MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS DE MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique
Département de Mathématiques et d'Informatique
Filière Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THEME :

VERS UN SYSTEME DE RECOMMANDATION P2P

Etudiant(e) : « BELKHIRA MOHAMED »

Encadrant(e) : « KENNCHE AHLEM »

Année Universitaire 2015/2016

Sommaire

REMERCIMENTS

En tout premier lieu, nous remercionsle bon Dieu, tout puissant, de nous avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés.

Nous tenons à exprimer toute notre reconnaissance à notre Directeur de mémoire Madame **KENNICHE AHLEM**. Nous la remercions de nous avoir encadrées, orientées et aidées.Nous avons profité pendant longtemps du savoir et du savoir-faire dont nous avons pu bénéficier au cours de nombreuses discussions. Nous aimerons aussi la remercier pour l'autonomie qu'elle nous a accordés, et ses précieux conseils qui nous ont permis de mener à bien ce travail.

Nous exprimons toute notre reconnaissance à notre jury de la faculté de l'informatique et sciences exactes de l'université d'Abd El Hamid Ibn Badis, trouvent ici l'expression de nos vifs remerciements pour avoir bien voulu juger ce travail.

Nous adressonsnos sincères remerciements à tous les intervenants et toutes les personnes qui par leurs paroles, leurs contributions dans ce travail, leurs écrits, leurs conseils et leurs critiques ont guidé nosréflexions et ont accepté de répondre à nos questions durant nos recherches. Nous ne pouvons achever ce projet, sans exprimer nos sincères gratitudes à tous les professeurs de notre faculté, pour leur dévouement et leur assistance tout au long de notre formation.

Nous remercionsnos très chers parents, qui ont toujours été là pour nous,
« Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous nous avez données un magnifique modèle de labeur et de persévérance. Nous sommes redevables d'une éducation dont nous sommes fières ».

Nous remercions nos familles pour leur encouragement.
Enfin, nous remercions très spécialement Maamar, Adel, *p*our leur sincère amitié et confiance, leur soutien inconditionnel et leur encouragement, et à qui nous devons notre reconnaissance et notre attachement.

À tous ces intervenants, nous présentons nos remerciements, notre respect et notre gratitude.

DEDICACE

Au nom du dieu le clément et le miséricordieux louange à **ALLAH** *le tout puissant.*

Je dédie ce modeste travail en signe de respect, reconnaissance et de remerciement :

A mes parents,

Sans leurs soutiens et leurs conseils, mes accomplissements n'auraient pas eu lieu, ils ont été derrière moi dans chacun de mes pas tout au long de ma vie, ma plus profonde gratitude leurs ai exprimé, aucun mot ne pourrait qualifier l'estime que je leur porte ni le bien qu'ils m'ont fait, apporter et donner.

A mon frère Ali,

Le meilleur exemple, un modèle que j'ai toujours voulu suivre durant mes accomplissements, merci pour tous les conseils que tu m'as soigneusement prodigué.

A ma très chère sœur Wissame,

Pour son amour, son soutien et sa présence dans ma vie.

A tous mes ami(e)s Ali, Maamar, Asma, karim et tous ceux qui me sont chers. .

A tous ceux qui m'aime

A tous ceux que j'aime

A tous qui m'ont aidé de près ou de loin,

Que Dieu vous garde!

Mohammed,

LISTE DE FIGURE

Aucune entrée de table d'illustration n'a été trouvée.

L'internet a été conçu dans ses débuts comme étant une plate-forme pair à pair où tous les partenaires partagent et stockent les fichiers de façon égale. Vu l'accroissement rapide du matériel informatique et du nombre d'utilisateurs, cet espace est devenu difficile à maintenir ce qui a produit des domaines de recherche portant des défis importants pour le partage et la localisation de données.

Les systèmes pair-à-pair sont actuellement un cheval de bataille où la localisation des fichiers est au cœur. Ainsi, les systèmes se différencient de la façon avec laquelle ils indexent et routent les requêtes. Cette technologie pair-à-pair a évolué de centralisé à l'hybride en passant par la technologie pure où il n'y a pas de maitre qui gère les ressources et les demandes. La recherche de fichiers a elle-même évolué d'aveugle (où la requête touche à beaucoup de pairs) à sémantique où la destination détentrice des réponses est rapidement localisée.

Problématique et objectif :

Plusieurs problèmes apparaissent dans les systèmes peer to peer nous nous intéressons dans notre cas aux systèmes de recherches d'informations et de partage de fichiers peer to peer. En effet ses systèmes sont très pauvres en fonctionnalités d'indexation et de recherche d'information. L'objectif et d'arrivé à une recherche aussi efficace que dans les systèmes client/ serveur.

Beaucoup de travaux se font dans ce sens mais le problème principale et qu'il n'existe pas jusqu'à maintenant une plate forme de simulation qui peut permettre d'évaluer ses systèmes vue la difficulté de concevoir des systèmes réel et dynamique tél que le paradigme P2P.

L'objectif de notre travail est d'implémenté un réseau peer to peer dans le but d'alléger la tâche des ultrapairs. Dans ces travaux, nous avions tenu compte de l'aspect utilisateur. Autrement dit nous composions les vues des pairs (la couche basse du système) selon les requêtes échangées. Comme suite de ces travaux, nous proposons d'appliquer cette méthode sur la couche maitresse du système, cette fois-ci nous tenons compte de l'aspect structurelle, c'est-à-dire les vues des ultrapairs sont composées à partir des documents réellement dans le system.

Nous proposons d'appliquer une méthode de routage et nous passons à l'aspect sémantique afin que nous résolvions les requêtes. Nous expérimentons ces méthodes sur des données réelles sur une architecture pair-à-pair hybride.

Le présent mémoire se compose des chapitres suivants :

- Dans le premier chapitre, nous donnons un état de l'art plus ou moins exhaustif de la technologie pair-pair.
- Le chapitre deuxième présent les aspects des systèmes de recommandation ; nous présentons des techniques de filtrage d'information pour Les système de recommandation et nous présentons des exemples.
- Le chapitre troisième contient les détails fins de la conception de notre approche.
- Le quatrième chapitre présente les résultats de l'expérimentation.

- Nous concluons ce mémoire et nous donnons les perspectives de nos travaux en fin de ce mémoire.

1.1. Introduction:

Dans ce chapitre, nous allons décrire les différents concepts des systèmes P2P. Nous y fournissons leur définition, les motivations d'utilisation des ces systèmes. Ensuite, nous détaillerons les différentes classes des systèmes P2P, et la recherche d'information sur les systèmes P2P.

1.2. Présentation des systèmes pair-à-pair :

1.2.1. Historique et Définitions [3] :

Historiquement, le P2P est devenu très populaire en 1999 aux Etats Unis, avec le logiciel de partage de musique en ligne Napster [Napster]. Rapidement, ce dernier a connu un grand problème de copyright pour les compagnies et les éditeurs de disque, ce qui a mené Napster à des poursuites judiciaires en 2000.

Ce problème a été la cause principale de l'apparition des systèmes P2P totalement décentralisés, pour rendre le contrôle judiciaire plus difficile. Cependant, il y a des nombreuses utilisations légales qui sont aujourd'hui impliquées dans le monde du P2P, mais il est dur d'interdire l'échange de fichiers illégaux (musique, films...).

Définition [10]:

Le pair-à-pair (en anglais, peer-to-peer, abrégé P2P) est un modèle de réseau informatique proche du modèle client-serveur. À la différence du modèle client-serveur où un seul gros ordinateur (serveur) dessert de l'information à de nombreux terminaux (clients), dans le modèle P2P, chaque client est aussi un serveur. Tous les ordinateurs récupèrent de l'information et resservent l'information obtenue.

2.1. Architecture des systèmes P2P :

Dans cette section nous allons présenter les architectures des systèmes pair-à-pair. Elles sont divisées en deux grandes classes : *architecture des réseaux non structurés, architecture des réseaux structurés.*

2.1.1 Les systèmes P2P non structurés :

Les systèmes P2P non structurés sont des réseaux où il n'y a ni répertoire centralisé ni contrôle précis sur la topologie du réseau et sur l'emplacement des fichiers. Ces systèmes sont les plus simples et les plus anciens, où la recherche se fait de proche en proche à travers le réseau. Ils se divisent en trois architectures :

- Architecture centralisée,
- Architecture décentralisée,
- Architecture hybride.

2.1.1.1 Architecture centralisée :

Cette architecture se compose d'un unique serveur relié à tous les utilisateurs (voir figure 1.2). Son rôle est de recenser les fichiers proposés par les différents clients. Contrairement au mode Client/serveur, ce serveur centralisé ne dispose pas de fichiers, il possède uniquement des index sur les fichiers. La recherche se fait à partir du serveur mais le téléchargement se fait à partir du nœud possédant le fichier. Le représentant le plus connu de ce mode de P2P est Napster.

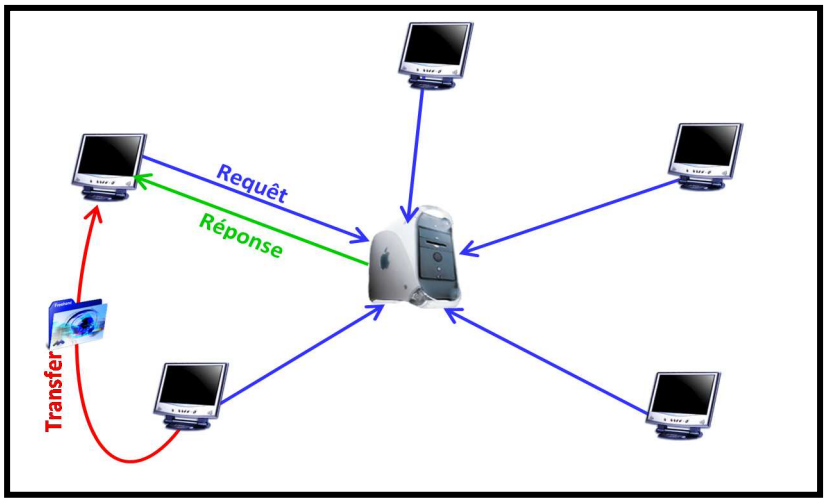


Figure 1 : Exemple d'architecture centralisée [3].

Napster [20] :

En Mai 1999, Shawn Fanning, un jeune homme de 19 ans étudiant à l'Université de Boston écrit un code qui va donner naissance à un programme, qu'il appelle Napster (en référence au surnom que lui ont donné ses amis). Napster est une application de partage de fichiers musicaux mp3. Dès décembre 1999, il fut attaqué par les majors du disque et finalement, en juillet 2001, fut contraint d'arrêter ses serveurs.

Figure 1.2 : Logo Napster [20].

La figure 1.3 schématise les étapes de fonctionnement présenté ci-dessous.

1. Un pair se connecte directement sur le serveur central et lui communique la liste des fichiers qu'il partage, ainsi qu'un numéro de port TCP où il pourra être connecté pour un éventuel téléchargement.

2. Chaque utilisateur désirant obtenir une donnée lance une requête au serveur. Ce dernier dispose principalement de deux informations : celles sur les fichiers (nom, Taille, etc.), et celles sur les utilisateurs (@ IP, nombre de fichiers, type de connexion, etc.).

3. Le serveur répond alors en donnant la liste des utilisateurs possédant la donnée.

4. L'utilisateur peut se connecter directement à l'utilisateur possédant la donnée pour la télécharger.

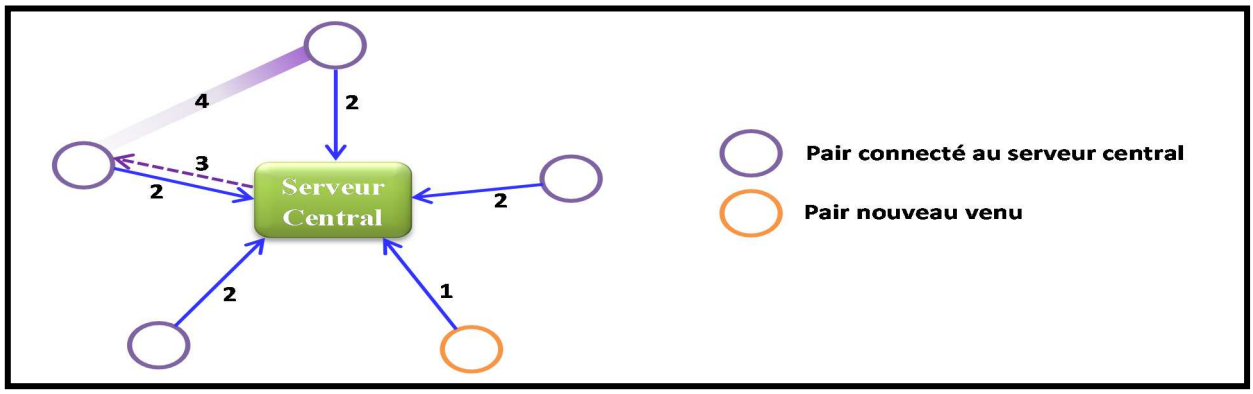
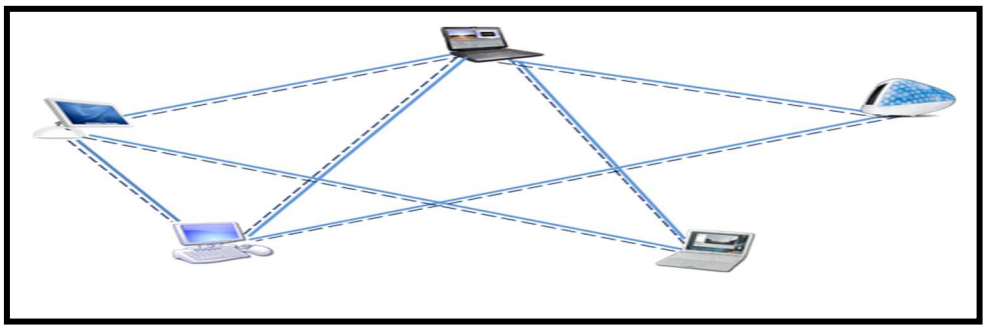


Figure 1.3 : fonctionnement de Napster [3].

2.1.1.2 Architecture décentralisée :

Les réseaux pair-à-pair ayant cette architecture sont appelés p2p pur, du fait de l'absence d'une quelconque entité jouant un rôle d'administrateur du réseau. Chaque nœud joue le rôle



de client et de serveur en même temps (voir figure 1.5).

Figure 2: Exemple d'architecture décentralisée [3].

Chaque nouveau nœud doit connaître un nœud appartenant au réseau, qui lui sert de bootstrap pour s'insérer dans le réseau. La recherche se fait soit par inondation comme dans GNUtella soit en profondeur comme Freenet. Nous allons maintenant présenter un système décentralisé : GNUtella

GNUtella [8] :

GNUtella (version 0.4) est le premier système d'échanges de fichiers où la recherche et le transfert de fichiers sont complètement décentralisés. Il a été créé en 2000 par Justin Frankel et Tom Pepper,

Figure 2.1 : logo de GNUtella [8].

Dans GNUtella, chaque nœud possède un ensemble de fichiers qu'il met à la disposition des autres et qu'il indexe localement. Il est connecté à un ensemble de voisins (typiquement une dizaine). La recherche se fait par inondation : un nœud envoie une requête de recherche à tous ses voisins qui la retransmettent à leur tour à tous leurs voisins et ainsi de suite. Pour limiter le nombre de retransmissions, un champ *ttl* (Time-To-Live), initialisé à 7, est associé à chaque message et est décrémenté à chaque saut. Quand le *ttl* est égal à 0, le message n'est plus retransmis.

Si un nœud trouve dans ses index, un fichier qui convient à la requête, il envoie la description du fichier et son adresse IP au nœud qui lui a transmis la requête. La réponse remonte en suivant le chemin inverse de la requête vers le nœud initiateur de la requête. Ce nœud choisit ensuite un fichier parmi les réponses reçues puis le télécharge directement à partir du nœud qui possède le fichier. Une fois le fichier téléchargé, il est indexé et mis à la disposition des autres nœuds, ce qui augmente la disponibilité du fichier. La figure 2.2 illustre un exemple de recherche d'un fichier. Le nœud C lance une requête de recherche (query) à tous ses voisins (B, D, E), le nœud B va retransmettre la requête au nœud A qui est son unique voisin. Le nœud A possède la donnée, il envoie alors une réponse (query hit) à B, qui la retransmet à C. C va donc télécharger la donnée directement à partir du nœud A.

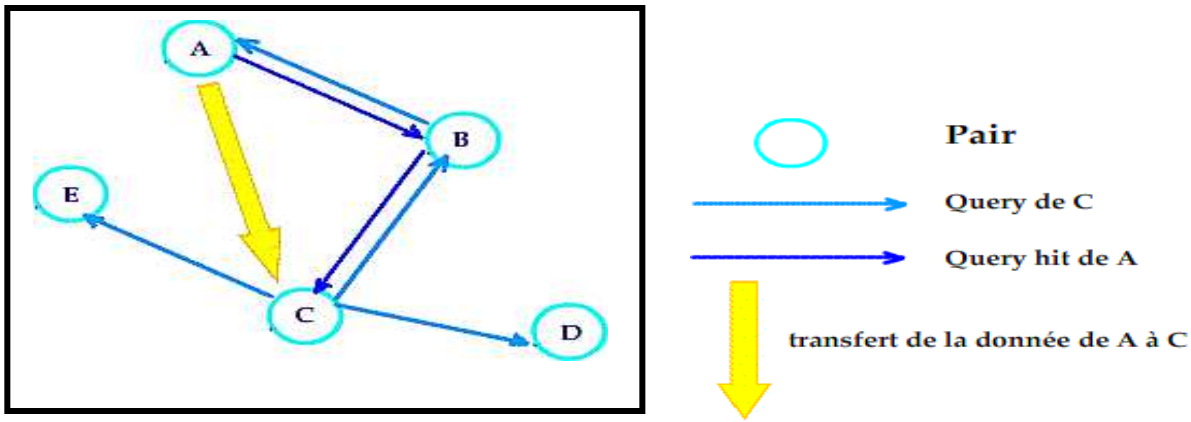


Figure 2.2 : recherche et récupération de donnée. [3]

2.1.1.3 Architectures hybrides :

Le modèle hybride a pour but d'utiliser les avantages des deux types d'architecture (centralisé et décentralisé). En effet, sa structure permet de diminuer le nombre de connexions sur chaque serveur (super pair) et ainsi d'éviter les problèmes de goulot d'étranglement. D'autre part les supers pairs forment entre eux un réseau décentralisé (voir figure 3).

