Routage sémantique des requêtes dans les systèmes pair-à-pair

Taoufik YEFERNY[*] – Khedija AROUR[**] – Yahya SLIMANI[*]

[*] *Faculté des Sciences de Tunis*
Campus Universitaire, Tunis 1060, Tunisie
yeferny.taoufik@gmail.com – yahya.slimani@fst.rnu.tn
[**] *Institut National des Sciences Appliquées et de Technologie de Tunis, 1080 Tunisie*
Khedija.arour@issatm.rnu.tn

RÉSUMÉ. *Les systèmes pair-à-pair (P2P) se sont imposés ces dernières années comme la technologie majeure d'accès à différentes ressources sur Internet. De nombreuses recherche concernant la sélection des meilleurs pairs contenant les données appropriées à une requête,ont émergé et constituent un axe de recherche très actif. L'efficacité de la recherche dans ces systèmes, et surtout le cas non structuré, peut être améliorée en introduisant de la sémantique dans le processus de routage des requêtes. Cette sémantique est généralement construite à partir du contenu des pairs, mais peut également faire intervenir le comportement explicite des utilisateurs. Nous présentons dans cet article un nouvel algorithme de routage des requêtes par apprentissage basé sur le comportement implicite des utilisateurs qui est déduit à partir d'un historique de requêtes. Pour tester l'algorithme proposé, nous avons défini une couche de routage sur le simulateur PeerSim qui nous a permis d'évaluer l'efficacité de notre algorithme.*

ABSTRACT. *The Peer-to-Peer systems (P2P) were led these last years as the major technology of access upon various resources on Internet. The efficiency of the researches in these systems, and especially the not structured case, can be improved by introducing of semantics into the process of routing of the requests. This semantics is generally built from the contents of the peers, but can also bring in the explicit behavior of the users. We present in this paper a new algorithm of routing of the requests by learning on the implicit behavior of the users which is deducted from a history of requests. To test the proposed algorithm, we defined a layer of routing on the simulator PeerSim.*

MOTS-CLÉS : *Systèmes P2P, Routage sémantique, Recherche d'information, Apprentissage, Simulateur PeerSim.*

KEYWORDS: *P2P, Semantic routing, Information retrieval, Learning, PeerSim Simulator.*

Taoufik Yeferny

1. Introduction

Ces dernières années les systèmes P2P sont devenus très populaires car ils offrent la possibilité aux utilisateurs de partager et d'accéder à des ressources diverses, distribuées à large échelle. Il existe de nombreuses architectures des systèmes P2P, qui utilisent différentes techniques de localisation des données, qui se traduisent par différentes méthodes de routage des requêtes (Defude, 2007). Nous pouvons classifier ces systèmes selon leur modèle de recherche sous-jacent, qui peut être soit non structuré (propagation aléatoire des requêtes dans le graphe des pairs), soit structuré (propagation des requêtes selon une structure d'organisation des pairs basée en général sur des fonctions de hachage). Chaque type de système possède des avantages et des inconvenants. Parmi les avantages des systèmes non structurés, nous pouvons signaler le fait qu'ils respectent au mieux l'autonomie des pairs et ils supportent des langages de requêtes plus expressifs. Cependant, ils ne sont pas efficaces sur le plan du routage des requêtes. Plusieurs travaux de recherche ont tenté d'améliorer la méthode de routage des requêtes dans ces systèmes en introduisant la sémantique dans le processus de propagation de requête (Christoph *et al.,* 2004, Defude, 2007). Dans cet article, nous allons présenter un algorithme de routage sémantique de requêtes dans les systèmes P2P qui :

– Sélectionne les pairs les "plus pertinents" pour répondre à une requête en se basant sur le profil utilisateur.

– Propage la requête à ces pairs.

– Génère périodiquement les profils, en mettant à jour une base de connaissances.

Le reste de l'article est organisé comme suit. Dans la section 2, nous présentons un état de l'art sur le routage des requêtes dans les systèmes P2P. Une synthèse sur les méthodes de routage actuelles fera l'objet de la section 3 . La section 4 définit notre algorithme de routage sémantique de requêtes. Les résultats des différentes experimentations de notre algorithme seront discutés dans la section 5. Enfin, nous concluons cet article par une conclusion et quelques perspectives intéressantes de prolongement de notre travail.

