) \cup L(O')|}$$

Où L(.) est une fonction qui retourne les noms des relations, classes et des propriétés avec leur synonymes.

La mesure de similarité linguistique des couples d'ontologies O et O' est calculée entre les noms des concepts (les propriétés, les instances et les classes). Le calcul de la similarité linguistique intègre aussi les commentaires et les étiquettes. La similarité linguistique a été calculée par l'intermédiaire des fonctions de JAROWINKLER ou MONGE-ELKAN **[Monge *et al*, 1996]**. La mesure de JAROWINKLER est adaptée pour les chaînes de caractères courtes, telles que celles représentant des noms et des étiquettes **[Monge *et al*, 1996].** Par contre, la mesure MONGE-ELKAN est plus appropriée pour les chaînes longues telles que les commentaires **[Monge *et al*, 1996]**.

## *Figure4 : algorithme de calcul de la similarité linguistique*

// parcours des propriétés de l'ontologie *O*

**pour chaque** *(propriétéi € O)* **faire**

// Parcours des propriétés de l'ontologie *O'*

**Pour chaque** *(propriétéj€ O')* **faire**

**si** *propriétéi*.type== *propriétéj*.type **alors**

*SimL* = *SimL*+ CALCULSIMLING(*Nud*1,*Nud*2,*FonctSL*)/ Max(|O|,|O'|)

Pour calculer la similarité structurelle, nous essayons de calculer la somme des nombres communs des types de données qui existe dans les deux ontologies, et qui nous divisons en suite sur le nombre maximale de propriétés qui définissent les deux ontologies :

**Sim str (O,O')=ΣtypeiTypeCommuni(O,O') / Max(O,O')**

Nous utilisons la mesure de similarité déduite de la distance sémantique idéale de Hamming pour calculer la similarité sémantique entre deux ontologies O et O'. On effet l'ensemble des conséquences logiques Cn des propriétés qui définissent une ontologie, sont enregistrées sous forme d'un vecteur, puis la similarité sémantique est définie par :

**SimSem(O,O')= |Cn(O)∩Cn(O')| / |Cn(O) ∪Cn(O')|**

La similarité globale est une mesure de similarité agrégée des quatre similarités : lexical, linguistique, structurelle et sémantique (similarité partielle). Pour des raisons d'importance d'une mesure par rapport à une autre, nous affectons un poid à chaque type de similarité : Πlex pour la similarité lexicale, Πlin pour la similarité linguistique, Πstr pour la similarité Structurelle et Πsem pour la similarité sémantique. Pour avoir une similarité globale normalisée nous affectons les valeurs de poids de telle façon à avoir : ΣΠi=1. La similarité globale entre les ontologies O et

O' est donnée par :

**Simg(O,O')=ΠlexSimlex(O,O')+Πlinsimlin(O,O')+ΠstrSimstr(O,O')+ΠsemSimsem(O,O')**

Cette mesure est réaliste, car elle prend on considération plusieurs mesures et élimine certaines d'entre elles ou elle les intègre grâce à la manipulation de la valeur des poids (0 pour éliminer une mesure qui ne semble pas réaliste, cas de la mesure sémantique).

**III.3.2. Algorithme de Clustrering :**

Après avoir à présenter la technique utilisée pour calculer la similarité entre deux ontologies, nous essayons dans cette section de présenter l'algorithme qui assure le Clustering du système P2P, l'algorithme proposé est présenté à travers la figure 5.Au niveau de chaque noeud Vj nous avons une table qui a comme structure, les adresses des noeuds Vi avec lesquels il a communiqué par envoie de messages et la similarité (Vi,Vj) qui représente la similarité entre les ontologies O et O' qui sont Logées respectivement sur les nœuds Vi et Vj. Un message est défini par: d: il correspond au nombre de sauts maximum à effectuer (peut être considérer comme le TTL d'un message), idVi: l'identifiant du noeud créateur du message, idMsg: identificateur du message. Chaque noeud Vi joue le rôle d'un créateur et récepteur des messages.