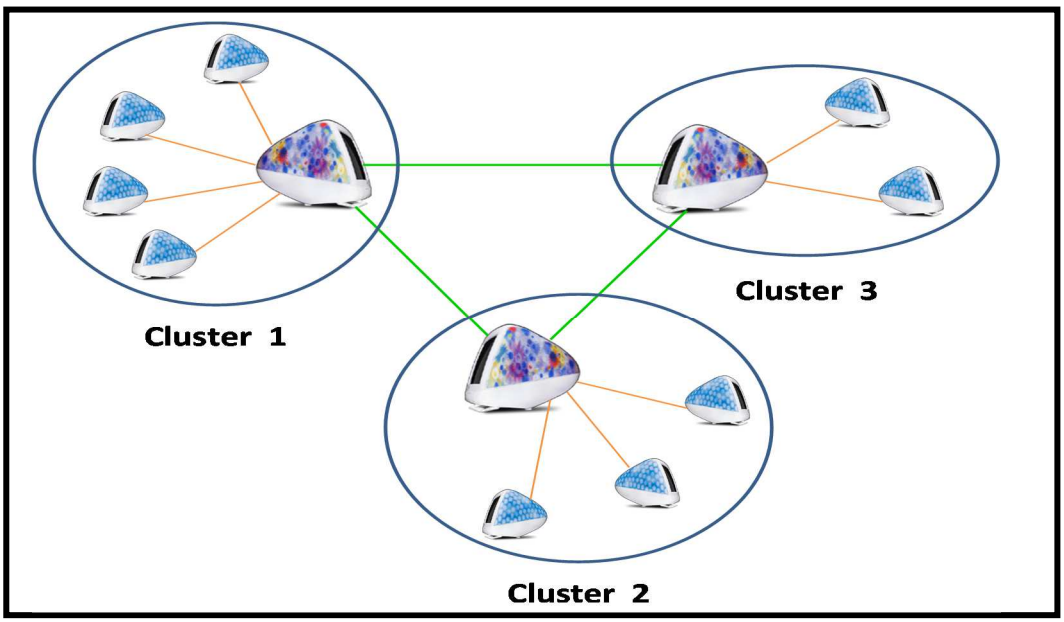


Figure 3: Exemple d'architecture hybride [3].

KazaA [15, 17]:

KazaA est une application P2P basée sur le réseau FastTrack. Niklas Zennström fondateur de KazaA, est également développeur de Skype. KazaA fut le premier Logiciel à offrir une décentralisation de serveur : Les fichiers partagés sont stockés dans un dossier spécialisé ; ce dossier étant sur l'ordinateur de chaque utilisateur.

Ce système de partage de fichiers en temps réel ne nécessite pas de serveur centralisé. Sur le réseau chaque ordinateur sert de serveur pour les autres. Pour optimiser la vitesse du trafic sur le réseau, KazaA, opère avec une distinction entre deux niveaux de pairs : ceux qui ont une connexion haut débit et ceux qui ont une connexion bas débit.

Les ordinateurs disposant d'une connexion bas débit se relient à un ordinateur ayant une connexion haut débit. Ce dernier devient dès lors un « super-pair ». Chaque super-pair indexe alors les fichiers des pairs bas débits qui lui sont rattachés, comme le fait le serveur central dans une architecture centralisée.

En revanche, entre les super-pairs, le système fonctionne comme un réseau décentralisé. Mais la propagation des données est plus rapide, puisqu'elle n'utilise plus que les connexions haut débit. Une fois la donnée trouvée, l'adresse IP du nœud possédant la donnée va être retransmise au nœud initiateur de la requête, une connexion directe peut alors s'établir entre les deux pairs comme le montre la figure 1.11.

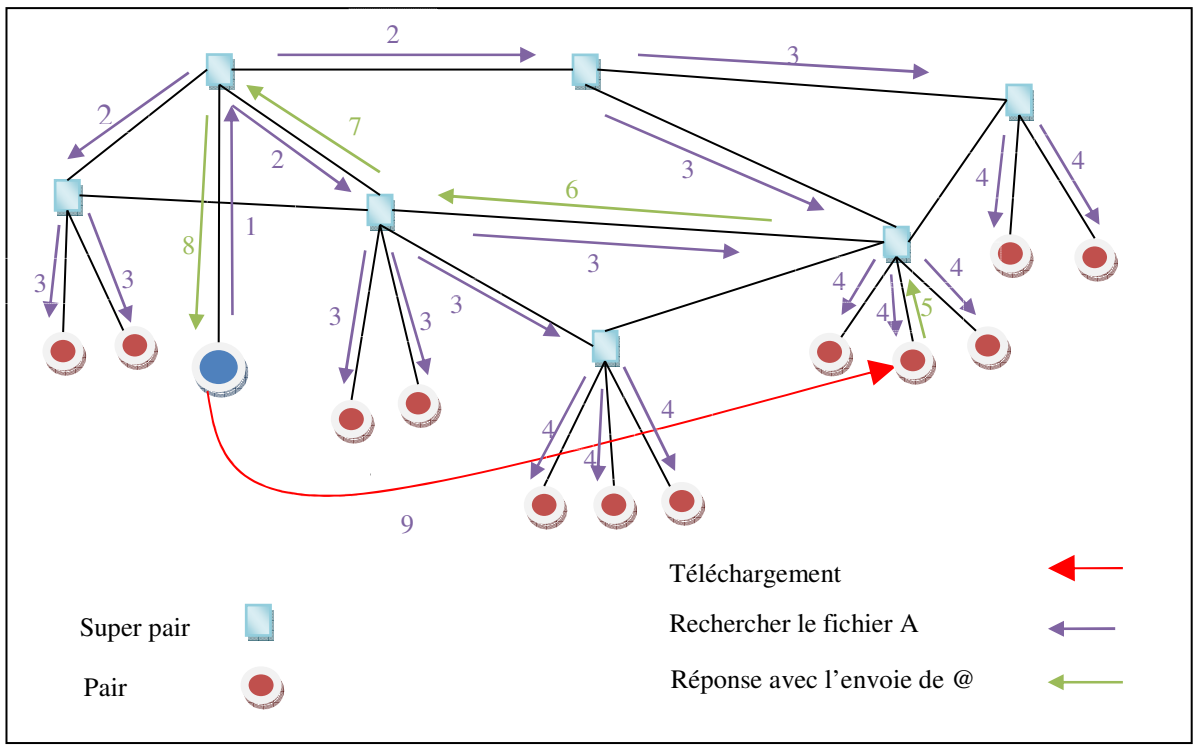


Figure 3.1 : Principe de fonctionnement de KazaA.

2.1.2 Les systèmes P2P structurés :

Pour diminuer la complexité de la recherche, les concepteurs de système P2P se sont tournés vers des structures de données connues. En effet, améliorer les algorithmes de recherche

Nécessite d'organiser le réseau, donc de le structurer. Une organisation de connexion est maintenue entre les nœuds. La plupart de ces systèmes, comme Chord **[30]** et Tapestry **[6]** sont basés sur les tables de hachage distribuées, permettant de réaliser des recherches en un nombre de messages croissant de façon logarithmique avec le nombre d'utilisateurs du réseau.

Nous pouvons citer CAN **[5]** (*Content Adressable Network*) qui est un réseau en tore dont la taille est fixée initialement ou Chord qui est développé selon une architecture hyper cube. Pour finir nous avons Tapestry dont le but est d'offrir un réseau de partage de données garantissant leur pérennité.

3.1. La recherche d'information sur les systèmes P2P :

Dans un système P2P, les résultats d'un système de recherche d'information ne dépendent plus seulement de la manière dont sont indexés les documents et de la mesure utilisée pour comparer les documents avec les requêtes. En effet la répartition des données, la topologie du réseau et les algorithmes de routage utilisés pour diffuser les requêtes vont avoir une grande influence dans les performances (pertinence des résultats, temps d'exécution et charge du réseau).

3.1.2. Caractéristiques de la recherche d'information dans les systèmes P2P :

La répartition des données :

Dans un système P2P, les pairs contiennent des données qui leur sont propres. Selon la répartition des données dans le réseau, les résultats du système de recherche d'information seront plus ou moins bons. Par exemples, si les documents pertinents sont contenus dans les voisins directs du pair initiateur de la requête, alors les résultats seront bons, même avec un faible TTL. Par contre, si les documents pertinents sont très éloignés du pair initiateur, ils ne sont pas nécessairement accessibles les résultats seront donc moins bons.

La topologie du réseau :

La topologie du réseau a aussi un impact important sur les résultats. En effet, plus la connectivité des pairs est importante et plus les pairs sont accessibles rapidement. Cela permet donc d'accéder aux données pertinentes de manière efficace. Par contre, le nombre de messages échangés est plus important.

Les algorithmes de routage :

L'algorithme de routage choisi pour rechercher l'information dans un système P2P a une influence sur les résultats retournés par le système de recherche d'information. Par

exemple, il est évident que la valeur du *TTL* a une influence sur la qualité des résultats retournés par le système de recherche d'information. Plus le *TTL* est grand et plus le nombre de pairs recevant la requête est importante : cela augmente les chances d'accéder à des sources de documents pertinents. Par contre, il faut noter que plus le *TTL* est grand plus le nombre de messages échangés est important et donc plus le temps de réponse est long.

4. Conclusion :

Le Peer to Peer (P2P) repose sur différentes architectures, il fournit de nouvelles opportunités pour construire des systèmes de gestion de données distribués à grande échelle. Contrairement au client-serveur, le P2P est un environnement très dynamique, où les pairs peuvent rejoindre ou quitter le réseau à tout moment. Ceci offre des avantages importants comme la décentralisation du contrôle, l'autonomie des pairs et le passage à l'échelle en nombre de pairs.

Le P2P a connu et connaît toujours un franc succès auprès du grand public grâce aux logiciels de partage et de communication. Néanmoins, le Peer to peer se développe également auprès du monde professionnel par son utilisation notamment dans le calcul distribué et le travail collaboratif.

1.1. Introduction:

Dans ce chapitre, nous allons décrire les différents concepts des systèmes de recommandation et Les deux entités de base qui apparaissent dans tous les systèmes de recommandations et la notion de recherche d'informations. Aussi on présente les techniques de filtrage d'information : filtrage collaboratif, filtrage basé sur le contenu et filtrage hybride.

1.2. Présentation des systèmes de recommandations :

1.2.1. Historique et Définitions :

Les racines des systèmes de recommandation remontent aux travaux étendus dans les sciences cognitives, la théorie d'approximation, la recherche d'informations, la théorie de la prévoyance et ont également des liens avec la science de la gestion et le marketing, dans la modélisation des choix du consommateur.

On retrace les premiers travaux portant sur le filtrage d'information à un article de Luhn sous le nom de « diffusion sélective d'une nouvelle information ». Le terme « Filtrage d'information » a été proposé par Denning, qui s'est concentré sur le filtrage des courriers électroniques. En 1987, deux catégories du filtrage d'information étaient proposées par Malone. La première catégorie est le filtrage cognitif nommé actuellement filtrage basé sur le contenu. La deuxième catégorie est le filtrage social qui correspond au filtrage collaboratif.

Définition [4] :

Une définition générale de Robin Burke qui les définit comme suit : "*Des systèmes capable de fournir des recommandations personnalisées permettant de guider l'utilisateur vers des ressources intéressantes et utiles au sein d'un espace de donnée important*". Leur but est de réduire la surcharge de l'information grâce à un processus de recueil, filtrage et recommandation de l'information d'une manière proactive.

2. Les deux entités de base des systèmes de recommandations :

- **L'«usager»** est la personne qui utilise un système de recommandation, donne son opinion sur diverses items et reçoit les nouvelles recommandations du système.
- **Le mot «item»** est le terme général utilisé pour dénoter ce que le système de recommandation recommande aux utilisateurs.

L'utilisateur peut avoir une recommandation pertinente par le système en fonction de connaissances variées (feedbacks d'utilisateurs, profil de l'utilisateur, le contexte, les items à recommander) et par l'action de l'utilisateur qui peut être enregistrée d'une manière implicite ou explicite. Ces actions génèrent des nouvelles recommandations pour la prochaine interaction avec le système. **[7]**

3. Notion de profil:

De façon générale, le profil d'un objet est un ensemble de caractéristiques permettant de l'identifier ou de le représenter. Plusieurs types de profils sont exploités dans les différentes techniques d'accès à l'information

Le profil utilisateur : il s'agit de la description des caractéristiques d'un utilisateur (données démographiques, centres d'intérêt, préférences, etc.).

Le profiling consiste à scruter, enregistrer et analyser les actions et successions d'actions d'un utilisateur lors de différentes sessions de recherche pour déterminer son profil.

Le profil de document : il correspond à la description d'un document qui est souvent réduite, en RI ou FI, à une liste de mots-clés pondérés décrivant le contenu sémantique du document. Les mots-clés et leurs poids sont obtenus en général par une opération d'indexation. Plusieurs travaux permettent actuellement de décrire les documents en utilisant également d'autres critères que ceux liés à leur contenu effectif.

A travers la notion de profil, les techniques d'accès à l'information tentent d'améliorer la pertinence des réponses renvoyées aux utilisateurs. A l'heure actuelle, la tendance est à la prise en compte de métadonnées (ou propriétés), obtenues par annotation, qui permettent d'améliorer la pertinence des résultats.

4. La recherche d'informations :

La recherche d'information sur fonde sur un principe d'indexation des données afin de répondre aux requêtes d'utilisateurs. Plus spécifiquement, la recherche documentaire, sous discipline de la recherche d'information, consiste _a interroger une base de connaissance par le bais de requêtes écrites en langues naturelles ou bien sous forme de mots clefs (nommées requêtes ad hoc). Ces mots clefs sont alors comparées a l'ensemble des indexes des documents présents dans la base de données du moteur de recherche. Dés lors, l'utilisateur se voit retourner un ensemble de résultats plus ou moins pertinents par rapport a sa requête initiale. Le problème inhérent a ce type de résultats et la surcharge d'informations que l'utilisateur doit filtrer. Le lien avec la recommandation est ici trivial, l'un des objectifs des systèmes de recommandation étant de faire le tri dans cette masse d'informations pour l'utilisateur de manière transparente.

La construction de profils d'utilisateurs constitue une première avancée dans le domaine de la recherche documentaire en termes de recommandation. En effet, comme le pro- pose Google notamment, l'utilisateur se voit maintenant pondérer les résultats obtenus suite a sa requête en fonction des précédentes recherches qu'il a effectuées et des pages qu'il a consultées. Il s'agit d'un système d'identification de l'utilisateur permettant alors d'analyser ses actions. Cette approche peut être qualifie d'analyse de traces et constitue l'un des fondements du filtrage d'informations sur lequel s'appuient les systèmes de recommandations.

C'est ainsi que l'on différencie les systèmes de recherche d'informations, pour lesquels la demande de l'utilisateur de l'orienter et le guider vers des choix appropries est explicite, et les systèmes de recommandation où la participation de l'utilisateur du système est non volontaire

5. Comment classer les différents systèmes de recommandation ?

Plusieurs facteurs entrent en considération afin de catégoriser les systèmes de recommandation.

- La connaissance de l'utilisateur (c.-_a-d. son profil en fonction de ses goûts).
- Le positionnement d'un utilisateur par rapport aux autres (la notion de classes ou réseaux d'utilisateurs).
- La connaissance des items a recommandé.
- La connaissance des différentes classes d'items a recommandé.

De ces facteurs sont produits divers type de recommandations dont les plus utilisées dans la littérature sont le Filtrage basée sur le contenu et le filtrage collaboratif. Ce document présente dans un premier temps ces deux approches ainsi que leurs hybridations.

6. Grandes familles de filtrage :

Les systèmes de recommandation sont définis comme étant “des outils logiciels et des techniques qui suggèrent aux usagers des éléments utiles” **[9]**. Afin d’identifier les informations à recommander, plusieurs stratégies ont été proposées dans la littérature. Elles sont généralement classées en trois catégories :

6.1. Basée sur le contenu :

Le système recommande des items qui sont similaires à ceux que l'utilisateur a aimés dans le passé. La similarité des items est calculée en se basant sur les caractéristiques associées aux items comparés. **[11]** L’objectif des SR à base du contenu est de cibler des objets pertinents issus d’un large espace de sources possibles d’une façon personnalisée pour les utilisateurs. Son principe

consiste à recommander les items similaires à ceux préférés par l'utilisateur dans le passé. Dans le but de recommander de nouveaux items intéressants, les SR à base de contenu essayent de faire correspondre les attributs des items avec les préférences et les intérêts de l'utilisateur. Pour un nouvel item, le système compare l'item avec le profil de l'utilisateur afin de prédire le score que pourrait porter l'utilisateur sur l'item. Les items sont alors recommandés en fonction de leur proximité aux utilisateurs **[12].**

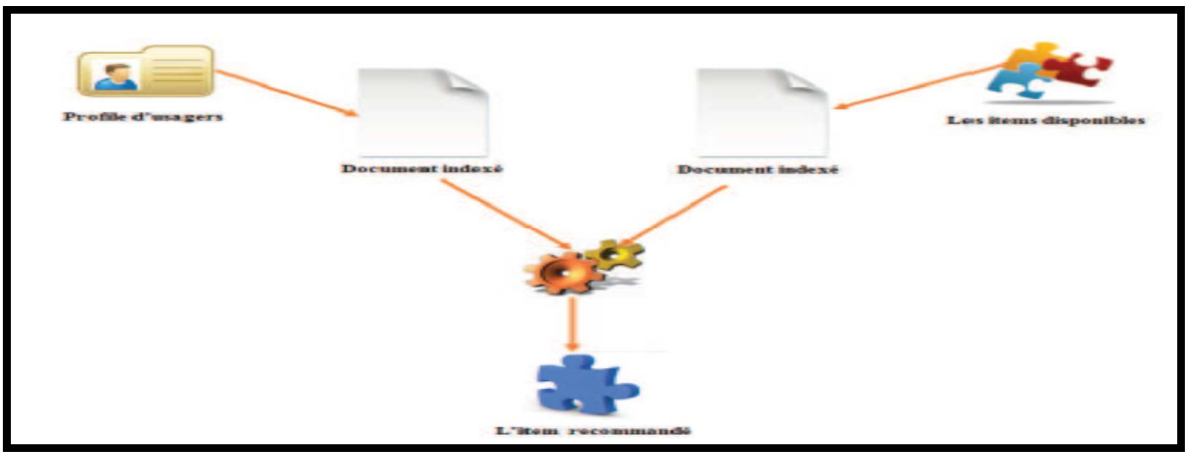


Figure 4: recommandation basé sur le contenu

L'avantage des systèmes de filtrage cognitifs, basés contenu est qu'ils permettent d'associer des documents à un profil utilisateur. Notamment, en utilisant des techniques d'indexation et d'intelligence artificielle. L'utilisateur est indépendant des autres ce qui lui permet d'avoir des recommandations même s'il est le seul utilisateur du système **[13].**

Ce type de systèmes présente certaines limitations :

- L'effet "entonnoir" : les besoins de l'utilisateur sont de plus en plus spécifiques, ce qui l'empêche d'avoir une diversité de sujets. Même pire, un nouvel axe de recherche dans un domaine bien précis peut ne pas être pris en compte car il ne fait pas parti du profil explicite de l'utilisateur.
- Filtrage basé sur le critère thématique uniquement, absence d'autres facteurs comme la qualité scientifique, le public visé, l'intérêt porté par l'utilisateur, etc.
- Les difficultés à recommander des documents multimédia (images, vidéos, etc.) et ceci à cause de la difficulté à indexer ce type de documents, c'est en fait la même problématique dont souffrent les systèmes de recherche.