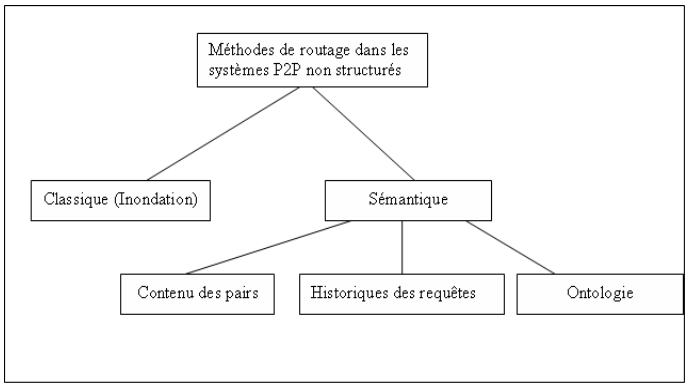

Figure 1. *Classification des méthodes de routage*

2. Etat de l'art sur le routage des requêtes dans les systèmes P2P

La solution idéale pour le routage des requêtes dans les systèmes P2P, consiste à ce qu'une requête, formulée par l'utilisateur, soit automatiquement envoyée vers l'ensemble des pairs susceptibles de fournir une réponse. Dans la littérature, plusieurs travaux de recherche ont essayé d'améliorer la méthode classique de routage des requêtes, qui propage une requête à un ensemble de pairs choisit d'une manière aléatoire, en introduisant la sémantique dans le processus de propagation de requêtes (Christoph *et al.,* 2004, Defude, 2007). Une première approche de cette méthode est appelée système de routage par contenu. Le contenu de chaque pair est résumé dans une étiquette de contenu et un routeur fait suivre les requêtes de recherche selon les meta-informations des pairs qui se sont enregistrés chez lui . Une autre approche consiste à utiliser la sémantique associée aux pairs pour router les requêtes. La figure 1 présente une classification des différentes méthodes de routage dans les systèmes P2P non structurés. Parmi celles qui sont basées sur le contenu de pairs, nous pouvons citer les approches CORI (Collection Retrievel Inference Network) (Callan *et al.,* 1995) et gGLOSS (generalized Glossary of Server's Server) (Luis *et al.,* 1994, Luis *et al.,* 1995) qui représentent la collection de chaque pair voisin, par un document de taille assez grande (superdocument). L'ensemble de tous les superdocuments forme une collection spéciale, utilisée pour calculer un score de classement de collections de chaque pair en fonction des termes de la requête. Par la suite, la requête est propagée aux voisins ayant le plus haut score. Le système de recherche d'information, PlanetP (Raja *et al.,* 2006), représente le contenu de chaque pair de manière compacte à l'aide d'un filtre de Bloom décrivant les termes des documents stockés par celui-ci. Ces filtres de Bloom sont distribués dans le réseau en utilisant un algorithme de propagation. Chaque pair possède un index global constitué d'une liste d'autres pairs, associés chacun à leur filtre de Bloom (Raja *et al.,* 2006), permettant de donner au pair une vision partielle et approchée du contenu global du réseau. Un noeud qui reçoit une requête commence d'abord par faire une recherche dans son index local. S'il ne peut pas hono-

rer la requête, il calcule les rangs des pairs de son index global et il propage la requête aux pairs de plus grand rang.
Peu de méthodes utilisent l'information sur l'historiques des requêtes. Néanmoins, la méthode REMINDIN (Christoph *et al.,* 2004) exploite les métaphores sociales pour définir une stratégie de routage des requêtes basée sur l'historique des requêtes. Pour cela, elle utilise une ontologie (un dépôt local). Dans l'implémentation de REMINDIN, ce sont des déclarations RDF construites à partir des réponses aux requêtes passées. Le choix des pairs est basé sur des observations de la connaissance des autres pairs. En effet, l'évaluation de requête, en utilisant le dépôt local associé à un noeud renvoie un ensemble de pairs triés selon les valeurs de confiance dans des ressources spécifiques pour chaque pair et des valeurs de confiance globales pour chaque pair. Si le nombre de pairs choisis est insuffisant (inférieur à un seuil), la méthode REMINDIN rafraîchie la requête et répéterait le même processus plusieurs fois.

3. Synthèse sur les méthodes de routage

La difficulté de fournir l'accès au contenu des pairs réside essentiellement dans le problème d'échelle. En effet, pour pouvoir trouver les pairs pertinents à une requête, il faut disposer de l'information sur le contenu de tous les pairs. Dans ce cas, il est nécessaire de construire un index global qui serait consulté pour le routage des requêtes. Cette solution peut être viable à petite échelle, mais elle ne s'adapte pas à la taille du réseau pair à pair. L'espace de stockage nécessaire pour l'index est également trop important et sa mise à jour est difficile. Une autre solution consisterait à envoyer chaque requête à tous les pairs (par inondation). Même si cette méthode assure que tous les pairs pertinents seront trouvés, elle implique trop de trafic sur le réseau et une surcharge de travail pour les pairs. En fait, il faut une solution intermédiaire qui permettrait d'acheminer une requête à un ensemble réduit de pairs et ceci sans accéder à un index global. Un autre problème est celui du nombre de réponses qui peuvent être obtenues à la suite d'une recherche sur un nombre important de pairs. Si l'utilisateur reçoit trop de réponses à sa question, il ne pourra pas les bien exploiter. Pour éviter cela, il faudrait imaginer, un moyen de la découverte progressive de l'espace d'information et la spécialisation des requêtes. Un problème important se pose pour les approches sociales (comme REMINDIN par exemple) qui suppose avoir une ontologie partagée par l'ensemble des pairs, ce qui n'est pas toujours disponible pour la communauté des systèmes P2P. De même, l'autre problème qui se pose est que le processus de routage est itératif : plusieurs formulation de la requête peuvent être effectuées (Chernov *et al.,* 2005).
Le tableau 1 compare les différentes méthodes étudiées selon différents critères, à savoir :

– *Information utilisée* : informations utilisées pour définir la sémantique.

– *Représentation de l'information sémantique* : structures de données utilisées pour stocker les informations.