Pour former les clusters, l'algorithme doit regrouper l'ensemble des pairs qui doivent appartenir au même cluster puis va lancer la procédure de description du cluster. Les étapes définis précédemment répondent seulement, si deux pairs doivent appartenir au même Cluster, mais, il ne traite pas le problème de formation des clusters (identifier tous les pairs du mêmes Cluster). La solution consiste à chercher le pair (le représentant du Cluster) dont la valeur de maxq est maximale, puis il effectuera le Clustering. Pour cela, premièrement, il envoie sa valeur de maxq à tous ses d-voisins, puis il attend celle de tous ses d voisins, il se déclare exécuteur de Clustering, s'il constate que sa valeur maxq est supérieure à celle de ses d voisins, s'il y a deux représentants proches d'une distance inférieur ou égale à d alors le nœud du plus faible id cesse sa fonction. Finalement il essaie de rassembler tous les pairs de

ses d-voisin qui ont des valeurs de Simg(Vi,Vj) >q. Pour décrire ce cluster (partie interface intéressante pour les requêtes), il suffit de présenter comme interface, les valeurs des propriétés pour lesquels Simg(Vi,Vj) >=Q.

**Figure 5 : Algorithme de Clustrering des** pair**s**

**Au niveau du noeud créateur Vi: ……...(1)**

Initialiser idVi, idMsg, d, Q,

Envoyer le message à tous ses voisins

**Au niveau de chaque récepteur Vj ……. (2)**

Decrémentation de la valeur de d

Maxq=0 //le nombre le valeur supérieur à Q

Calculer Simg(Vi,Vj)

Enregistrer dans sa table de similarité les valeurs de Simg(Vi,Vj) et idVi

**Si** Simg(Vi,Vj) **alors**

Maxq := Maxq+1 ; //pour savoir le nombre le valeur

//supérieur à Q

**Fin si**

**Si** d>0 **alors**

**Pour chaque noeud Vt € Nbr(Vj) faire**

Envoyer le message à Vt

**Fin pour.**

**Fin si**

**Processus de décision ……..... (3)**

Après avoir tous les d=0, nous essayons de rejoindre un ou plusieurs Clusters

**Si** toutes les Simg(Vi,Vj)<Q **alors** le noeud Vj reste isolé

**Fin si**

**Représentant et constructeur du Cluster Ci:……(4)**

// Celui dont maxq est maximal

Envoyer à tous ses d voisins sa valeur maxq

Recevoir de tous ses d voisins leurs valeurs maxq

S'il constate que son maxq est supérieur à celui de ses d voisins

Il se déclare représentant de cluster

Inclure tous les pairs de ses d voisin dont Simg(Vi,Vj)>Q

Enregistrer son idVi dans **repid** dans chaque pair inclus

//**repid** est l'adresse du représentant du cluster.

**Traitement des noeuds isolés**:

Pour chaque noeud isolé, il envoi des messages à tous ses voisins pour récupérer l'identifiant de leur représentant de Cluster. Il répète ensuite les étapes (1), (2) et (3) mais cette fois, avec les représentants des clusters et sans respecter la contrainte de d. Si toutes ses valeurs de Simg(Vi,Vj) obtenues sont inférieures à Q, il forme un cluster en lui-même. Si non il rejoint le Cluster avec lequel Simg(Vrep, Vj) est maximale.

Vu la dynamique des systèmes P2P, il est nécessaire de définir un moyen de reconnaissance des autres pair d'une façon transparente (dès qu'un pair s'ajoute tous ses voisins seront informés d'une façon transparente). Cette contrainte peut être réalisée par l'utilisation du patron observateur.