- Problème de démarrage à froid : Un nouvel utilisateur du système éprouve des difficultés à exprimer son profil en spécifiant des thèmes qui l'intéressent. Ceci malgré les techniques d'apprentissage ou l'utilisateur fournit des textes exemples.

6.2. Le filtrage collaboratif :

S'appuie sur les appréciations données par un ensemble d'utilisateurs sur un ensemble d'articles. Ces appréciations, traduites en valeurs numériques, peuvent être des notes, des comptes d'achats effectués, des nombres de visites, etc. **[1]** Par exemple, les personnes, qui veulent regarder un film ou lire un livre, demandent à leurs amis leurs opinions. Donc, dans le filtrage collaboratif, la sélection des documents à proposer à un utilisateur ne dépend plus des termes constituant le document (filtrage basé sur le contenu), mais des évaluations faites par les membres de son voisinage.

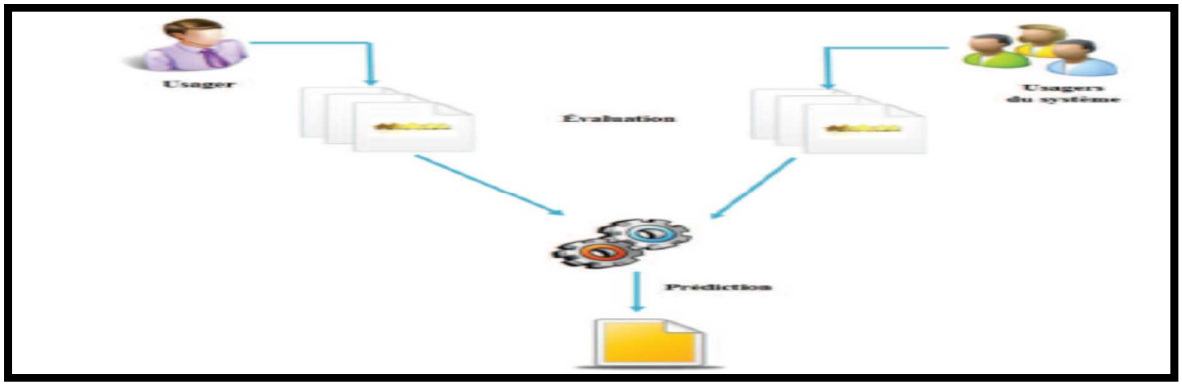


Figure 5: Recommandation basé sur le filtrage collaboratif.

On distingue deux grandes approches de filtrage collaboratif. L'approche se référant aux utilisateurs **[1]** consiste à comparer les utilisateurs entre eux et à retrouver ceux ayant des goûts en commun, les notes d'un utilisateur étant ensuite prédites selon son voisinage. L'approche se référant aux articles consiste à rapprocher les articles appréciés par les mêmes personnes et à prédire les notes des utilisateurs en fonction des articles les plus proches de ceux qu'ils ont déjà notés. Un problème du système collaboratif est que sa performance dépend beaucoup de la distribution des évaluations (notes) données par utilisateurs. Dans le cas ou il y a plusieurs items qui ont été utilisés et évalués par très peu d'utilisateurs, ces items seraient recommandés très rarement, même si ces utilisateurs ont donné des notes très hautes pour ces items. De la même façon, si dans le système il existe des utilisateurs qui ont des goûts très différents en comparaison avec les autres, le système ne peut pas trouver des similarités entre utilisateurs et donc ne peut pas donner des bonnes recommandations.

La figure 6 représente un tableau de films avec sur un axe les utilisateurs d'une même Système (ex : un groupe d'amis) et sur un autre les films. Chaque cellule de la matrice contient l'avis donné par un utilisateur pour un film, la cellule vide signifie qu'il n'a pas d'avis particulier sur ce film. Afin de prédire si Illyes apprécierait le film "Harry Potter" et probablement lui recommander ce film, on compare les Votes d' Illyes à ceux des autres utilisateurs choisis. On peut alors voir que Illyes et Imen ont des Votes identiques, et que Imen n'a pas aimé le film «Harry Potter », on pourrait alors prédire que Illyes n'aimera pas aussi ce film et de ne lui pas faire cette suggestion.

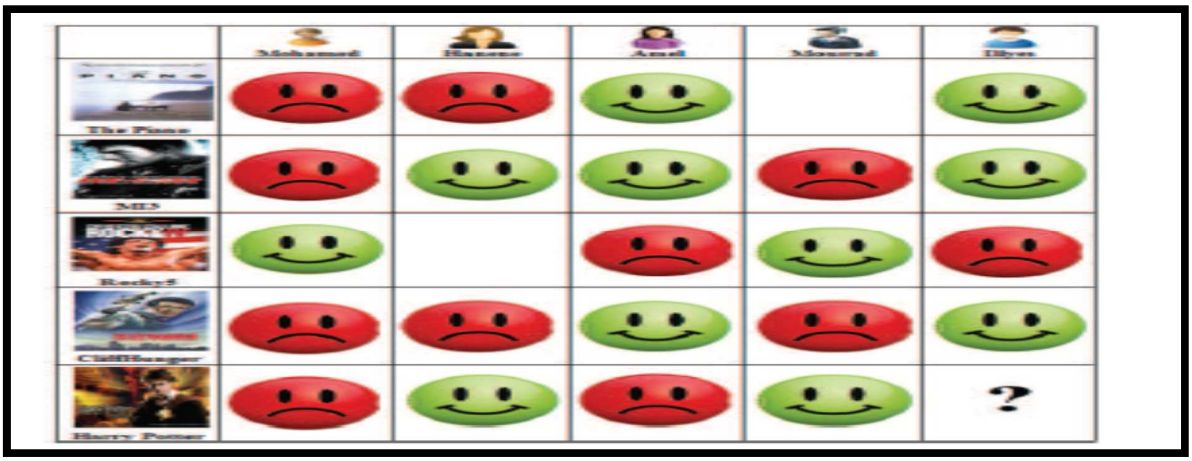


Figure 6:Exemple de recommandation basée sur le filtrage collaboratif.

6.3. Approches hybrides :

Les faiblesses des méthodes traditionnelles ont fait l'objet de plusieurs études dans la littérature scientifique concernant les systèmes de recommandation. De nombreuses solutions sont proposées. Elles sont, généralement, des méthodes basées sur la combinaison de méthodes traditionnelles présentées auparavant. Selon ce dernier, un système hybride est généralement organisé en deux phases :

- Effectuer de manière indépendante les filtrages des items via des méthodes collaboratives ou par le contenu (ou autres)
- Combiner ces ensembles de recommandations via des méthodes d'hybridations telles que des pondérations, commutations, cascade, etc.

7. Avantages et inconvénients des systèmes de recommandation :

Le tableau 1 résume les forces et faiblesses des méthodes traditionnelles utilisé par les systèmes de recommandation, en l'occurrence le Filtrage Collaboratif (FC), le Filtrage à Base de Contenu (FBC),

Techniques	Avantages	Inconvénients
Filtrage à base du contenu	Pas besoin d'une large communauté d'utilisateurs pour pouvoir effectuer des recommandations. • Une liste de recommandations peut être générée même s'il n'y a qu'un seul utilisateur. • La qualité croit avec le temps. • Pas besoin d'information sur les autres utilisateurs. • Prendre en considération les gouts uniques des utilisateurs.	• L'analyse du contenu est nécessaire pour faire une recommandation. • Problème de recommandation des images et de vidéos en absence de Métadonnées. • Nécessité du profil d'utilisateur.
Filtrage collaboratif	Ne demande aucune connaissance sur le contenu de l'item ni sa sémantique. • La qualité de la recommandation peut être évaluée. • Plus les nombre d'utilisateurs est grand plus la recommandation est meilleure.	• Démarrage à froid. • Nouvel Item. • Nouvel utilisateur. • Problème de confidentialité. • La complexité : dans les systèmes avec un grand nombre d'items et d'utilisateurs, le calcul croit linéairement.

Tableau 1 : Avantages et inconvénients des systèmes de recommandation

8. Conclusion :

Ce chapitre a présente un certain nombre d'approches visant à produire des systèmes de recommandation. Nous avons ainsi évoque deux types d'approche et la combinaison des deux telles que listées ci-dessous.

- Les approches basées sur le contenu.
- Les approches utilisant un filtrage collaboratif.
- Les méthodes hybrides.

La tendance actuelle des systèmes de recommandation est plutôt axée sur des méthodes nouvelles, multicritères, multidimensionnelles ou encore se fondant sur des notions psychologiques comme les émotions, les opinions. Notons cependant qu'un système de recommandation doit avant tout s'adapter aux données, celles là mêmes que l'on proposera à un utilisateur. Ainsi, le choix d'une méthode de recommandation doit en premier lieu être dirigé par ce critère.

CHAPITRE 3 CONCEPTION & MODÉLISATION

CHAPITRE 3 CONCEPTION & MODÉLISATION

Ce chapitre se divise en deux parties. La première partie consiste à présenter le modèle sur lequel nous exécuterons la méthode de recherche que nous présenterons dans la deuxième partie.

Partie 1: Modèle du système

1.1. Introduction et problématique

Les systèmes pairs à pairs sont récemment devenus un média populaire pour le partage d'énormes quantités de données. Ils permettent de distribuer les données sans avoir recourt à de puissants et coûteux serveurs. En plus de leur capacité à mettre en commun et d'exploiter de grandes quantités de ressources, les systèmes pairs à pairs comprennent l'auto-organisation, l'équilibrage de la charge, l'adaptation et la tolérance aux pannes.

En raison de ces qualités, beaucoup de recherches se sont focalisées sur la compréhension des questions entourant ces systèmes et l'amélioration de leurs performances. Beaucoup de recherches qui ont été faites récemment, se focalisent sur l'amélioration et l'efficacité de la localisation et du routage de l'information dans ces systèmes **[48, 49, 50,51].**

Néanmoins plusieurs problèmes doivent être considérés dans les systèmes p2p de partage de fichiers. Par exemple, la recherche dans Gnutella 0.4 (ou Gnutella1) s'effectue de la façon d'inondation en utilisant un TTL (Time To Live) pour contrôler la profondeur d'une requête. Certes, cette approche permet de retourner un grand nombre de résultats, mais ne garantit pas de trouver les bonnes réponses, car aucun mécanisme, pour sélectionner les sous ensembles des pairs les plus pertinents, n'est utilisé dans le processus de localisation. De plus, Gnutella se sert du TTL pour contrôler la propagation d'une requête. Cependant, il n'est pas facile de choisir un TTL approprié. Si le TTL est trop élevé, la requête va surcharger le réseau inutilement; s'il est trop bas, le pair peut ne pas trouver l'objet même s'il existe une copie quelque part. Donc les réponses dans Gnutella ne sont pas exhaustives et sont sujettes à la dynamique du système. Pour palier ces inconvénients, les concepteurs de Gnutella ont proposé plusieurs changements et sont passés à la version de Gnutella 0.6. Cette version introduit le schéma des ultrapairs et des nœuds feuilles pour créer une structure hybride du réseau Gnutella. Dans cette nouvelle version, seulement les ultrapairs acceptent des connexions avec les autres nœuds. Aucune feuille ne peut se connecter à une autre feuille directement sauf pour le téléchargement. Les ultrapairs indexent le contenu des feuilles qui sont rattachées à eux et répondent aux requêtes de recherche en incluant les documents de leurs feuilles. Les avantages de ce réseau est qu'il combine des éléments des systèmes à la fois pures et hybrides, donc il combine l'efficacité des deux recherches, mais lorsque les demandes de requêtes augmentent, les frais engagés sur l'index centralisé au niveau du nœud ultrapair sont très élevés; ce qui constitue un goulot d'étranglement. De plus la même idée utilisée dans Gnutella 0.4 concernant le TTL a été reprise dans Gnutella 0.6, où les Ultrapairs renvoient la même requête (sonde) à leurs ultrapairs voisins toutes les 2 ou 3 secondes afin de maximiser le nombre de réponses en espérant que la dynamique du système les rapprochera des bonnes réponses. La recherche dans ces systèmes permet de donner un nombre considérable de résultats, mais les réponses retournées ne sont pas forcément satisfaisantes,

car le processus de propagation des requêtes ne se base sur aucun critère de qualité pour la recherche des pairs pertinents pour une requête.

Il est clair, que Gnutella n'est pas un protocole spécialement intelligent, il dépend de la valeur du TTL et de la dynamique du système pour trouver les réponses et ne se base sur aucun critère de qualité pour localiser les pairs pertinents : en effet ces systèmes donnent des réponses quantitatives et n'apportent aucune qualité à leurs résultats.

Une solution consiste à organiser le réseau hybride en un ensemble de clusters de pairs similaires. Cette similarité peut être représentée soit par l'intérêt des pairs, leurs contenus, où l'historique des pairs, Haase et Broekstra **[52],** Nejdl et al **[53],** Chiky et al, **[54]** Defude **[55].** Ainsi les pairs avec un contenu similaire sont connectés à un même ultrapair. Ce qui augmente la performance de recherche par rapport à Gnutella. Après avoir donné une vue théorique d'ensemble sur les différents aspects qu'il nous a été nécessaire d'étudier, passons maintenant à la présentation de la partie conception de nos propositions.

Au fait, nous proposons deux méthodes de recherche sémantique pour l'acquisition des résultats :

1-La première méthode consiste à retourner les documents répondants exactement aux requêtes.

Cette proposition se justifie par le fait que si un utilisateur introduit une requête de cinq mots, signifie qu'il souhaite recevoir la liste de documents contenant exactement les cinq mots de la requête. Nous appelons cette méthode 'exact matching'.

2-Conformément à toutes les méthodes de recherche d'information, dans cette deuxième méthode nous cherchons à maximiser l'ensemble des réponses. Ainsi, nous retournons tous les documents qui partagent au moins un mot avec la requête.

Dans toutes ces méthodes, nous appliquons un 'ranking' sur les réponses basées sur les occurrences des termes dans les documents.

Motivations :

Voici quelques points qui nous ont motivés pour le choix de cette stratégie de recherche :

- Chaque individu est caractérisé par le vocabulaire qu'il construit à partir de mots appartenant à ses requêtes reçues.
- En utilisant un mot, on aura à donner un ensemble énorme de résultats.
- Un mot souvent corrélé avec un autre pourra donner un ranking considérable, ce sont les mots les plus populaires qui vont déterminer le ranking de plus, il va s'agir d'un filtre entre les mots.
- Apprentissage du comportement des utilisateurs.

1.2. La recherche dans les systèmes hybrides

Afin d'améliorer l'efficacité de la recherche, un pair doit envoyer la requête tout d'abord à un ensemble de ses voisins, au lieu de l'envoyer directement à son Up. Afin de faire cela sans sacrifier la précision de la recherche, les voisins sélectionnés devrait être ceux qui sont

les plus susceptibles de répondre à la requête. Les connaissances sur le contenu peuvent être représentées par un modèle de langage; ce qui peut être obtenue soit en demandant au pair voisin de leur fournir directement, ou bien par apprentissage de leurs réponses aux requêtes passées, qui ont été déjà traitées. Différentes approches peuvent être utilisées pour développer différents algorithmes de sélection des ressources dans les réseaux p2p hybrides. La plupart des algorithmes présentés ici requièrent la connaissance du nombre de termes de recherche qui doivent être appariés pour une requête « chaines descriptives ». Selon les algorithmes, les termes de requête sont comparés aux noms de documents, à la collection de vocabulaires, ou au contenu du document. Ils sont décrits plus en détail ci-dessous **[47].**

1.2.1. Indexation des fichiers (méthode de hachage)

Nous avons procédé comme suit dans le but de bien préparer la plate-forme de notre base de données :

a) Extraction des index

- Les documents de notre base de données sont analysés afin d'extraire les mots de chaque document. Ensuite, les mots extraits sont stockés avec l'ID du document.

b) Tri des index

- Après l'analyse de tous les documents, l'index inversé est trié alphabétiquement.

c) Regroupement des index

- Une seule entrée par couple (index, document)
- Indication du nombre d'occurrence par document: Term Frequency
- Enlever la redondance en gardant le nombre d'occurrence pour chaque terme.
- Enlever les mots indésirables (stop mots), ce sont les mots inutiles dans notre base de données, afin de gagner de l'espace mémoire et faciliter la recherche.

d) Codages des index

Le but du codage est de manipuler des codes et non pas les chaînes de caractères qui peuvent êtres lourdes et coûteuses en espace mémoire. Nous avons utilisé le codage de la manière suivante :

Le fait de créer un hash code à partir d'un index (terme) peut engendrer un problème de « collision », c'est-à-dire qu'à partir de deux termes différents, la fonction de hachage pourrait renvoyer la même valeur de hash, et par conséquent donner accès à la même position dans la « table de hachage ». Pour minimiser les risques de collisions, il faut donc choisir soigneusement sa fonction de hachage. Nous avons utilisé, en premier lieu, la fonction de hachage cryptographique (SHA-1), et en second lieu, nous avons compressé les SHA-1 codes générés pour deux raisons :

1- Les codes SHA-1 générés sont trop longs. Par exemple, le code généré pour le terme 'acquisition' est 'e6520fc757684242686b284554238b9f1bd95e90',

2-Le changement de l'emplacement d'un seul caractère génère un code différent ; par exemple : pour le terme 'acquisitino', le code généré est : 'ba9430512b7bdc8c9052c0291a067236cec48b64' .