– *Passage à l'échelle* : par passage à l'échelle, nous voulons savoir si la méthode permet de supporter un grand nombre de pairs (i.e + +), un nombre moyen (i.e + -) ou peu (i.e - -).

– *Maintenance* : fréquence des mises à jour des informations sémantiques. La fréquence peut être importante (i.e + +) ou plus au moins importante (i.e + -).

– *Espace de stockage utilisé* : espace mémoire utilisé pour stocker les informations sémantiques.

– *Partage de connaissances* : coopération des pairs pour construire des connaissances. La coopération peut être forte (i.e + +) ou sans coopération (i.e - -).

Critères / Méthodes	gGLOSS, CORI	PlanetP	REMINDIN
Information utilisée	Contenu des pairs voisins	Contenu des pairs	Historique des requêtes+ Ontologie
Représentation de l'information sémantique	Superdocument	Filtre de Bloom	Ensemble de déclarations RDF
Passage à l'échelle	- -	+ -	+ +
Espace de stockage utilisé	+ +	+ +	+ +
Partage de connaissances	- -	+ +	- -
Maintenance	+ +	+ -	+ +

Tableau 1. *Tableau comparatif des différentes méthodes de routage*

En se qui nous concerne, nous proposons une solution basée sur le modèle de routage sémantique. Le système de routage sémantique est une amélioration des méthodes classiques de routage en utilisant l'historique des requêtes. En plus des documents, notre méthode de recherche supporte les nouvelles connaissances que nous avons désigné par le terme profil. Un profil est une corrélation entre les requêtes passées et les pairs positifs ou les requêtes passées et les termes associés. Par pair positif, nous entendons les pairs qui ont agi positivement vis-à-vis d'une requête (il y a eu téléchargements des documents). Ainsi, nous pouvons dire dire que notre approche est dérivée des approches sociales, dans le sens où une requête sera routée vers les pairs partageant des connaissances communes.

4. Modèle proposé

4.1. *Introduction*

L'idée sous-jacente à notre proposition est de substituer le routage classique, basé sur la propagation par inondation itulisé dans Gnutella (Gnutella, 2007), par un routage sémantique du meilleur contenu basé sur un ensemble de profils. La solution que nous proposons supporte le facteur d'échelle en organisant les routeurs par pair et en préparant les profils d'une manière hors ligne.

4.2. *Architecture globale*

Dans notre proposition nous avons repris l'architecture du système Gnutella (Gnutella, 2007) qui est le système le plus représentatif des systèmes de recherche d'information P2P non structurés. Gnutella est un système de partage de fichiers sur Internet qui représente aujourd'hui l'archétype des systèmes P2P non structurés. Dans ce système, une requête est d'abord évaluée sur le pair d'émission puis propagée récursivement sur un sous-ensemble aléatoire des pairs voisins (principe d'inondation). La terminaison de la recherche est assurée par une détection de cycles ainsi qu'une profondeur maximale de recherche (appelée TTL ou Time To Live).
Dans le but d'améliorer l'efficacité de cette méthode, nous avons remplacé le module de propagation des requêtes par un autre module de propagation " guidée " qui utilise une base de connaissances afin de router les requêtes vers les pairs "pertinents". De plus nous avons ajouté deux autres modules, à savoir :

– Un module de gestion de profils qui s'exécute périodiquement. Son rôle consiste à construire la base de connaissances à partir des informations extraites de l'historique des requêtes. Cette historique se trouve dans un fichier log.

– Un module gestion du fichier log qui s'exécute à la réception d'une réponse.

La figure 2 illustre l'architecture globale de notre approche.

4.3. *Module gestion de fichiers logs*

A la réception d'une réponse, ce module met à jour le fichier log en ajoutant les informations relatives à cette requête, à savoir l'identifiant de la requête, l'ensemble de ses termes, les documents téléchargés et les pairs associés.

4.4. *Module de gestion de profils*

L'objectif de ce module est de générer un ensemble de profils qui représentent des corrélations sémantiques entre les requêtes passées et les pairs positifs. La génération de cet ensemble de profils se fait hors ligne au niveau de chaque pair. Pour cela, nous