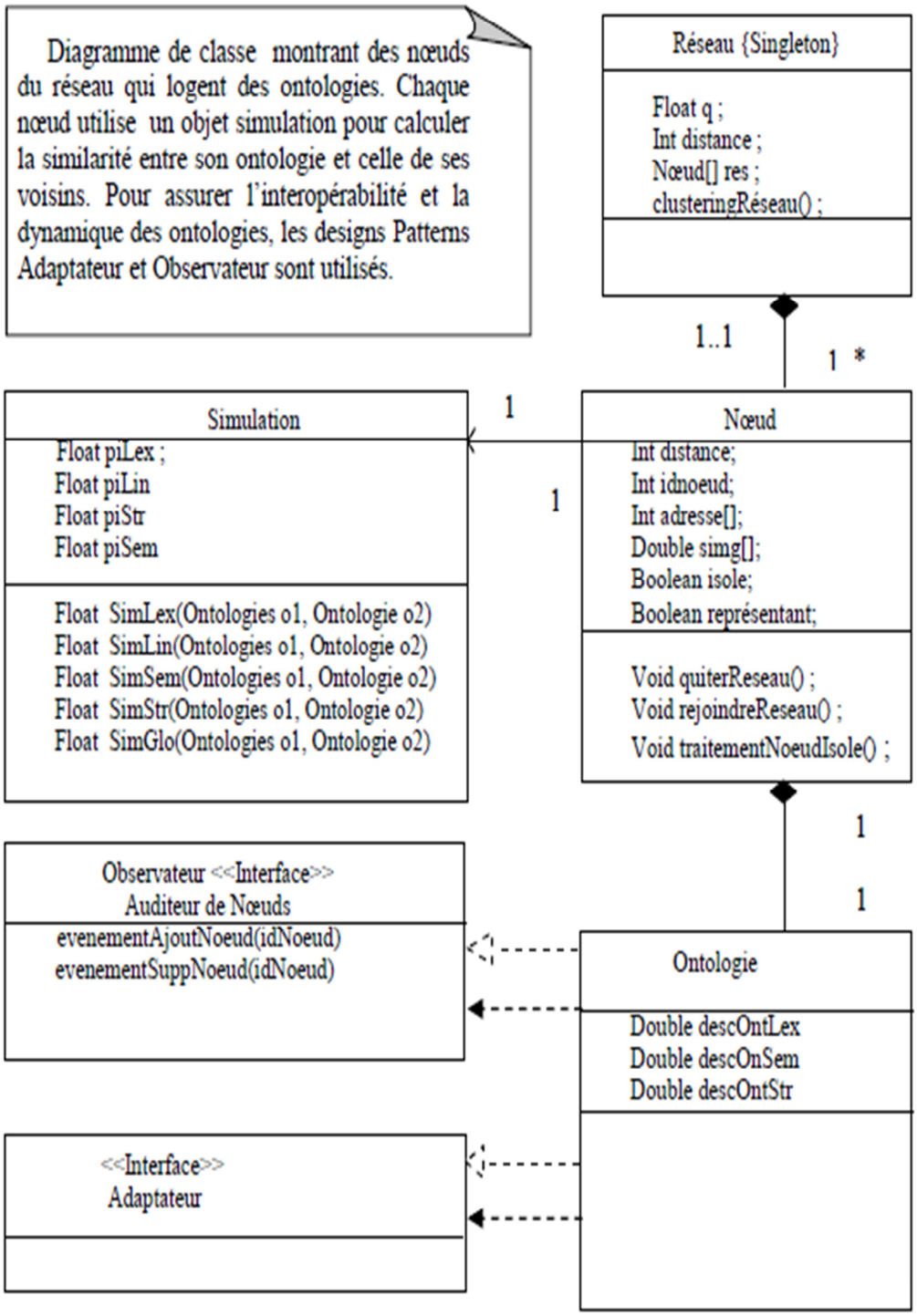


**Figure III.6: Diagramme de classes pour le processus de Clustering.**

Pour le traitement des requêtes, celles-ci sont reçues par les pairs représentants des clusters. Si le représentant estime que son ontologie contient des réponses pour cette requête, alors il la renvoie vers toutes les ontologies de son cluster. Pour les sélectionner, il consulte sa table de similarité (décrite dans la partie (2) de l'algorithme de clustering des pairs) pour récupérer les valeurs de simg(Vi,Vj) tel que simg(Vi,Vj)>q. si c'est le cas, il renvoi la requête au pair concernée après récupération de son adresse (idVi) dans sa table de similarité.

**III.4 Experimentation :**

Pour notre expérimentation, nous avons implémenté le diagramme de classe précédent à l'aide de l'environnement de développement Eclipse. Pour ne pas se préoccuper de la saisie des descriptions de chaque ontologie, nous nous somme contenté d'utiliser les méthodes de saisie automatique. Ces méthodes utilisent la méthode **Math.*random*()** qui génère un nombre aléatoire compris entre 0.0 et 1.0. les méthodes de similarité lexicale, linguistique, structurelle et sémantique s'applique sur les valeurs lues dans les tableaux descOntLex[], descOntLin[], descOntStr[], et descOnSem[] de chaque ontologies.