La compression du code SHA-1 nous a permis d'obtenir des codes acceptables. En cas de collision, nous procédons à la gestion de collisions avec chainage linéaire décrit dans le chapitre précédent. A ce stade, nous avons utilisé les tables de hachage (HashTables). La figure 4.1 donne les exemples des codes compressés.

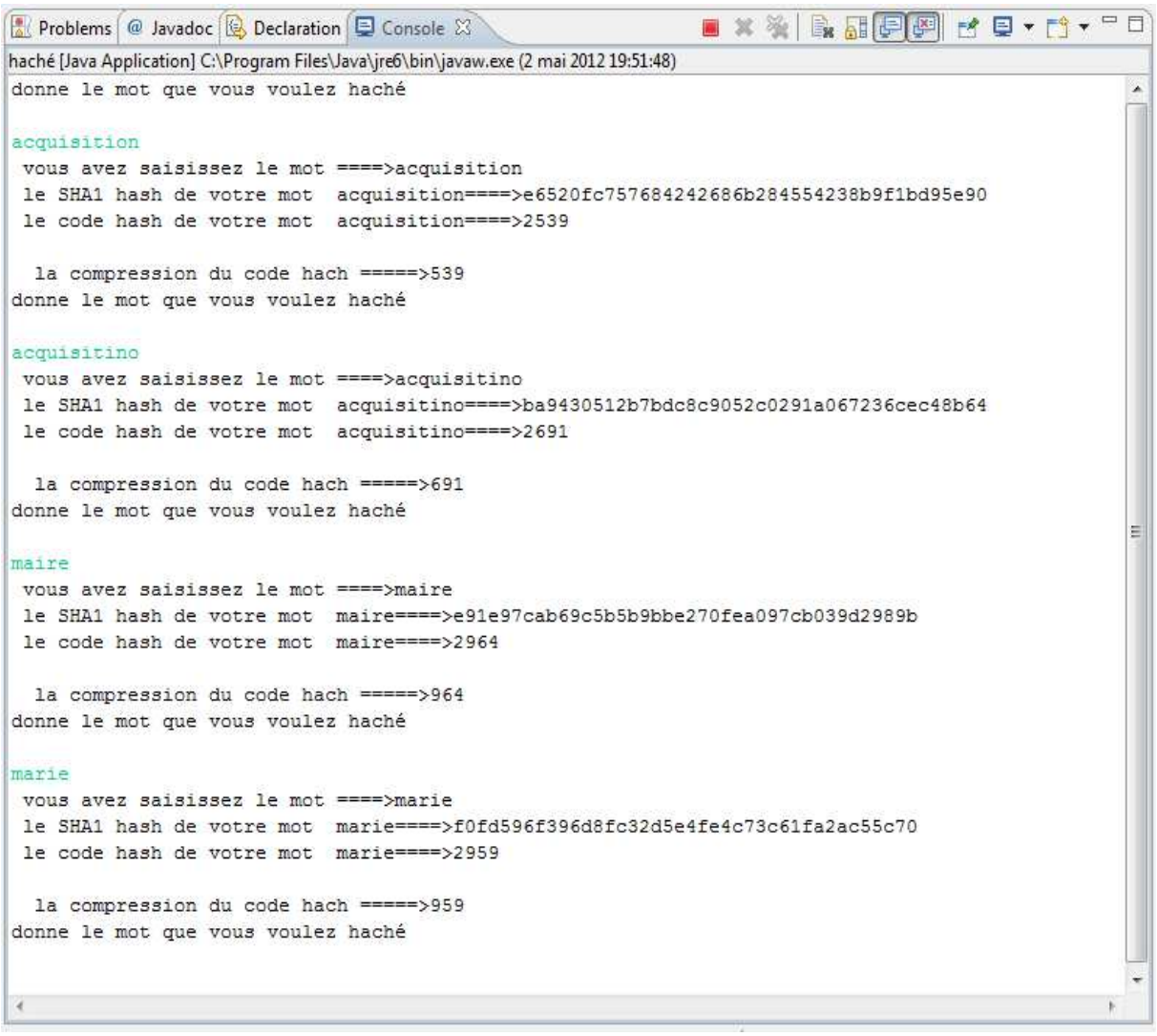


Figure 4.1 Codage Terme

Et par précaution de collision lorsqu'on veut insérer la valeur de l'indice (terme), on crée une liste chainée et on met le terme de la clé dans le premier nœud de la liste chainée et au niveau de ce dernier, on affecte les fichiers qui le contiennent et leurs occurrences.

En cas de collision on aura les traitements suivants :

Si on obtient après le codage, des nombres identiques de code. On crée un nouveau nœud pour chacun de ces codes identiques. Et on insère ces nœuds dans la liste chainée de ce dernier, sans oublier de mettre les informations de chacun d'eux (les fichiers et les occurrences du terme dans chaque fichier) et ainsi de suite comme le montre la figure 4.2 selon l'algorithme 4.1

.

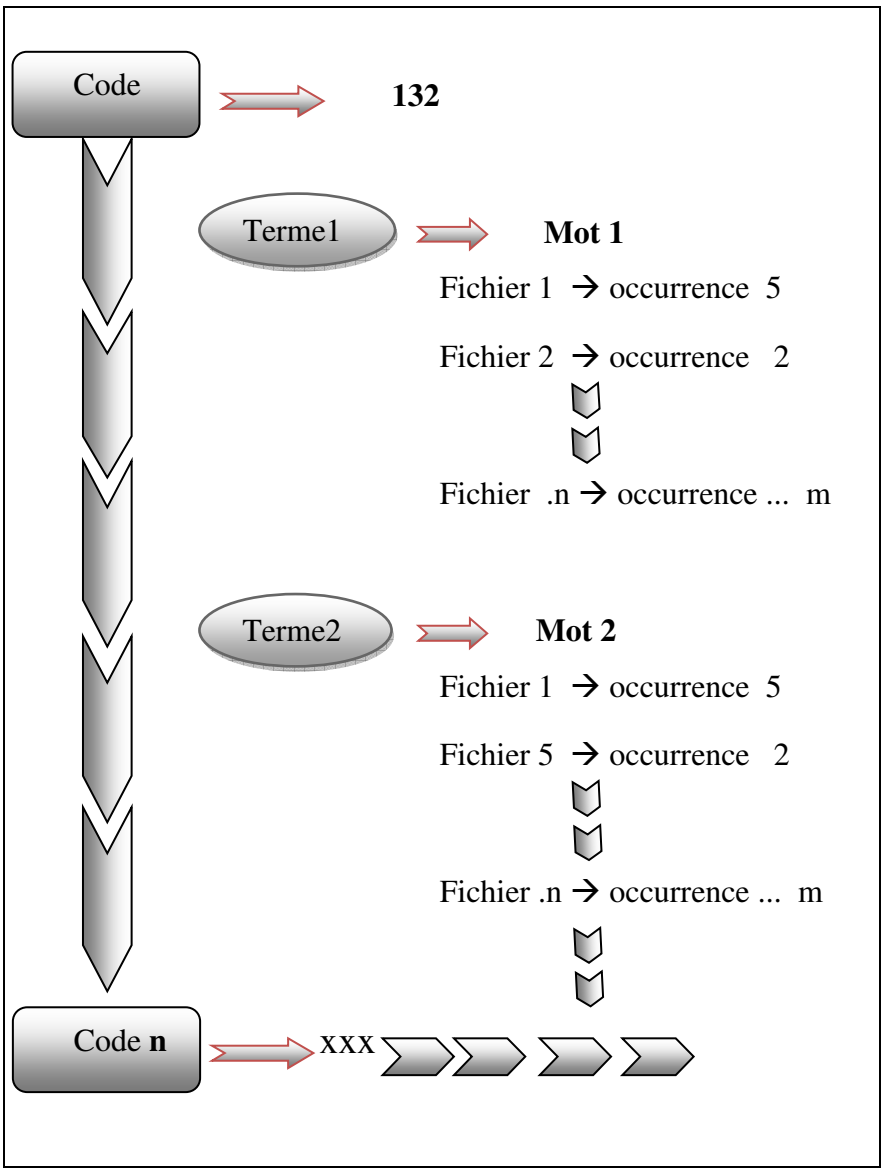


Figure 4.2 Indexation avec la table de hachage

1.2.2. La recherche basée sur le hash du mot

On a utilisé cette méthode pour faciliter la recherche, gagner en espace mémoire et accéder rapidement aux ressources. C'est-à-dire si les codes hash (termes) de la requête sont en correspondance avec les hashs codes des mots de documents. Quand un pair feuille soumet une requête, on procède comme suit pour la résoudre :

1-Coder les termes comme nous l'avions expliqué précédemment ;

2-Le pair feuille cherche dans son répertoire s'il ya des fichiers qui correspondent à cette requête ;

3-Transmettre la requête à l'ultrapair auquel il est attaché ;

4-L'utrapair consulte ses index et transmet la requête aux :

- feuilles attachées à lui ;

-aux ultrapairs voisins.

Comme nous l'avons montré dans **[42]**, nous avions initialisé le TTL à 3 car c'est une valeur assez acceptable pour résoudre une requête dans un système pair-à-pair hybride. Le TTL se décrémente à chaque passage par un ultrapair et lorsqu'il devient nul, la propagation de la requête s'arrête. A chaque passage, le système récolte les résultats selon l'algorithme 4.2 suivant qui donne la collection des résultats.

***Remarque* :**

L'acheminement des requêtes se fait selon l'une des méthodes proposées comme indiqué dans l'introduction de ce chapitre.

1.3. Description des données en entrée

Pour modéliser un système, on doit lui présenter ses paramètres d'entrées. Notre projet à besoin des données qui décrivent :

1.3.1. Les nœuds du système

Il existe deux types de nœuds dans le système :

Les ultrapairs :

Ceux sont les pairs qui ont le plus fort degré. Chaque ultra-pair est relié à un ensemble de pairs appelés feuilles pour lesquels il joue le rôle de serveur centralisé.

Les feuilles :

Ceux sont les pairs avec un faible degré, chaque pair appartient à un cluster de pairs qui sont tous reliés au même Up.

Chaque **nœud** est représenté par un pair ayant les capacités de communiquer, rechercher, partager, apprendre de nouveaux documents, de nouveaux profiles et de nouveaux voisins.

Les voisins :

Un pair *v* est dit voisin du pair *u*, s'il connait son identifiant (on travaille sur un graphe non-

orienté, donc l'un connait l'identifiant de l'autre).
De nouveaux voisins sont ajoutés à cette liste grâce aux échanges des différents messages. Deux pairs qui n'appartiennent pas aux mêmes clusters ne peuvent être voisins.
Les voisins ultrapairs : chaque ultrapair connaît la liste de voisins ultrapairs.

1.3.2. Les index

Il existe deux types d'index dans le système :
a) Index au niveau des feuilles :
Chaque feuille possède un index (InvertedList) contenant tous les termes, les documents où ils apparaissent et les occurrences ainsi que les tf*idf comme indiqué au début du chapitre.
b) Index au niveau des ultrapairs (Up) :
Chaque Up possède :

1) Une liste des Ups voisins ;
2) Une liste inversé contenant, pour chaque mot, la liste des feuilles qui le possède.

1.3.3. Les messages

Il existe deux types de messages qui circulent dans le réseau : message de recherche (requête) et message de réponse; leurs structures sont les suivantes :

- ***Message de Requête*** **:** est de la forme suivante :

@Source	*@Emetteur*	*motsClés*	*TTL*

  - ***@Source*** : représente l'adresse du nœud initiateur de la requête.
  - ***@Emetteur*** *:* l'adresse du dernier nœud qui a acheminé la trame.
  - ***motsClés*** *:* ensemble des mots clés entrés par l'utilisateur ou générés aléatoirement, désignant les fichiers recherchés.
  - ***TTL*** *:* Time To Live, compteur qui sert à limiter la profondeur d'une recherche.

- ***Message de Réponse*** **:** il contient les champs suivants :

@Source	*@Emetteur*	*@Destination*	*Méta-données*

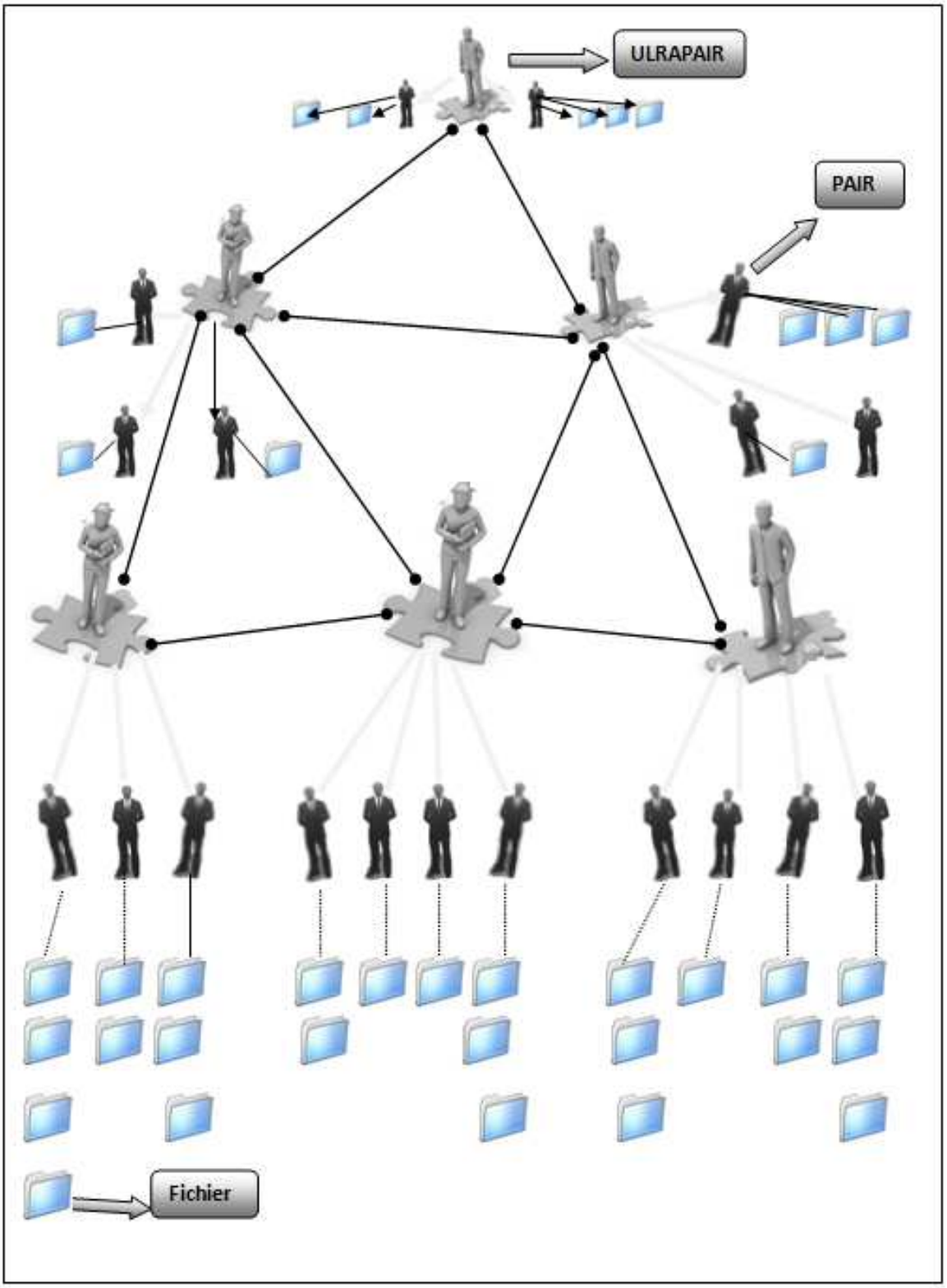


Figure 4.3 Création du réseau

- ***@Source :*** le nœud qui détient l'objet recherché.
- ***@Emetteur :*** le dernier nœud par lequel la réponse a transité.
- ***@Destination :*** est celui qui doit recevoir la réponse, on note que la réponse suit le chemin inverse de la requête.
- ***Méta-données*** : est l'ensemble des triplets qui décrivent les documents retrouvés qui correspondent aux mots de la requête.

Ces Triplets contiennent le nom du fichier retrouvé, les mots de la chaine descriptive qui figurent dans ce fichier, le score et par quel ensemble il a été trouvé.

Une réponse (fichier) est de la forme :

Words F	*Filname*	*Score*

1.3.4. Attribution et réplication des fichiers

L'attribution et la réplication de fichiers consiste à affecter et répliquer l'ensemble des fichiers aléatoirement sur l'ensemble des pairs. Chaque fichier est affecté, aléatoirement, au plus à une feuille.

1.4. Les requêtes lancées dans le réseau

Il existe deux façons avec lesquelles les requêtes sont introduites dans le système.
1-Requête composée directement par l'utilisateur, par une saisie ;
2-Afin d'étudier le comportement de notre simulation, nous avons réalisé un fichier log contenant des requêtes générées aléatoirement et affectées aussi aléatoirement à travers le réseau. L'exécution du fichier permet de tirer les observations et les statistiques.
Afin de générer ce fichier, nous avons pris tous les mots existants dans les fichiers, et généré des requêtes de longueurs variant de 1 à 5 mots.

1.4.1. Distribution des requêtes

Après avoir montré la façon dont nous composons nos requêtes, nous allons maintenant expliquer la façon dont elles vont être affectées aux utilisateurs.

L'affectation des requêtes peut être faite selon deux possibilités :

- Affectation aléatoire : on choisit un nœud aléatoirement parmi l'ensemble de pairs existants dans le réseau pour envoyer la requête.
- Affectation par User : l'utilisateur choisit le pair qui va lancer la requête.

1.5. Evolution du réseau

On parle d'évolution du réseau au sens où comment le réseau augmente en nombre total d'utilisateurs (et non pas en nombre d'utilisateurs connectés) avec le temps.