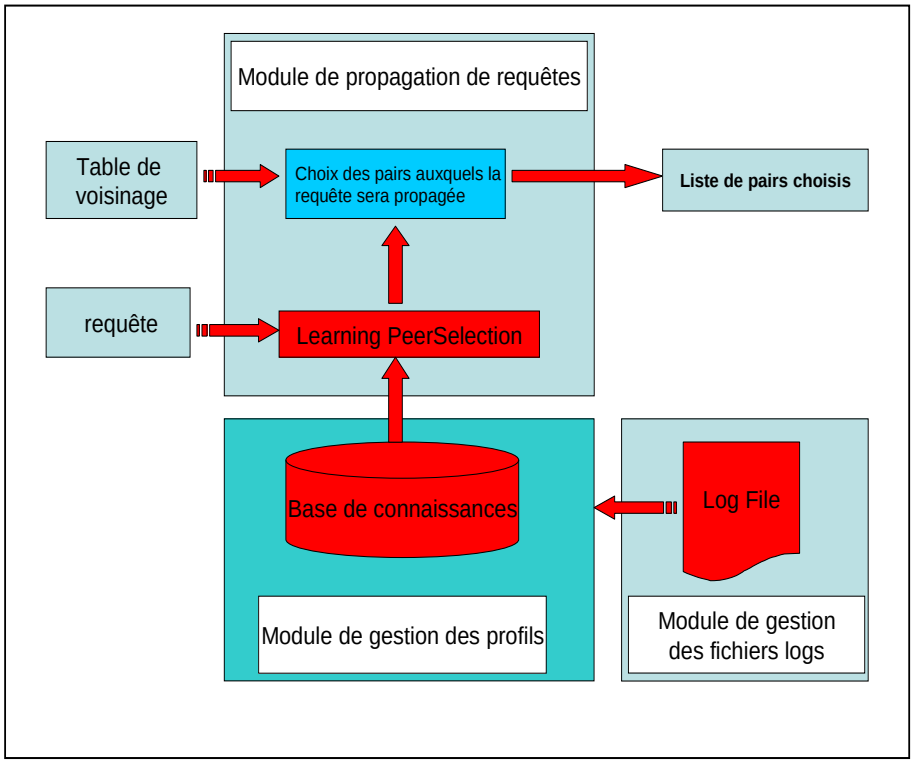

Figure 2. *Architecture globale de notre approche*

avons utilisé une approche formelle de construction de ces profils. Cette approche est basée sur l'Analyse des Concepts Formels.

– *l'Analyse Formelle de Concepts* utilise comme point de départ un « contexte formel » qui définit une relation binaire reliant les objets à leurs attributs (Ganter *et al.,* 1997) .

– *Un contexte formel* : est un triplet $K = (O, A, R)$, où O et A sont, respectivement, l'ensemble des objets et des attributs (caractéristiques), et $R \subseteq O \times A$ est une relation exprimant que $\forall(o, a) \in R$, a est un attribut de l'objet o.
Nous notons par f et g deux applications caractéristiques de R définies comme suit :

$f : P(O) \rightarrow P(A)$
$X \rightarrow \{y \in A | \forall x \in X, (x, y) \in R\}$
$g : P(A) \rightarrow P(O)$
$Y \rightarrow \{x \in O | \forall y \in Y, (x, y) \in R\}$

Un concept associe un ensemble maximal d'objets à l'ensemble d'attributs que ces objets partagent.

– Un concept est une paire $C = (X, Y)$ où $X \subseteq O, Y \subseteq A$

et

$X = \{o \in O | \forall y \in Y, (o, y) \in I\}$ est l'extension d C (objets couverts), notée $Ext(C)$

$Y = \{a \in A | \forall x \in X, (x, a) \in I\}$ est l'intension C (attributs partagés), notée $Int(C)$

De plus, nous avons la propriété suivante, pour tout concept $C = (X, Y), Y = f(X)$ et $X = g(Y)$.

Dans notre cas, nous nous sommes basés sur deux contextes qui sont des projections sur le fichier log. La première projection représente le lien entre les requêtes passées et les termes associés, appelé contexte $C1$. La seconde représente le lien entre les requêtes passées et les pairs positifs (i.e les pairs à partir desquels y a eu des téléchargements), appelé contexte $C2$. En effet, les attributs du contexte $C1$, respectivement de $C2$, sont les termes des requêtes et les pairs ayant répondu à ces requêtes. Un algorithme de génération de concepts formels est par la suite appliqué pour générer deux ensembles de concepts, notés $E1$ et $E2$. Pour ce faire, nous avons utilisé l'algorithme de Godin (Godin *et al.,* 1995) implémenté dans la plate-forme Galicia V3 (Valtchev *et al.,* 2003). Les concepts de l'ensemble E_1, respectivement de E_2, seront sous la forme suivante $(\{R_1, .., R_i\}, \{T_1, .., T_k\})$, respectivement, $(\{R_1, .., R_i\}, \{P_1, .., P_j\})$ avec $\{R_1, .., R_i\}$ un ensemble d'identifiants de requêtes, $\{T_1, .., T_k\}$ un ensemble de termes et $\{P_1, .., P_j\}$ un ensemble de pairs. Ces ensembles constituent une base $B(E_1, E_2)$ qui servira par la suite comme une base de connaissances pour l'algorithme de sélection des pairs. L'exemple de la figure 3représente les étapes de construction des concepts formels à partir du fichier log.

4.5. *Module de propagation*

A la réception d'une requête, ce module fait appel à un algorithme de sélection des pairs (LearningPeerSelection ou LPS) qui retourne un ensemble de pairs "pertinents" pour répondre à la requête. Si le nombre de pairs choisi par l'algorithme LPS est inférieur à un certain seuil nous rajoutons aux pairs sélectionnés un ensemble de pairs choisis de manière aléatoire parmi les pairs voisins (procédure $AjouterAleatoire()$ dans l'algorithme 1).

4.5.1. *Modélisation*

Le diagramme d'activité de la figure 4 modélise le processus de propagation des requêtes.