Pour l'expérimentation, nous essayons de varier la distance d (TTL dans notre cas), le seuil Q qui indique quand il faut regrouper les ontologies et le nombre de noeuds. Vue le non réalisme de la mesure sémantique, et le temps de calcul de la mesure linguistique nous essayons de configurer les poids des mesures comme suit :

PiLex=0.4, PiStr=0.4, PiLin=0.1 et PiSem=0.1. Les valeurs obtenues sur: nombres de Clusters formés, nombre de noeuds non clusterisés et la distance moyenne des nœuds isolés après intégration aux Cluster adéquats sont représentées dans le tableau ci-dessous.

| *Distance (TTL)* | ***Seuil Q*** | *Nombre de Noeuds du réseau* | *Nombre de clusters formés* | ***Nombre de noeud isolés*** | *Distance moyenne après intégration des nœuds isolés à des clusters adéquats.* |
|---|---|---|---|---|---|
| **5** | **0.5** | **1000** | **134** | **0** | **0** |
| **6** | **0.5** | **1000** | **114** | **0** | **0** |
| **7** | **0.5** | **1000** | **104** | **0** | |
| **5** | **0.5** | **2000** | **263** | **1** | **6** |
| **5** | **0.6** | **2000** | **261** | **15** | **7.8** |
| **6** | **0.6** | **2000** | **233** | **7** | **9.142** |
| **7** | **0.6** | **2000** | **205** | **3** | **10.0** |
| **8** | **0.6** | **2000** | **187** | **1** | **14.0** |
| **9** | **0.6** | **2000** | **172** | **0** | **0** |
| **10** | **0.6** | **2000** | **156** | **0** | **0** |
| **5** | **0.7** | **55000** | **6628** | **15226** | **13.195** |
| **6** | **0.7** | **55000** | **6033** | **12148** | **14.191** |
| **7** | **0.7** | **55000** | **5520** | **9170** | **14.952** |
| **10** | **0.7** | **50000** | **3923** | **4090** | **18.015** |
| **20** | **0.7** | **40000** | **17642** | **321** | **28.14** |
| **30** | **0.7** | **40000** | **1223** | **46** | **36.956** |
| **5** | **0.8** | **100** | **5** | **3** | **14.614** |
| **6** | **0.8** | **100** | **5** | **2** | **15.343** |
| **20** | **0.8** | **400** | **9** | **372** | **29.408** |
| **30** | **0.8** | **400** | **10** | **346** | **37.300** |
| **10** | **0.9** | **400** | **0** | **400** | **0** |
| **10** | **0.9** | **400** | **3** | **300** | 11 **(prendre on considération que la mesure lexicale)** |

***Tableau III.2 : représentation du changement du nombre de clusters, de nœuds isolés et de leurs distances moyenne après intégration des clusters adéquats.***

**A. Explication des résultats :**

Nous remarquons que, le nombre de clusters formés est inversement proportionnel à la valeur de TTL et le Seuil Q. Le nombre de clusters isolés converge vers une valeur nulle quand le seuil Q est proche de la valeur de 0.5. Nous commençons à avoir un Clustering non efficace (peu de clusters et beaucoup des noeuds isolés), quand la valeur du seuil Q est proche de 1, cela peut s'expliquer par la difficulté d'avoir des ontologies qui sont presque identiques.

**III.5 Conclusion et perspectives :**

Dans ce rapport, nous avons présenté une nouvelle méthode de Clustering des réseaux P2P en exploitant les techniques d'alignement des ontologies et les techniques de Clustering des réseaux P2P.

Pour les techniques d'alignement des ontologies, nous avons utilisé les quatres approches existantes (lexicale, linguistique, structurel, et sémantique) pour renforcer au maximum l'alignement des ontologies.

Pour le Clustering des systèmes P2P nous avons utilisé un algorithme qui s'appuie sur l'envoi des messages et une distance d pour assurer le passage à l'échelle.

Les résultats obtenus montrent que nous pouvons maîtriser le nombre de liens, la cohésion d'un cluster et les noeuds isolés par la configuration de la valeur de d et de Q. Notre approche fonctionne principalement pour tous les cas d'ontologies et cela par la configuration des poids des mesures de similarité entre ontologies, la valeur de Q et de d. La flexibilité de la dynamique des pairs est assurée grâce au Patron Observateur. La pertinence des clusters (fortement cohésive) dépend de la valeur de Q et des types d'ontologies à clusteriser.