Deux types de stratégies sont généralement utilisés concernant l'évolution du réseau :

- Certains modèles considèrent qu'initialement tous les utilisateurs sont déjà connectés au réseau.
- Au contraire, d'autres modèles considèrent que le réseau croît de manière linéaire ou exponentielle jusqu'à atteindre un nombre maximal de nœuds.

Dans notre simulation, nous supposons que le nombre total d'utilisateurs est fixé au départ, car nous considérons que la simulation est faite durant un temps limité, ce qui nous permet de supposer que durant cette période aucun nouveau nœud n'arrive dans le système.

1.6. Les données en sortie (résultat)

Notre logiciel fournit en sortie des données décrivant l'évolution de la simulation de notre réseau :

- Pour chaque méthode on donne la liste des requêtes envoyées et un descriptif pour chaque requête (mots clés) et l'initiateur de chacune.
- Le nombre de messages envoyés et le nombre de résultats positifs durant le traitement de chaque requête.
- La liste des réponses d'une requête saisie par l'utilisateur, triée selon le score de chaque fichier, ce qu'on va voir en détail ensuite.
- Des statistiques en courbes, points, histogrammes, et camemberts.

1.7. Etapes de construction du système

Nous présentons un diagramme de cas d'utilisation (figure 4.5) pour récapituler les étapes de construction de notre système.

Configuration : L'utilisateur du logiciel va saisir, à partir de l'interface graphique les paramètres de construction du système (taille du réseau, le nombre de requêtes qui seront lancées pendant la simulation, TTL (profondeur), nombre de nœuds (paires, ultra-pairs).

Simulation : C'est le cas d'utilisation le plus important dans notre logiciel, il permet à l'utilisateur de simuler les protocoles de recherche d'information afin de les évaluer.

Observation : Ce cas d'utilisation permet à l'utilisateur de suivre l'évolution du réseau simulé et différents changements des résultats observés.

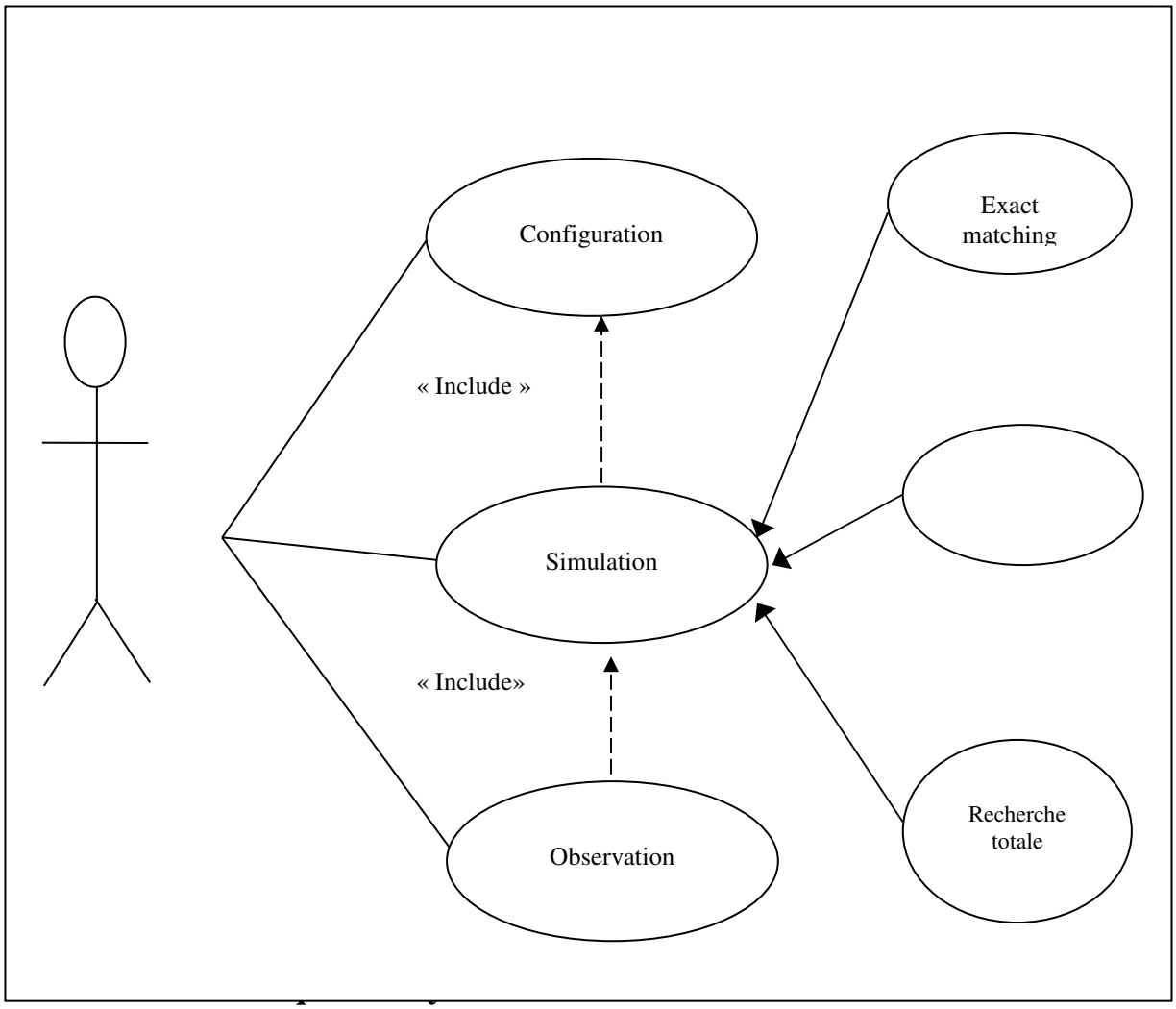


Figure 4.4 Diagramme de cas d'utilisation

Les caractéristiques générales d'un réseau pair-à-pair qui caractérise notre système sont les suivantes :

- **L'auto-organisation :**

Il n'y a pas d'entité responsable de l'organisation du système. Chaque nœud (ultrapairs) crée lui-même son propre voisinage (ses contacts) et choisit aussi ses partenaires de communication.

- **Performance et tolérance aux pannes :**

Il n'y a pas de serveur central, et donc pas de point unique qui serait surchargé par toutes les sortes de communication transitant dans le réseau. Notons que d'autres travaux ont eu comme objet l'étude de la dynamique des réseaux pair-pair **[36]** et ce n'est pas le cas de ce sujet.

Si un ou plusieurs nœuds venaient de tomber en panne au même moment, cela ne causerait pas la chute du système, ce qui le rend moins vulnérable et plus robuste. Toutefois, les crashs (départ brutale) des nœuds provoquent dans certains cas l'échec de l'opération en cours à cause d'un mauvais choix du nombre de nœud ou d'ultra-pair, qui provoquera des problèmes au niveau des ressources (famine), c'est pour cela qu'on doit bien vérifier la distribution des fichiers par rapport au nœud.

- **Expressivité des requêtes :**

L'objectif d'un système de partage de données est de permettre à l'utilisateur de retrouver les données qu'il recherche en un minimum de temps. Afin d'effectuer cette recherche, celui-ci devra spécifier un ensemble de critères, ce qui formera une requête. L'expressivité d'une requête décrit la richesse des critères possibles.

- **Exhaustivité des résultats :**

L'exhaustivité des résultats d'une recherche représente le taux de réponses par rapport au nombre total de données existantes dans le système répondant à la requête. C'est le cas de systèmes client/serveur avec lesquels il est possible d'atteindre des résultats exhaustifs, ce n'est pas le cas dans les réseaux pair-à-pair, à moins d'inonder le système tout entier. Dans notre cas, l'exhaustivité n'est évidemment pas complète. On se contentera des réponses trouvées. Et on donne le choix aux utilisateurs de choisir s'ils veulent une recherche exacte (Les 5 mots de la requête apparaissent), dans ce cas on aura les fichiers qui possèdent ces mots. Les 5 mots de la requête tapée), sinon la recherche sera globale (<= mot de la requête), on obtiendra juste les fichiers qui possèdent au moins un mot de la requête.

- **Coût de recherche :**

Le coût des algorithmes de recherche dans un système décentralisé peut être évalué selon le temps et le nombre de requêtes nécessaires avant d'obtenir les résultats d'une recherche. Ce coût ne doit pas s'accroitre linéairement avec la taille du système, sinon, il serait difficile de passer à l'échelle.

Un bon algorithme de recherche devrait avoir un coût en $O(log^x |N|)$, N étant la taille du système. C'est pour cela qu'on a renforcé notre système par l'utilisation de la table de hachage (accès en O(1)) avec traitement de collision linéaire. En outre, cela minimise le coût de recherche et rend l'accès beaucoup plus rapide que la recherche normale (avec utilisation des tables, liste, matrice,…).

Partie 2 : Expérimentation

2.1. Présentation des méthodes de recherches

Après avoir présenté la façon dont nous avons modélisé notre système, nous allons nous intéressés à la problématique de ce mémoire, à savoir, le routage exact, le routage totale dans les réseaux pair-à-pair.

Avant de présenter les méthodes de routage proposées, nous allons décrire les outils qu'on a développés et qui ont pour but d'améliorer les performances des méthodes présentées

2.2. Les outils du logiciel

- Un buffer de données.
- Une liste mots_fichiers.
- Une liste codesmots_fichiers (Inverted List codeswords_files).
- Une liste codesmots_pairs (Inverted List codeswords_peers).
- Une matrice mot-mot.

2.2.1. Le buffer de données

Chaque pair du réseau possède un buffer de données qui contient les fichiers qui lui sont affectés. Ce buffer est une structure de données dynamique (tableau 4.1).

Ultra-pairs	pairs
U1	**P1, P2, P3**
U2	**P4, P5**
U3	**P6**
…….	
U n	**Pn**

Tableau 4.1 ultra-pair _ pair

Chaque ultrapair du réseau possède un buffer de données qui contient des pairs qui sont affectés à lui. Ce buffer est une structure de données dynamique (tableau 4.2).

Ultra-pairs	pairs	fichiers
U1	P1 P2 P3	F1, F2, F3 F78, F8 F186, F1, F6
U2	P1 P2	F12, F745
U3	P1	F3, F55
…….		
U n	Pn	F1……Fn

Tableau 4.2 Ultra-pair_pair_fichiers

Chaque pair contient des fichiers, les fichiers sont éparpillés selon les pairs de façon à ce qu'on puisse avoir au plus 1 fichiers identiques dans le même ultra-paire.

2.2.2. Les Index

Nous dotons les pairs principalement de deux types de structure de données :

a) Les Index de recherche

Ce sont les index qui permettent de faire de la recherche en exploitant le contenu des pairs. On trouve :

Un index mots_fichiers : on aura l'indexation de manière générale (sans codage), juste le mot et les fichiers auxquels il appartient et le nombre d'occurrence, après avoir traité les mots de notre base de données (extraction, tries, regroupement des mots et séparation en dictionnaire et posting) dans le but de faciliter le codage de ces derniers.

Un index local codemot_fichier : implémenté au niveau de chaque feuille. Il capture pour Chaque codemot, la liste des fichiers qui le contiennent. Cet index nous permet de faciliter la recherche des fichiers à partir des codes de (mots clés) qui sont introduits dans la requête.

Pour notre exemple, le contenu de code_fichier (l'InvertedList) est comme suit :

CODESWORDS	FILES
$m_{0(584)}$	Fichier 1, fichier 6
$m_{1(4871)}$	Fichier 2, fichier 3, fichier 20, fichier 88, fichier 144
$m_{2(112)}$	Fichier 33, fichier 65, fichier 101.
…….	…….
$m_{n(12)}$	fichier 99, fichier 151.

Tableau 4.3 Codesmots-fichiers

Les Index sémantiques

Un profil sémantique est décrit par deux index sémantiques qui sont l'index Codemot_paire et la matrice mot_mot. Ces index sont comme suit :

b) L'index codemots_pairs (Inverted List codesmots_peers (wordscodes_peers)) **:**

Chaque ultrapair appartenant au réseau possède une liste inversée qui capture les mots et les pairs feuilles qui les possèdent ;

Pour notre exemple, le contenu de l'InvertedList est comme suit :

Exemple: Soit m_0, m_1,…, m_n des mots et P_0, P_1,…en, P_n les pairs du cluster. Notre Inverted List sera comme suit :

CODEWORDS	PEERS
m_0(584)	P_4, P_6
m_1(4871)	P_0, P_1, P_3
m_2(112)	P_{20}, P_{88}
…….	…….
m_n(n)	…….

Tableau 4.4 Codemot_pair

La matrice associée au nœud UP1 dans notre exemple est comme suit :

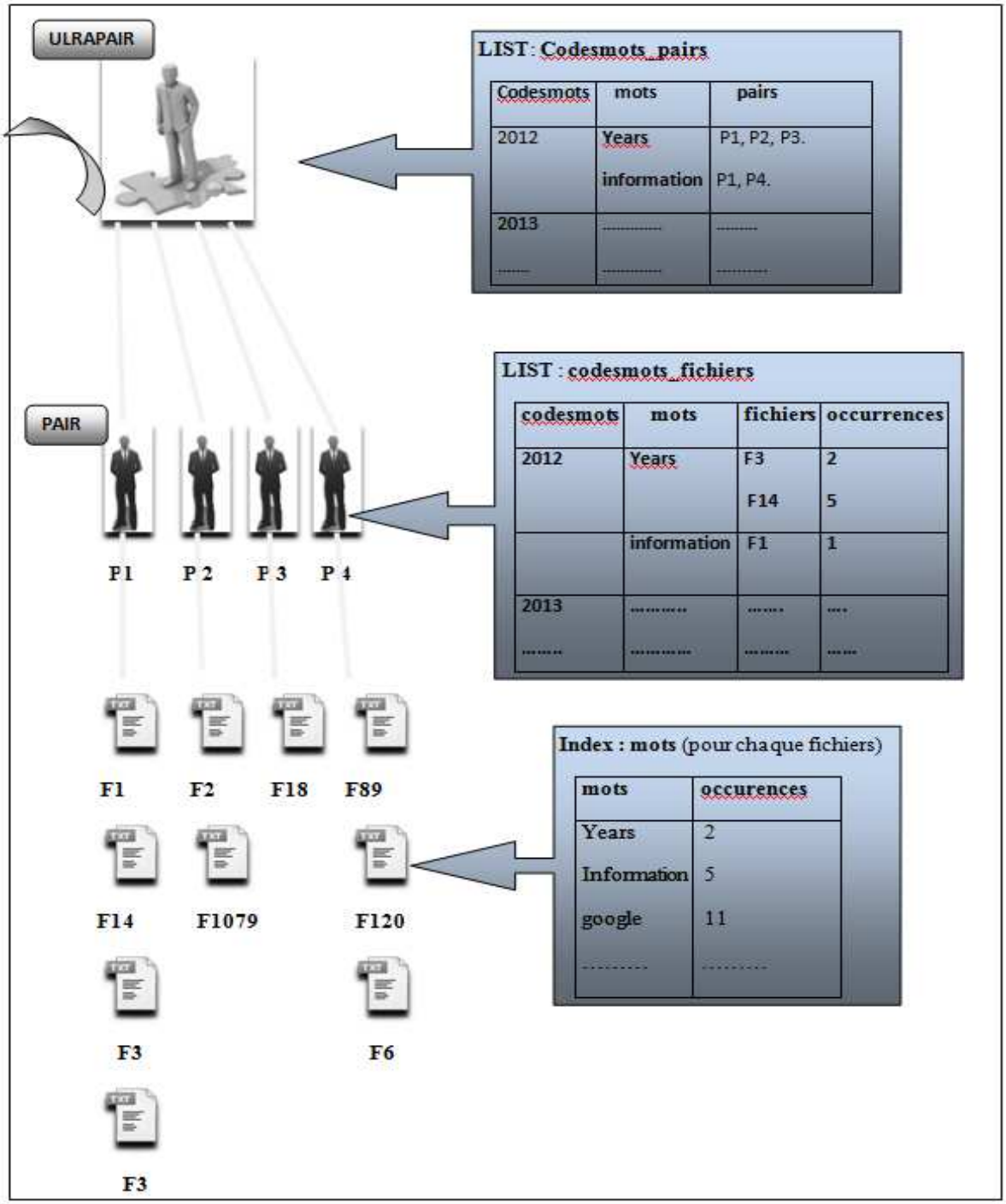


Figure 4.6 Création de la plate-forme du réseau

2.3. La méthode exact matching

La proposition de cette méthode se justifie par le fait que l'utilisateur s'attend, en premier lieu, à ce que les machines de recherche lui retournent les fichiers qui contiennent exactement les mots de sa requête. Encore plus, il serait souhaitable qu'il reçoive les fichiers qui contiennent les mots dans le même ordre de leur saisie.

Le pair feuille envoie sa requête à l'Up auquel il est attaché. L'Up cherche dans son index mot-pair l'ensemble de des feuilles de son cluster qui contiennent exactement les mots de la chaîne. Si cet ensemble n'est pas vide, alors il leur achemine la requête. Cette recherche est faite en profondeur. Pour la recherche en largeur, l'Up achemine cette requête à ses Ups voisins en commençant avec un TTL=3. Chaque Up fait de même : exécute une recherche en profondeur et une autre en largeur et décrémente le TTL s'il n'est pas nul.

2.4. La méthode totale

Il est fréquent qu'un utilisateur fasse entrer des requêtes en souhaitons qu'il reçoive des réponses qui répondent indirectement à ses attentes. Ainsi, il faut lui rendre un maximum de réponses ; c'est-à-dire, chercher toutes les combinaisons possibles entre ses mots. Dans cette partie, nous traitons les requêtes de la même manière que la méthode précédente mais cette fois-ci les Ups cherchent les ensembles de pairs qui partagent au moins un mot avec la requête et l'acheminent de cette requête.

Après avoir terminé la recherche, le pair demandeur passe à la recherche des documents les plus pertinents pour sa requête. Les réponses retournées sont sous la forme de triplet (Nom du Fichier, Mots, Score) où Mots contient l'ensemble de mots qui figurent dans la requête ayant été trouvés dans le document, et Score **[40]** est la popularité de ce dernier. Il est calculé par soit l'occurrence du mot ou par la pondération du TF*IDF (poids).

L'un des problèmes principaux des systèmes P2P non structurés comme GNUtella est que les requêtes sont envoyées à un très grand nombre de nœuds (broadcast). Proposer à l'utilisateur les fichiers souvent associés à une requête ou un téléchargement permet d'éviter cette grande surcharge du réseau, dans la mesure où l'on connait à l'avance les ressources à extraire. Il suffit d'enrichir les décisions de routage avec des informations complémentaires extraites de la relation utilisateurs-mots (chaque individu est caractérisé par son vocabulaire) et de celle mots-mots (la cooccurrence des mots).