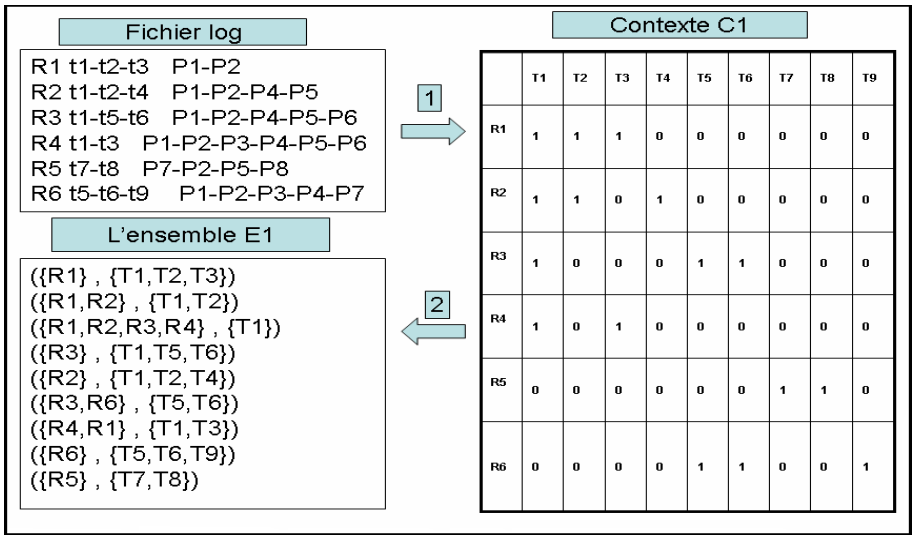

Figure 3. *Les étapes de construction des concepts formels à partir du fichier log*

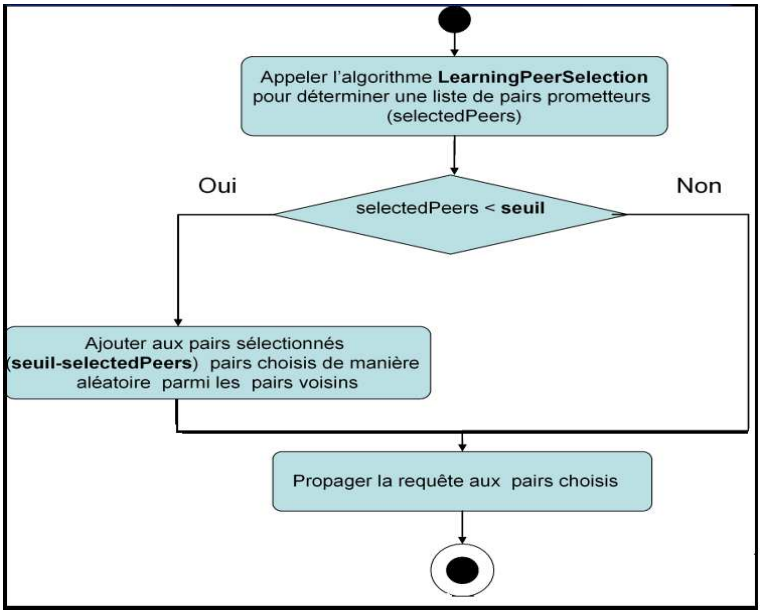

Figure 4. *Diagramme d'activité de l'algorithme de propagation*

4.5.2. *Algorithme LPS : LearningPeersSelection*

L'objectif de cet algorithme est de générer l'ensemble de concepts similaires pour une requête donnée. A partir de ces concepts, l'ensemble des pairs vers qui la requête sera routée est généré.

4.5.3. *Calcul de la similarité*

La génération des concepts similaires au concept de notre requête peut être considérée selon deux points de vue :

– Du point de vue système par exemple en utilisant le modèle vectoriel (Salton, 1989).

– Du point de vue sémantique en utilisant les propriétés communes entre deux objets (Salton, 1989).

Dans notre cas, nous avons opté pour la similarité sémantique.
Soit $k = (O, P, R)$ un contexte formel. La similarité entre deux objets a et b de O se caractérise par les propriétés communes. D'une manière formelle, cette similarité s'exprime comme suit :
Soit P_a et P_b les ensembles des propriétés respectives des objets a et b. Les propriétés communes de ces objets est l'ensemble $P_a \bigcap P_b$. La similarité se définit comme suit :

$$Sim(P_a, P_b) = \frac{|P_a \bigcap P_b|}{|P_a \bigcup P_b|}$$

4.5.4. *Présentation de l'algorithme LSP*

L'algorithme de sélection des pairs utilise la base $B(E_1, E_2)$, générée par le module de génération de profils pour choisir les pairs "pertinents". Pour une requête Q, l'algorithme détermine à partir de l'ensemble E_1 un ensemble de concepts similaires à Q (noté par $SQTC$: SimilarQueriesTermsConcepts), qui seront triés selon la valeur de similarité (fonction getSimilarConcepts() de l'algorithme 2). La similarité entre un concept $C \in E_1$ et une requête Q est calculée comme suit :

$$Sim(Q, Int(C)) = \frac{|Q \cap Int(C)|}{|Q \cup Int(C)|}$$

Où :
$Int(C)$: représente l'intention du concept C.