La contribution principale réside dans la proposition d'une approche qui prend en considération les techniques de Clustering des réseaux P2P et les techniques de similarité entre ontologies tout en assurant des meilleurs performances,l'interopérabilité des pairs et le passage à l'échelle.

Notre approche souffre de la pauvreté voir même la stérilité de la similarité sémantique, pour cela nous proposons d'affecter des poids faibles (voire même nul) à cette distance. En termes de perspectives, nous souhaitons de maitre en œuvre notre approche sur un système de P2P réel.

# *Bibliographie*

# Bibliographie :

**[Neches,1991]:** Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., Swartout, W.R.:Enabling Technology for Knowledge Sharing. Dans AI Magazine, pp. 36-56,(1991).

**[Gomez-Perez,1999] :** Gomez-Perez, A.: Ontological Engineering: A state of the art. Expert Update,2(3), 33-43, (1999).

**[Euzenat et al, 2004] :** Euzenat, J. et P. Valtchev (2004). Similarity-based ontology alignment in OWL-lite. In Proc. 16th European Conference on Artificial Intelligence (ECAI), Valencia(ES), pp. 333–337.

**[Euzenat, 2008]** : Euzenat. Quelques pistes pour une distance entre ontologies 8èmes Journées Francophones Extraction et Gestion des Connaissances Sophia Antipolis 29 janvier 2008 pp51-55.

**[Bach, 2004]:** Bach, T. L., R. Dieng-Kuntz, et F. Gandon (2004). On ontology matching problems – for building a corporate semantic web in a multi-communities organization. InICEIS (4), pp.236–243.

**[Gruber, 1993]:** Gruber, T.R. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing, In Formal Ontology in Conceptual Analysis and Knowledge Representation, edited by Nicola Guarino and Roberto Poli, Kluwer Academic Publishers.

**[Y. Li et al, 2006]**: Y. Li, L. Lao, and J.-H. Cui. Sdc: A distributed clustering protocol for peer-to-peer networks. In Networking, pages 1234–1239, 2006.

**[L. Ramaswamy et al, 2003]:**L.Ramaswamy, B.Gedik, and L.Liu.A distributed approach to node clustering in decentralized peer-to-peer networks. IEEE Transactions on Parallel and Distributed Systems, 16(9):814–829, 2005.

**[S. van Dongen, 1998]**: S. van Dongen. A New Cluster Algorithm for Graphs. In 281, page 42. Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-3681, 31 1998.

**[Guy Pujolle, 2004]** : Cours réseaux et télécoms, 2002 edition EYROLLES

**[Marlene Theriaut et al, 2002]** : Sécurité sous Oracle,2002, Edition Oracle Press

**[Craig Larman, 2002] :** UML et les Design Patterns, 2002, Craig Larman, Edition Campus Press.

**[Monge et al, 1996]:** Monge A., Elkan C., « The field-matching problem : algorithm and applications », Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, p. 267-270,1996.

**[Stevens Le-Blond et al, 2005] :** Stevens Le-Blond **1**, Jean-Loup Guillaume **1**, Matthieu Latapy **1**Clustering in P2P exchanges and consequences on performances. 4th International Workshop on Peer-To-Peer Systems (IPTPS'05), Ithaca, NY : États-Unis d'Amérique (2005).

**[Arturo Crespo, al, 2004]** : CRESPO Arturo (1) ; GARCIA-MOLINA Hector (1) Semantic overlay networks for P2P systems, Third international workshop, AP2PC 2004, New York NY,USA, July 19, 2004 ) ( revised and invited papers )

## Web sites:

**[1]** : membres-liglab.imag.fr/donsez/ujf/ricm3/ea/.../peertopeer.htm.

**[2]** : www-adele.imag.fr/~**donsez**/cours/**p2p**.pd

**[3]** : www.scribd.com/doc/55803288/8/Le-**futur-des-reseaux-P2P**

**[4] :** www.lgi2a.free.fr

**[5]:** www-clips.imag.fr

**[6]:** www-lmgm.biotoul.fr/enseignements/M2Pro_Bioinfo/**Clustering.pdf**

**[7] :** www.iro.umontreal.ca/~nie/IFT6255/**Clustering.pdf**

**[8]** *:* asso-aria.org/coria/2008/401.pdf