L'exploitation efficace d'informations sur un système P2P nécessite des mesures adaptées pour filtrer, classer et ordonner les ressources en fonction de différents critères. Un tel critère d'ordonnancement et de sélection peut dépendre soit de la pertinence d'une réponse pour une requête donnée, soit de la popularité des résultats trouvés. Nous avons choisi d'adopter cette dernière méthode de classement pour définir l'importance d'une réponse basée sur l'analyse du comportement des utilisateurs.

2.7. Edition des résultats

On donnera le privilège aux utilisateurs de choisir la recherche qui sera comme suit :

Recherche exact :

Les réponses retournées sont sous la forme de triplet (Nom du Fichier, Mots, Score) où Mots contient tous les mots qui figurent dans la requête trouvée dans le document, et si on a plusieurs triplets du résultat exact, dans ce cas on tri cette dernière selon le score le plus élevé (les plus grands scores pour les fichiers les plus populaires du document). C'est à ce niveau que l'index résultat est mis à jour.

Recherche totale :

Dans cette recherche on aura des résultats en vue globale selon l'apparition du terme *i* de la requête dans les fichiers résultats, on ne prend pas en considération le nombre des mots du triplet (Nom du Fichier, Mots, Score). Il suffit juste de trouver les mots (<=) aux mots de la requête, et afficher ces fichiers
L'édition des résultats pour la méthode sémantique se fait comme dans le cas de recherche totale.

3. Conclusion

Nous avons présenté dans ce chapitre, les détails de conception de nos méthodes de routage. Les méthodes présentées sont le fruit de travaux antérieurs. L'objectif est de réaliser principalement une recherche sémantique qui a pour objectif de réduire le coût de localisation dans les systèmes hybrides non sémantiques.
Le chapitre suivant présente les détails de notre implémentation.

1. Introduction

Après avoir donné nos algorithmes pour faire la recherche des requêtes dans notre système pair à pair, nous donnons et nous commentons dans ce chapitre nous passons à la partie implémentation de notre application, aussi nous présentons les différents résultats obtenus. Tout d'abord nous exposons l'ensemble d'outils et langage que nous avons utilisés afin d'arriver à nos objectifs.

2. Environnement de travail

Nous avons expérimenté nos algorithmes sur une machine ASUS ayant un processeur Intel(R) core I3, CPU Intel Core i3 2310M, 2.1GHz, doté d'une mémoire vive de 4,00 GO. Disque dure 500GO.
L'environnement de notre travail est *Microsoft Windows Seven (7)* avec l'EDI Java (Environnement Intégré de Développement) JBuilder 9, NetBeans IDE 7.0.1.

Environnement logiciel

Java est un langage de programmation à usage général, évolué et à objet dont la syntaxe est proche du langage C. Ses caractéristiques ainsi que la richesse de son écosystème et de sa communauté lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est notamment largement utilisé pour le développement des services web.

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans**[56]** permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).

3. Avantage du langage choisi : [56]

Comme principaux avantage de java, nous pouvons citer :

 Java est un langage « objet », par opposition au C++ qui lui est « orienté objet », et qui autorise la programmation procédurale.

 Tous les éléments de java à l'exception de quelques types de bases tels que les nombres, sont des objets.

 La conception orientée objet présentant de nombreux avantages pour les projets sophistiqués, elle a remplacé les techniques structurées précédente.

 Il est beaucoup plus facile d'obtenir un code sans erreur à l'aide de Java qu'avec C++. Selon certaines estimations, le code C++ contient au moins une erreur toutes les cinquante lignes. Les concepteurs de Java ont beaucoup réfléchi à la raison pour laquelle le code C++ contient autant d'erreurs. Cette réflexion les a amenés à ajouter dans Java des fonctions destinées à éliminer la possibilité de créer du code contenant les types d'erreurs les plus courants. Le choix des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs. Par ailleurs, toutes les variables sont typées et il n'existe pas de conversion automatique qui risquera une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser une caste ou une méthode statique fournie en standard pour la réaliser.

 Allocation de la mémoire pour un objet est automatique à sa création. Java récupère systématiquement la mémoire inutilisée grâce au (Garbage Collector) qui restitue les zones de mémoire laissées libres à la destruction des objets.

 La sécurité fait partie intégrante du système d'exécution et du compilateur. Quand un programme Java plante, il ne menace pas le système d'exploitation. Il ne peut pas y avoir d'accès direct à la mémoire.

 En outre, Java présente l'intérêt de pouvoir être utilisé comme un langage de programmation polyvalent pour développer des logiciels capables de fonctionner sur différentes plates-formes.

4. Les Package Utilisés

4.1. JFreeChart [37]

L'avantage du langage de programmation java est que l'on peut trouver sur Internet de nombreuses API gratuites, qui peuvent être utilisées dans nos applications. Une **API** (Application Programming Interface) contient toutes les interfaces et classes nécessaires au fonctionnement de la fonctionnalité. L'API JFreeChart, est une fonctionnalité qui permet la génération des graphes.
Le JFreeChart propose de nombreux types d'affichage de graphiques possibles :

- Histogrammes,
- Camemberts,
- Graphique d'évolution,
- Courbes, …etc.

Le JFreeChart possède deux avantages certains :

- Tout d'abord sa gratuité,
- On peut visualiser et modifier les codes sources de l'API,
- Fonctionnalité très puissante qui propose de nombreuses classes et méthodes,
- L'API est continuellement mise à jour avec de nouvelles fonctions.

Dans la suite de ce chapitre, nous aborderons les différentes fenêtres de notre application ainsi que les statistiques afférentes aux différents algorithmes étudiés.
En ce qui concerne la partie du fenêtrage, nous projetons premièrement la page d'accueil, deuxièmement la fenêtre principale et à la fin une série d'écrans des différents algorithmes implémentés.
Et pour nos statistiques, ce sont des résultats obtenus à partir des études comparatives entre les méthodes élaborées après plusieurs suites d'expérimentations.

5. Ecrans générés et Résultats

5.1. Fenêtre d'Accueil et fenêtre principale

Notre première fenêtre 5.1 est le point d'accès à notre application; elle permet à l'utilisateur d'accéder à notre application.

Figure 5.1 Page D'accueil

5.2. Description du logiciel

5.2.1. Interface

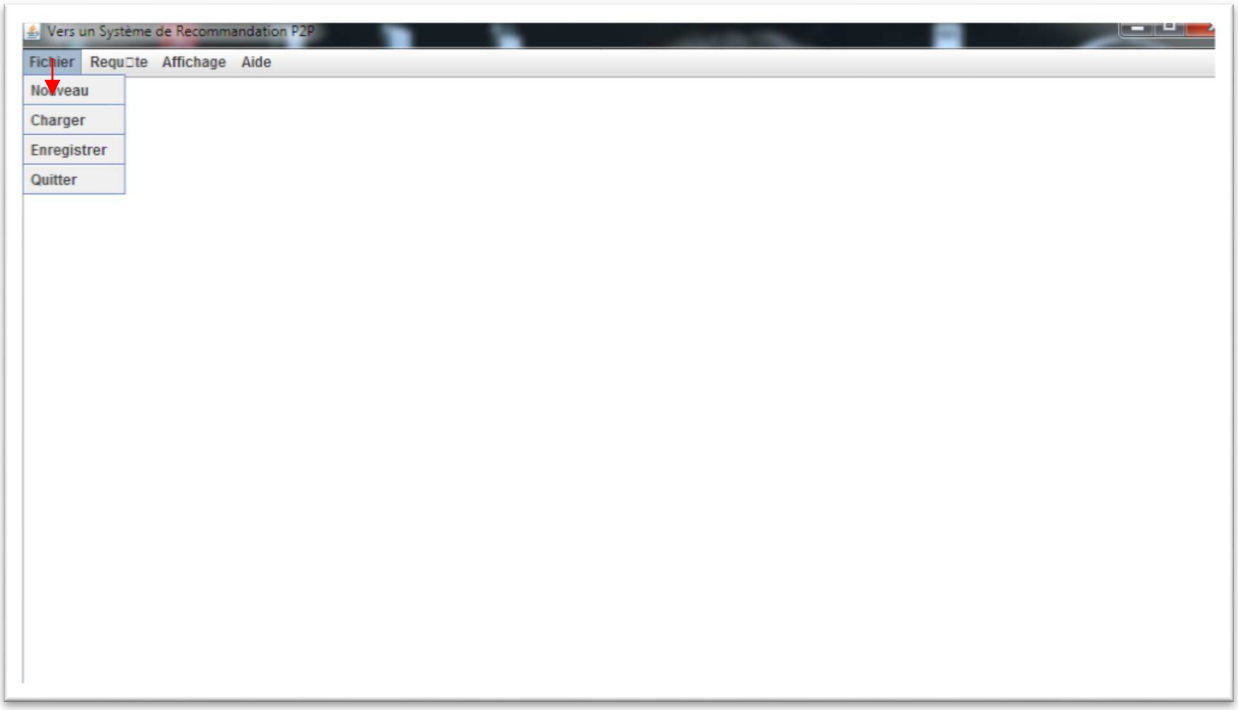
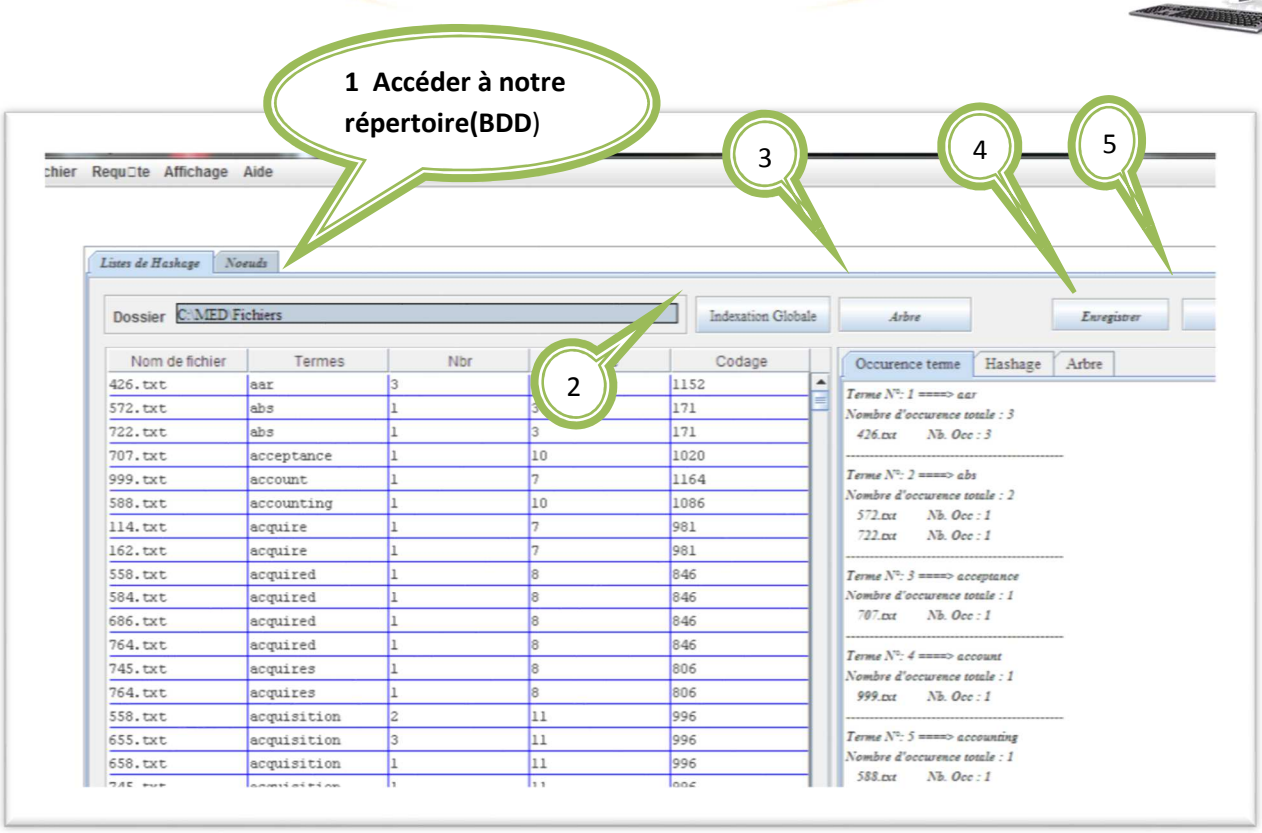


Figure 5.2 Fenêtre Principale

La fenêtre 5.2 accessible à partir de la fenêtre précédente permet à l'utilisateur de choisir s'il veut accéder à notre base sous format XML en cliquant sur le bouton **charger** ; cette dernière va transformer notre base de données du format XML en format TXT. Ou bien en cliquant sur **nouveau** l'utilisateur va accéder à notre base directement en format TXT et donner l'accès à la création de notre plate forme de travail.



Nom de fichier	Termes	Nbr		Codage
426.txt	aar	3		1152
572.txt	abs	1	3	171
722.txt	abs	1	3	171
707.txt	acceptance	1	10	1020
999.txt	account	1	7	1164
588.txt	accounting	1	10	1086
114.txt	acquire	1	7	981
162.txt	acquire	1	7	981
558.txt	acquired	1	8	846
584.txt	acquired	1	8	846
686.txt	acquired	1	8	846
764.txt	acquired	1	8	846
745.txt	acquires	1	8	806
764.txt	acquires	1	8	806
558.txt	acquisition	2	11	996
655.txt	acquisition	3	11	996
658.txt	acquisition	1	11	996

Figure 5.3 Indexation de la Base de Données

La **fenêtre 5.3** présente tout l'aspect de l'indexation, nous décrivons dans la Suite les utilisations des différents boutons de cette fenêtre.

2 : Ce bouton a pour effet de :

- **Extraction des index des documents :**
  - extraction de chaque occurrence
  - enlever les mots indésirables (stop mots)
  - enlever la redondance
  - mémorisation du document pour chaque occurrence

- **Tri des index :** par ordre alphabétique
- **Codages des Index**
  - En utilisent notre fonction de hachage

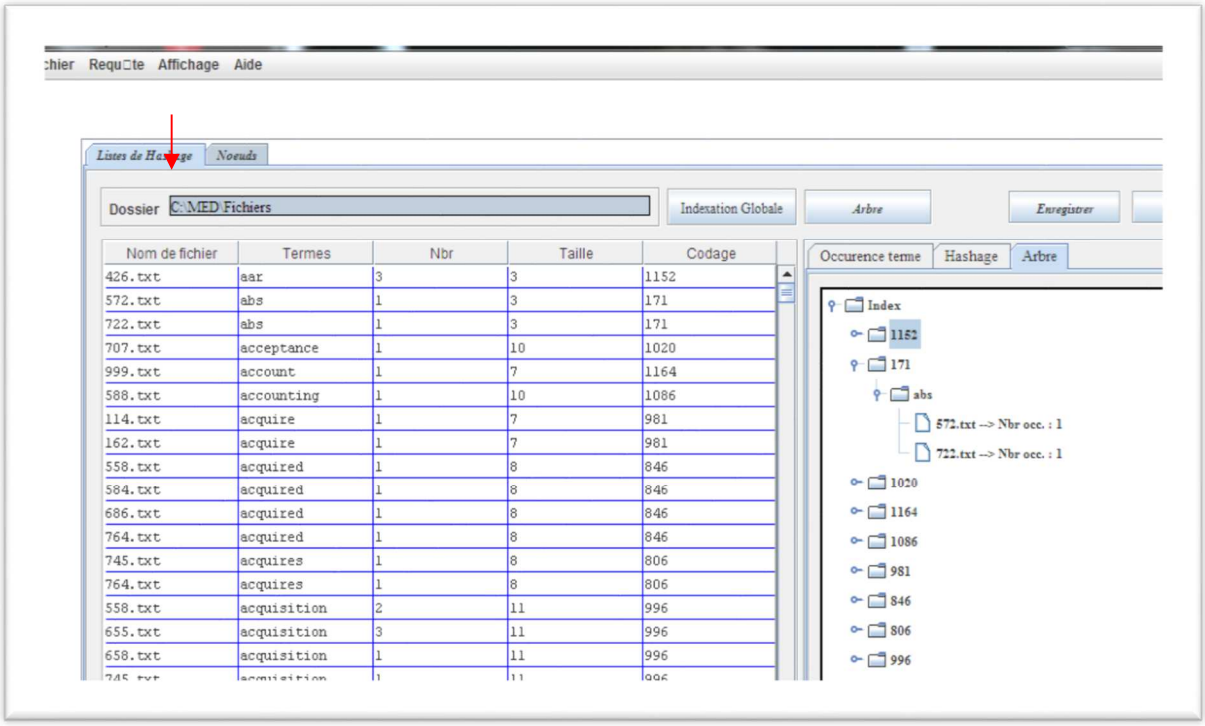
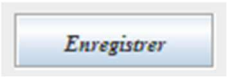


Figure 5.4 Affichage en Forme D'arbre

3 : Ce bouton a pour effet de :

Créer une sorte d'arbre d'index (**fenêtre 5.4**) pour mieux voir l'effet de notre méthode d'indexation.

Chaque code choisi va donner les détails de ces termes, occurrence et l'emplacement de son fichier pertinent.