Par la suite, pour chaque concept C_i de l'ensemble $SQTC$, nous déterminons, à partir de l'ensemble E_2, un ensemble de concepts similaires ($SQPC$: SimilarQueriesPeersConcepts) contenant les meilleurs pairs pour la requête Q. Dans ce cas, la

similarité entre un concept $C_i \in SQTC$ et un concept $C_j \in E_2$ est calculée comme suit :

$$Sim(Ext(C_i), Ext(Cj)) = \frac{|Ext(C_i) \cap Ext(C_j)|}{|Ext(C_i) \cup Ext(C_j)|}$$

Où :
$Ext(C)$ représente l'extension du concept C.

Finalement, l'ensemble des pairs choisis est égal à l'ensemble des pairs figurants dans l'intension de chaque concept de l'ensemble $SQPC$ (fonction getSelectedPeers() de l'algorithme 2.

Algorithme : PROPAGATION $(B, Q, Lv, Pmax)$
Entrées :
B : Base de connaissances.
Q : Requête à propager.
Lv : Liste des voisins.
$Pmax$: Nombre de pairs vers lesquels la requête sera propagée.
début
    $listeFinal := \emptyset$// : Liste de pairs vers lesquels la requête sera propagée.
    $selectedPeers$:=LearningPeerSelection(B, Q) ;
    $listeFinal := selectedPeers$;
    si $selectedPeers.size()$< $Pmax$ **alors**
        $N := Pmax - selectedPeers.size()$;
        AjouterAleatoire($Lv, ListeFinal, N$) ;
    Propager($ListeFinal, Pmax$) ;
fin

Algorithme 1 : ALGORITHME DE PROPAGATION

5. Expérimentations

5.1. *Environnement*

Pour tester notre approche, nous avons choisi le simulateur PeerSim (Jelasity *et al.,* 2007), qui est un outil Opensource écrit en Java qui présente l'avantage d'être déjà spécialisé pour l'étude des systèmes P2P et qui dispose d'une architecture ouverte et modulaire qui permet de l'adapter et de lui intégrer de nouvelles couches.

Plusieurs types de simulations sont possibles, mais celle qui nous intéresse ici est la simulation par cycles : pour cela, nous laisserons évoluer le réseau pendant un nombre prédéfini de cycles. Il nous suffira alors d'analyser l'état du réseau à la fin de la

Algorithme : LEARNINGPEERSELECTION(B, Q)
Entrées :
B : Base de connaissances ;
Q : Requête ;
Sortie :
$selectedPeers$: liste des pairs sélectionnés ;//
début
  $SQPC = \emptyset$;
  $SQTC$=getSimilarConcept($B.E1, Q$) ;
  for $C \in SQTC$ **do**
    $SQPC := SQPC \cup$ getSimilarConcept($B.E2, Extent(C)$) ;
  $selectedPeers$:=getSelectedPeers($SQPC$) ;
  Return ($selectedPeers$) ;
fin

Algorithme 2 : ALGORITHME DE SÉLECTION DE PAIRS

simulation pour en tirer les conclusions. De façon à simuler et à observer tout type de réseau P2P, tous les éléments de la simulation sont paramétrables. Les données utiles au simulateur sont regroupées dans un fichier de configuration, qui définit quatre types d'éléments :

– *Les Protocoles* : Un protocole qui est une fonctionnalité associée à un noeud. En effet, dans un réseau P2P que l'on veut simuler avec PeerSim, chaque noeud dispose d'une pile de protocoles qui vont être exécutés à chaque cycle de simulation, et ceci dans leur ordre de déclaration dans le fichier de configuration.

– *Les Initializers* : Ce sont des modules lancés en début de simulation (1er cycle) pour initialiser les protocoles. Par exemple, il servent à définir la topologie initiale du réseau (associer à chaque noeud ses voisins), à distribuer le jeu de données (requêtes et documents).

– *Les Dynamics* : Leur but est d'introduire un dynamisme dans le réseau, comme par exemple des suppressions de liens entre noeuds, ou des suppressions de noeuds.

– *Les Observers* : Comme leur nom l'indique, ils ont pour but de surveiller le réseau.

5.2. *Intégration de la solution*

Comme nous l'avons déjà mentionné, le simulateur PeerSim est un environnement de simulation pour les réseaux P2P qui n'est pas adapté à un système de recherche d'information P2P. Pour cela et afin de tester notre algorithme de routage, nous devons implémenter tout un système de recherche d'information P2P qui utilise la méthode de routage proposée dans cet article. Pour contourner cette difficulté,

nous avons décidé d'utiliser le simulateur générique de systèmes P2P de recherche d'informations (Sécard *et al.,* 2006) réalisé dans le cadre du projet de recherche RARE au sein de l'Institut National de Télécommunication de Paris (RARE, 2008). Ce Simulateur est construit au dessus de PeerSim et peut être vu comme une spécialisation de PeerSim pour la recherche d'information. L'utilisation de ce simulateur nous a permis de développer plus facilement notre système puis de le comparer avec le système Gnutella. Pour mettre en oeuvre la méthode proposée, nous avons implanté un nouveau protocol, un nouveau Initializer afin d'initialiser les bases de connaissances ainsi qu'un observateur pour afficher les résultats de simulation et mettre à jour les fichiers logs.