4 : Ce bouton a pour effet de :

Enregistrer les étapes précédentes pour donner l'opportunité à l'utilisateur de le sauvegarder en cas d'une nouvelle réutilisation.la Fenêtre 5.5 est générée en cliquent sur ce bouton

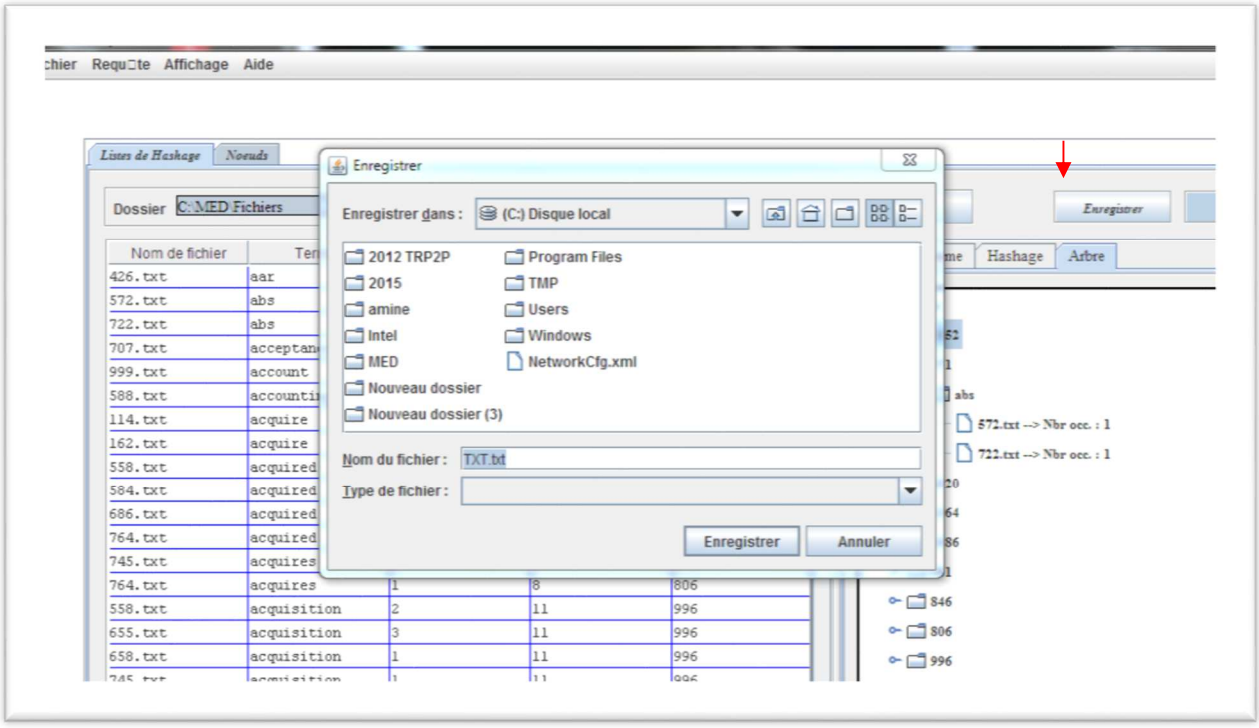


Figure 5.5 Ssauvegarder le Fichier D'indexation

5 : Ce bouton a pour effet de : générer la Fenêtre 5.6

Pour donner la main à l'utilisateur d'accéder au fichier sauvegardé précédemment dans le but de gagner du temps et de poursuivre les étapes de l'application.

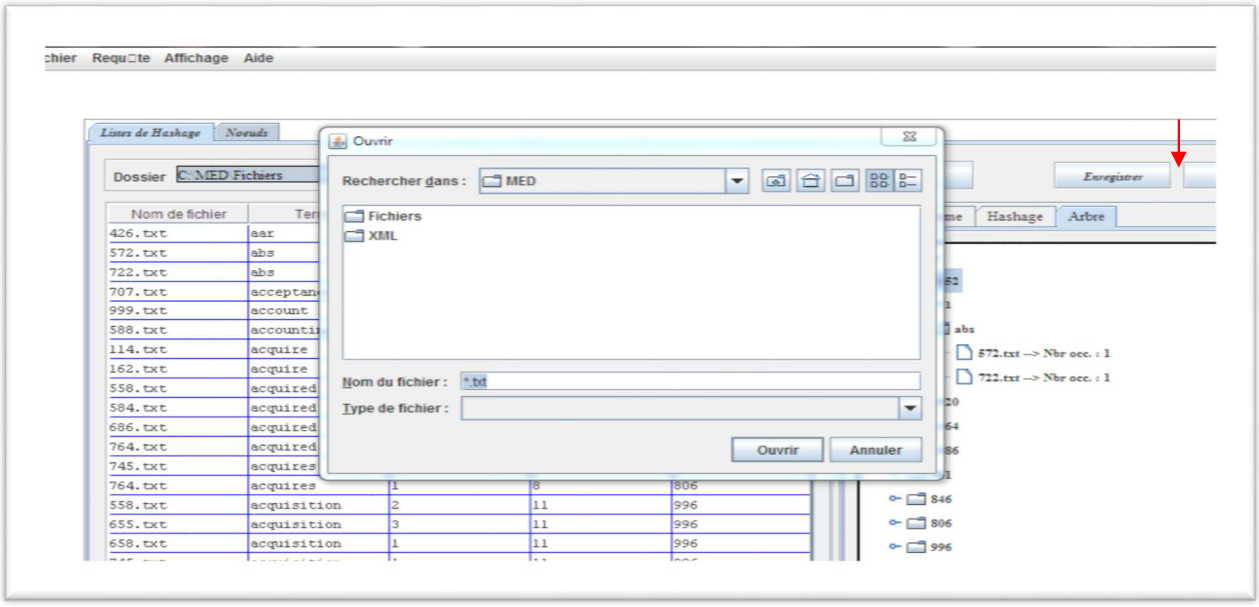


Figure 5.6 Ouverture du Fichier D'indexation Sauvegardé

5.3 Création de la plate-forme du réseau

- Création des pairs, en affectant à chacun un nombre aléatoire d'objets. (Fichiers)
- Création des ultra-pairs, en affectant à chacun un nombre aléatoire de pairs. Chaque pair est affecté, aléatoirement, au plus à un ultra-pair.

Après avoir crée les pairs et les objets, une association est faite. Ici, chaque ultra-pair va maintenir un certain nombre de pairs et ce dernier va aussi maintenir un certain nombre d'objets avec leurs valeurs, en traçant une liste triée.la Fenêtre 5.7 présente les éléments de création de l'environnement du système.

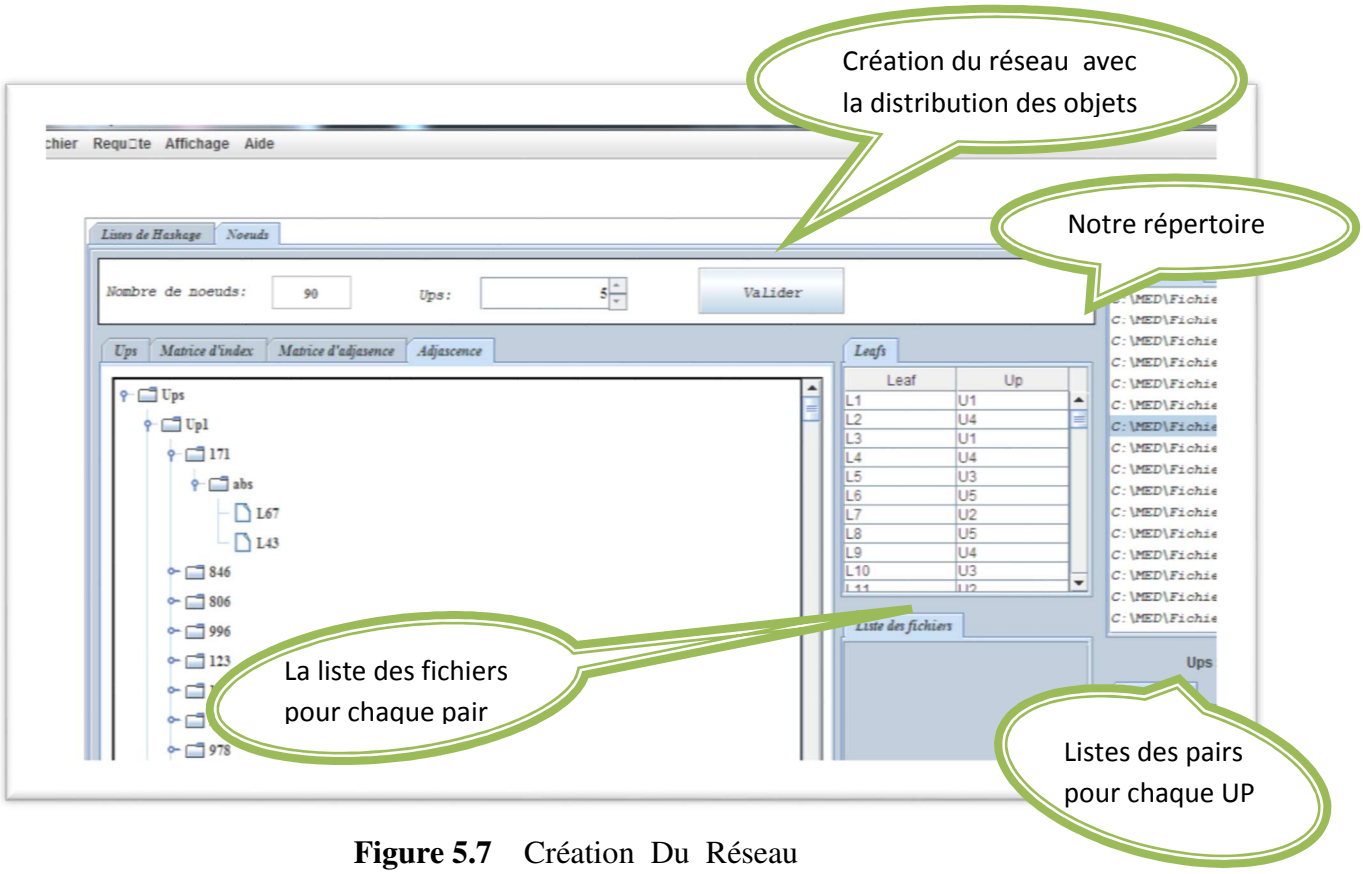


Figure 5.7 Création Du Réseau

Les ultra-pairs :

Ceux sont les pairs qui ont le plus fort degré. Chaque ultra-pair est relié à un ensemble de pairs appelés feuilles pour lesquels il joue le rôle de serveur centralisé.

Les feuilles (pairs) :

Ceux sont les pairs avec un faible degré, chaque pair appartient à un cluster de pairs qui sont tous reliés au même Up.

Nombre de noeuds: **Paires** : l'utilisateur choisi le nombre de nœuds feuille (leafs).

Ups: **Ultra-Paires** : l'utilisateur choisi le nombre de nœuds racine.

Toutefois, les crashs (départ brutal) des nœuds provoquent dans certains cas l'échec de l'opération en cours à cause d'un mauvais choix du nombre de nœud ou d'ultra-pair, qui provoquera des problèmes au niveau des ressources (famine), c'est pour cela qu'on doit bien vérifier la distribution des fichiers par rapport au nœud. De refaire l'action pour bien avoir une bonne distribution du réseau.

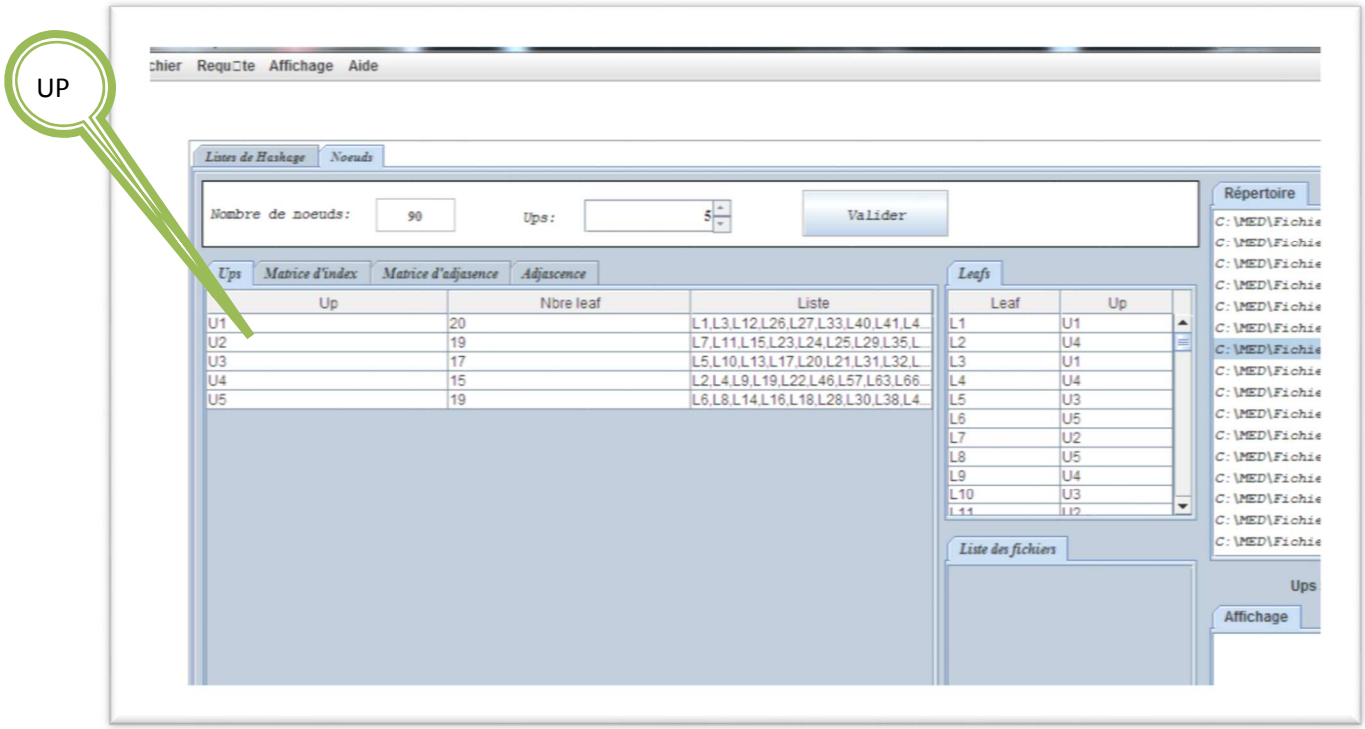


Figure 5.8 Distribution pairs-Fichiers

La figure 5.8 présente la distribution de différents fichiers sur les pairs

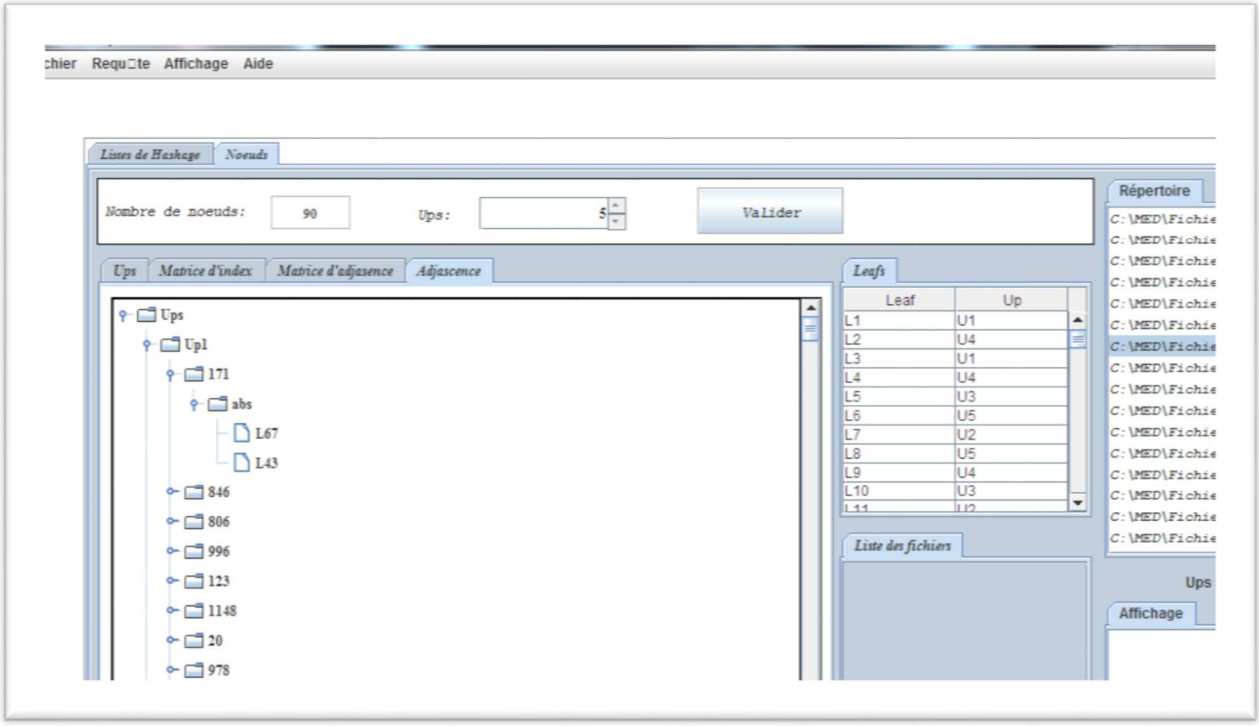


Figure 5.9 Indexation Ultra-pairs (codemot-Terme-pairs)

La figure 5.9 présente les index au niveau des ultra-pairs sous forme de cluster

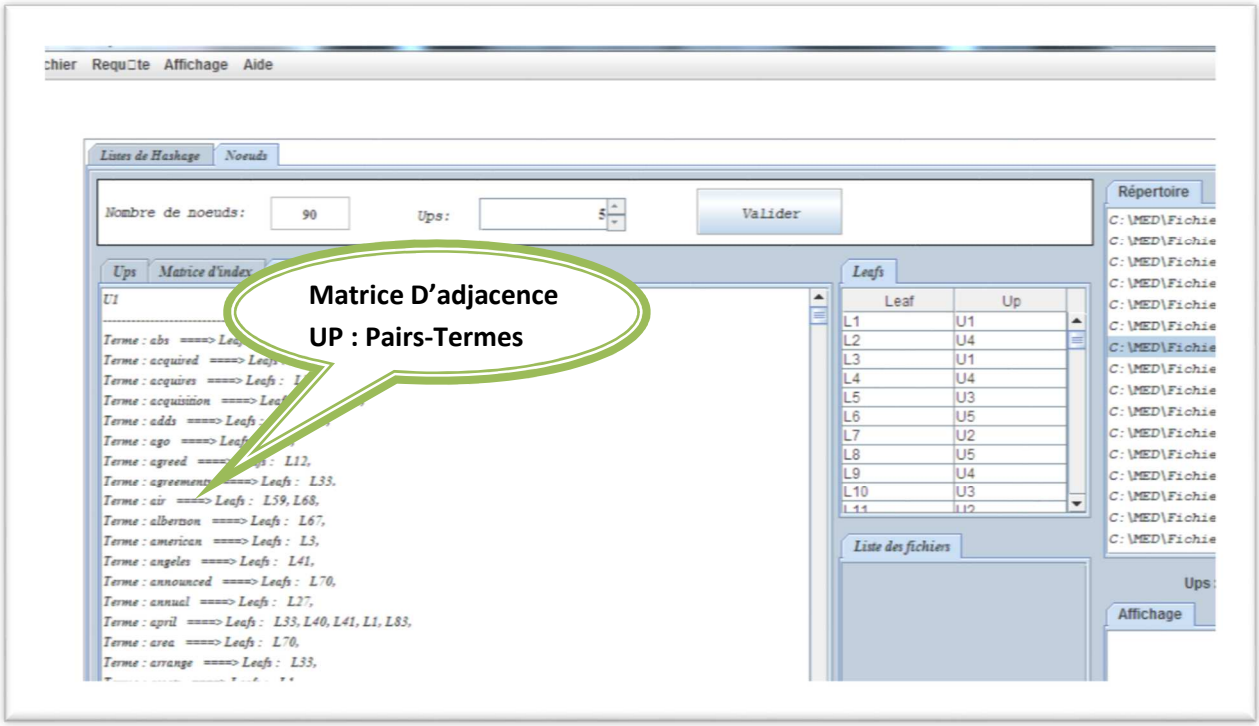


Figure 5.10 Distribution Ultra-pairs - pairs

La figure 5.10 présente l'index Terme-Pair (la liste d'adjacence Terme-pair) grâce a la quel se feras le routage