5.3. *Les données source*

Pour tester notre algorithme LPS, nous avons utilisé le jeu de données " BigDataSet ", développé dans le cadre du projet RARE. Ce jeu de données a été obtenu à partir d'une analyse statistique sur des données collectées d'un système pair-à-pair Gnutella (Sécard *et al.,* 2006) et des données de la collection TREC (TREC, 2008), ce qui nous permet de réaliser des simulations en conditions réelles.
BigDataSet est composé de 25 000 documents et 4999 requêtes qui est réparti avec duplication sur 499 pairs. En effet, quelques requêtes et documents sont répliqués. Ce jeu de données fournit des fichiers XML décrivant les noeuds du système et les documents qu'ils possèdent, ainsi que les requêtes qui seront lancées sur le réseau.

5.4. *Mesures d'évaluation*

Pour tester les performances de notre approche, nous avons utilisé les métriques rappel (R) et le nombre de messages, définies comme suit pour une requête Q et sachant que DPR est le nombre de documents pertinents retournés et DP est le nombre de documents pertinents

– *R(Q)* = $\frac{DPR}{DP}$

– *Nombre de messages* = nombre de messages échangés pour répondre à la requête Q

5.5. *Initialisation des paramètres de simulation*

La simulation de notre algorithme ainsi que celui utilisé par Gnutella s'est basée sur les paramètres suivants :

– *TTL* : Profondeur maximale de recherche, initialisée à 5.

– *Pmax* : Nombre de pairs auxquels la requête doit être propagée.

– *Overlay size* : Nombre de pairs dans le réseau, initialisé à 500 (nombre de pairs dans le jeu de données).

De plus notre algorithme a besoin d'une base de connaissances pour chaque pair. Pour cela, nous avons lancé les 20600 premières requêtes, en utilisant l'algorithme de Gnutella, afin de construire un fichier log initial pour chaque pair. Par la suite, nous avons lancé l'exécution du module de gestion de profils pour construire une base de connaissances pour chaque pair à partir de son fichier log, noté $B1$. Pour prendre en compte les nouvelles informations sur les requêtes passées, les bases des connaissances sont mises à jour après l'émission d'un nombre de requêtes. Le tableau 2 décrit les bases générées selon les caractéristiques suivantes :

– *Nombre de requêtes* : Nombre de requêtes émises par les différents pairs du système. Ce nombre, pour une base B_i est égal au nombre de requêtes relatives à une base B_{i-1} plus le nombre de nouvelles requêtes.

– *Taille moyenne* : C'est la somme des tailles des bases divisée par le nombre de pairs.

Nom de la base	# requêtes	Taille moyenne de la base (en Mo)
B1	20652	0.046
B2	28927	0.071
B3	34732	0.092

Tableau 2. *Caractéristiques des bases de connaissances de test*

5.6. *Résultats d'expérimentation*

Pour comparer les performances de notre algorithme par rapport à celui de Gnutulla nous avons calculé le rappel moyen et la moyenne de nombre de messages par intervalle de 2000 requêtes envoyées par les différents pairs du système. La figure 5, montre que le rappel moyen pour notre algorithme, varie entre 0.82 et 0.86 alors que le rappel moyen pour Gnutella varie entre 0.43 et 0.44. De plus, le rappel pour notre algorithme augmente à chaque enrichissement des bases de connaissances. La figure 6 montre que la moyenne de nombre de messages pour notre algorithme a diminué de 283 en utilisant la base initiale $B1$ à 262 en utilisant la base $B2$ et à 259 en utilisant $B3$. Par contre il reste stable pour Gnutella avec une valeur égale à 313. A partir de ces deux figures, nous pouvons déduire que la taille de la base de connaissances a un impact sur la qualité de routage de notre algorithme.

6. Conclusion

De nombreux travaux de recherche proposent d'exploiter la sémantique sur le contenu des pairs pour router plus efficacement les requêtes, mais peu de travaux

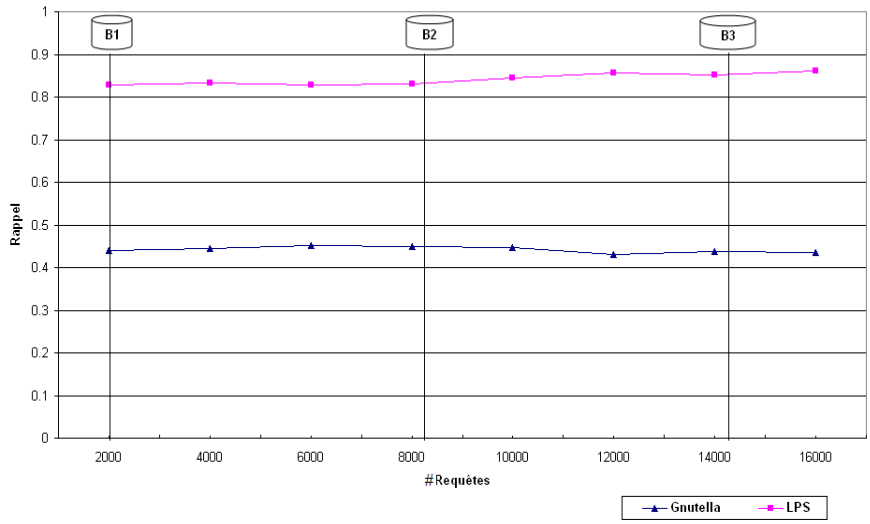

Figure 5. *Rappel en fonction du nombre de requêtes*

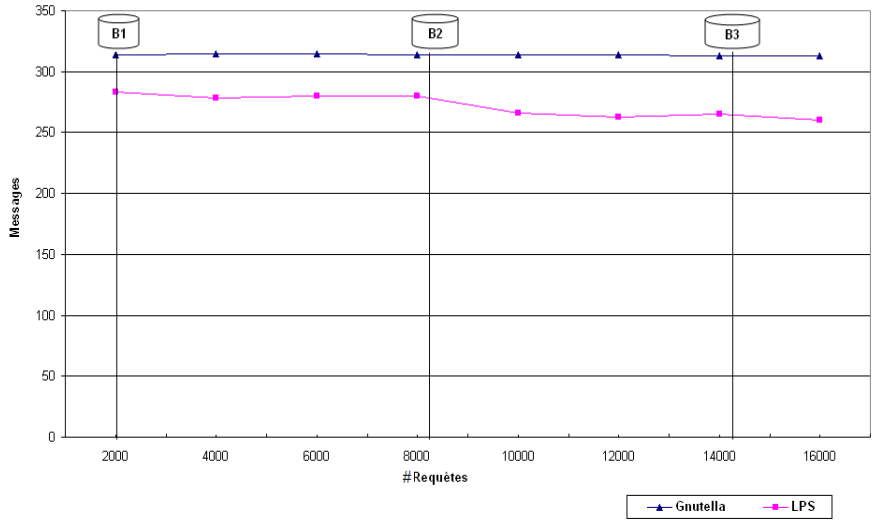

Figure 6. *Nombre de messages en fonction du nombre de requêtes*

explorent les informations relatives à l'historique des requêtes. Ainsi, nous avons proposé un algorithme de routage des requêtes par apprentissage basé sur les profils des utilisateurs qui sont déduits à partir des historiques de requêtes. Pour construire ces profils, nous avons utilisé une approche formelle basée sur l'Analyse Formelle de Concepts. Des tests, réalisés en utilisant le simulateur PeerSim, ont montré que notre algorithme de routage est plus performant qu'un algorithme de routage classique, en terme de rappel et de nombre de messages, et que la qualité de routage de notre algorithme dépend de la taille de la base de connaissances. Il nous reste encore à définir une stratégie de maintenance des bases de connaissances et à étudier plus finement son impact sur la qualité de routage.

7. Bibliographie

Callan J. P., Lu Z., Croft W. B., « Searching Distributed Collections with Inference Networks », *In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM Press, p. 21-28, 1995.

Chernov S., Serdyukov P., Bender M., Michel S., Weikum G., Zimmer C., « Database selection and result merging in P2P web search », *In DBISP2P 2005*, 2005.

Christoph T., Steffen S., Adrian W., « Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors », *13th International World Wide Web Conference, May 17-22, New York City, NY*, 2004.

Defude B., « Organisation et routage sémantiques dans les systèmes pair-à-pair », *Actes du XXVème Congrès INFORSID, May 22-25, Perros-Guirec, France*, p. 12-8, 2007.

Ganter B., Wille R., *Formal Concept Analysis : Mathematical Foundations*, Springer-Verlag New York, Inc, 1997.

Gnutella, « Gnutella website », `http ://www.gnutella.com/`, January, 2007.

Godin R., Missaoui R., Alaoui H., « Incremental concept formation algorithms based on galois (concept) lattices », *Computational Intelligence*, vol. 11(2), p. 246-267, 1995.

Jelasity M., Montresor A., Jesi G. P., Voulgaris S., « The Peersim Simulator », `http ://peersim.sf.net`, 2007.

Luis G., Hector G., Anthony T., « The effectiveness of gloss for the text database discovery problem », *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27*, ACM Press, p. 24- 27, 1994.

Luis G., Hector G.-M., « Generalizing gloss to vector-space databases and broker hierarchies », *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, Morgan Kaufmann, p. 78-89, 1995.

Raja C., Bruno D., Georges H., « Définition et diffusion de signatures sémantiques dans les systèmes pair-à-pair », *Extraction et gestion des connaissances (EGC'2006), Actes des sixièmes journées Extraction et Gestion des Connaissances, Lille, France, 17-20 janvier 2006, 2 Volumes*, vol. RNTI-E-6 of *Revue des Nouvelles Technologies de l'Information*, p. 463-468, 2006.

RARE, « Le projet RARE (Routage optimisé par Apprentissage de REquêtes) », `http ://www-inf.int-evry.fr/defude/RARE/`, 2008.

Salton G., « Automatic Text Processing : The Transformation, Analysis, and Retrieval of Information by Computer », *Addison Wesley*, 1989.

Sécard J., Bruno D., Chiky R., Georges H., Sorin M., « Un simulateur générique pour les systèmes de recherche d'informations en pair à pair », *3ème conférence en Recherche d'Informations et ses Applications, Lyon March 15-17*, 2006.

TREC, « Text REtrival Conference », `http ://trec.nist.gov/`, 2008.

Valtchev P., Grosser D., Roume C., Hacene M. R., « Galicia : an open platform for lattices », *In Using Conceptual Structures : Contributions to the 11th Intl. Conference on Conceptual Structures (ICCS'03*, Shaker Verlag, p. 241-254, 2003.