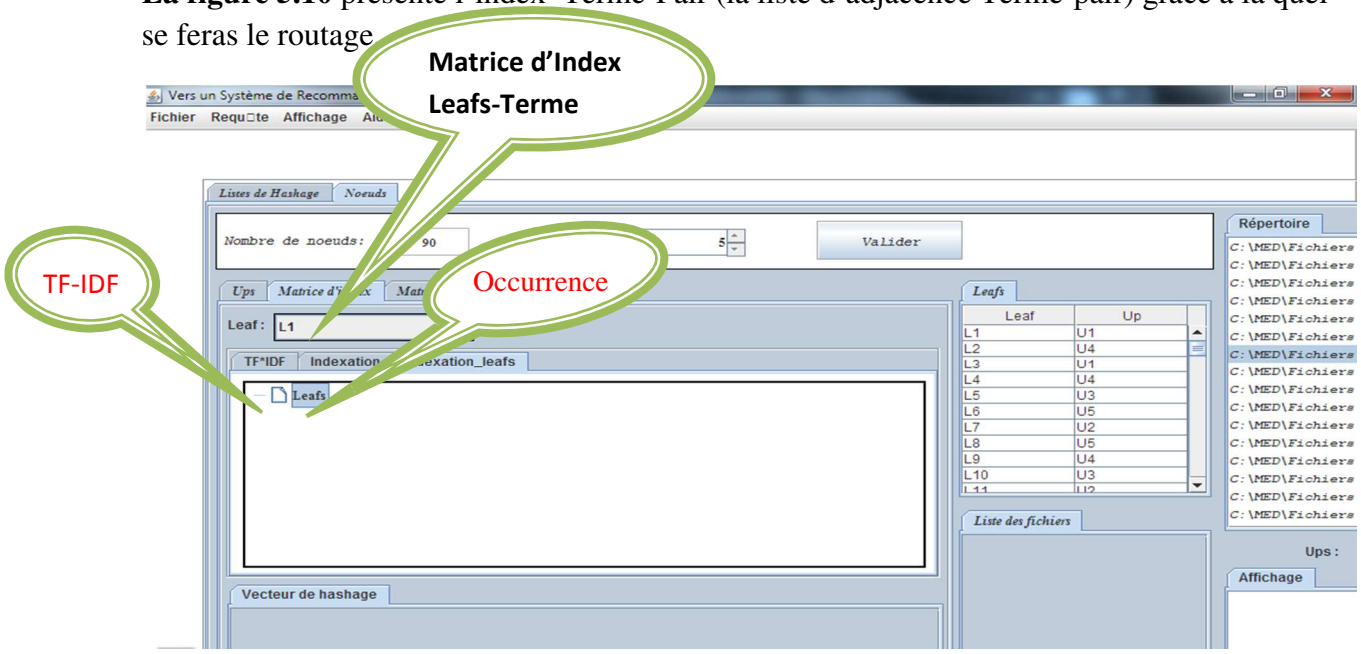


Figure 5.11 Matrice D'index pairs-Fichiers

5.4. Notre approche

5.4.1. Modes de Recherche

5.4.1.1 Recherche (méthode normale) :

Recherche exacte :

Elle renvoi les résultats qui correspondent exactement aux mots tapés par l'utilisateur.

Recherche globale :

Elle renvoi tous les résultats possibles qui correspondent aux mots tapés par l'utilisateur.

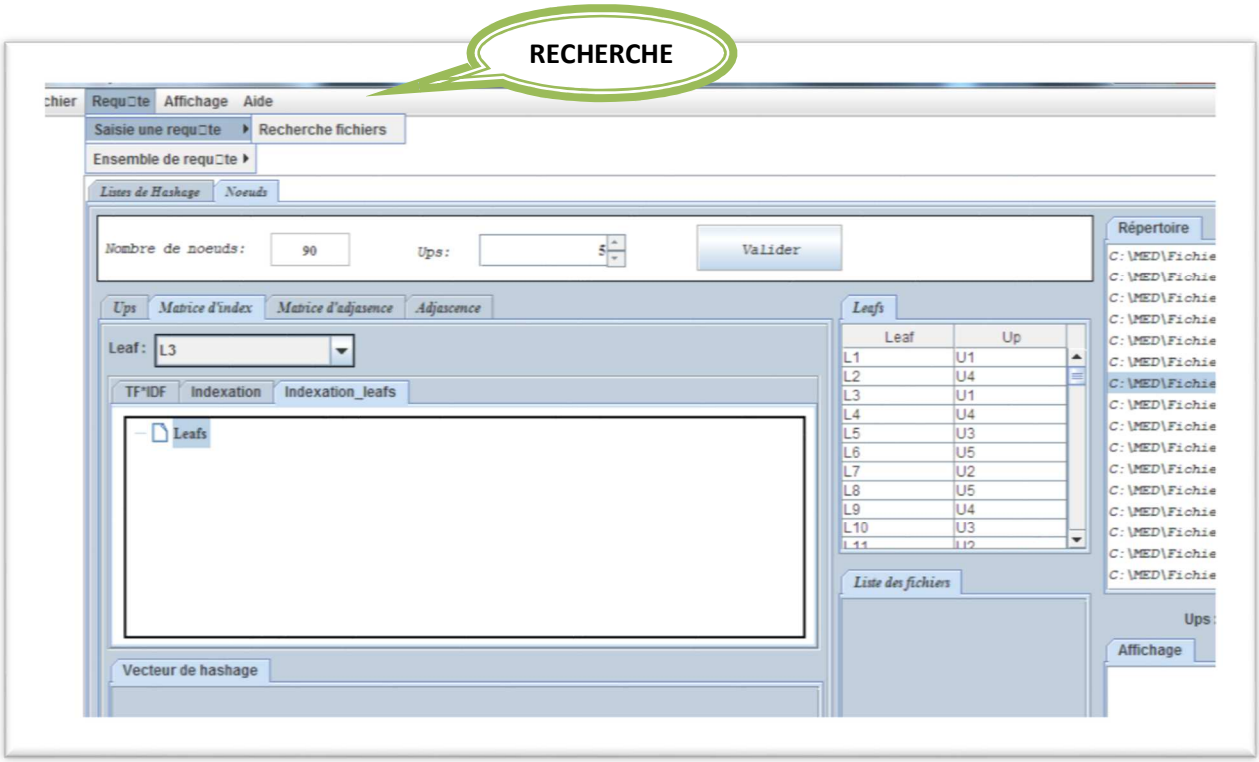


Figure 5.12 Type de Recherche

5.4.2. Distribution des requêtes

Il existe deux façons avec lesquelles les requêtes sont introduites dans le système.

1-Requête composée directement par l'utilisateur, par une saisie Saisissez la requête s.v.p : (Figure 5.13)

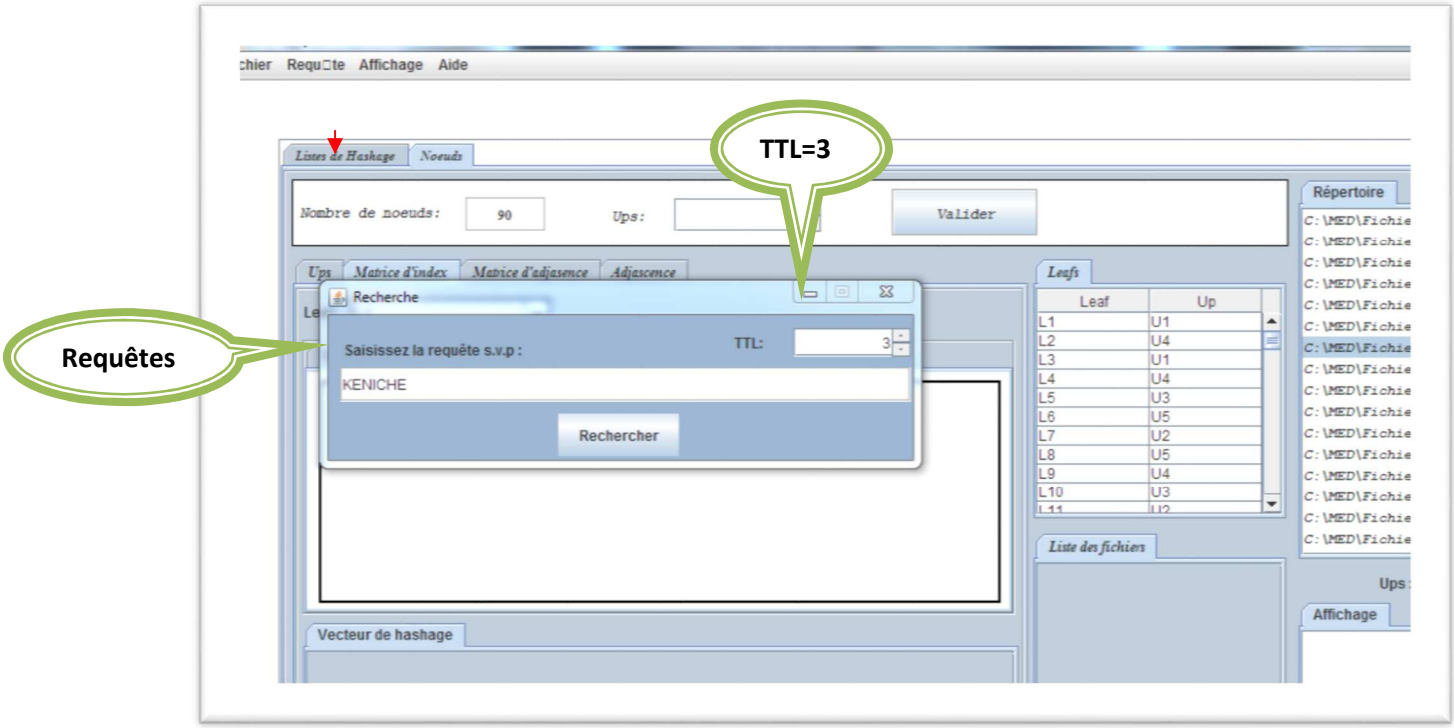


Figure 5.13 Requête Saisie Manuellement

TTL: ***TTL*** *:* Time To Live, compteur qui sert à limiter la profondeur d'une recherche.

La recherche dépend de la valeur du **TTL** pour contrôler la propagation d'une requête. Cependant, il n'est pas facile de choisir un TTL approprié. Si le TTL est trop élevé, la requête va surcharger le réseau inutilement; s'il est trop bas, le pair peut ne pas trouver l'objet même s'il existe une copie quelque part, par défaut nous avons choisi de donner au TTL la valeur 3 selon. **[42]**

2-Afin d'étudier le comportement de notre simulation, nous avons réalisé un fichier log contenant des requêtes générées aléatoirement et affectées aussi aléatoirement à travers le réseau. L'exécution du fichier permet de tirer les observations et les statistiques.

Afin de générer ce fichier, nous avons pris tous les mots existants dans les fichiers, et généré des requêtes de longueurs variant de 1 à 5 mots.

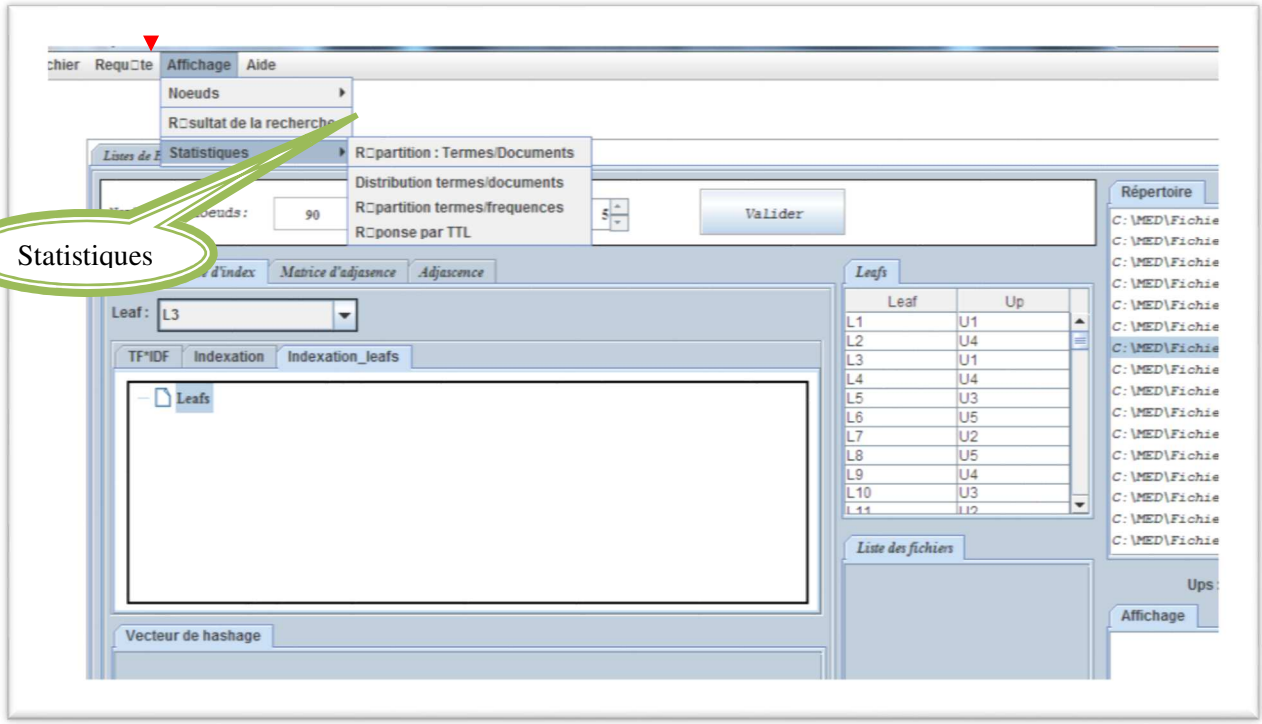


Figure 5.14 Accès aux Résultats et Statistiques

5.4.3. Résultat de la Recherche

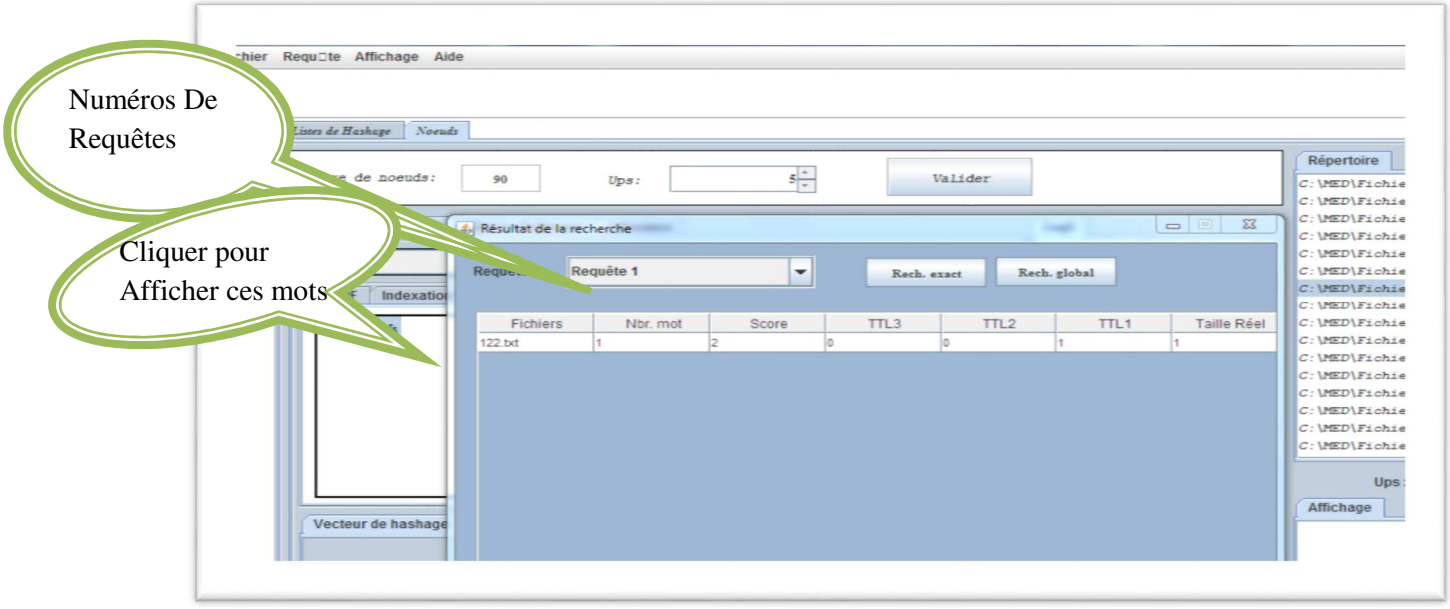


Figure 5.15 Fenêtre du Résultat

5.4.4. Résultat retourné

Les Résultats retournés dont l'ensemble des cinqdruplets qui décrivent les documents retrouvés qui correspondent aux mots de la requête.
Ces cinq druplets contiennent le nom du fichier retrouvé, les mots de la chaine descriptive qui figurent dans ce fichier, le score et les fichiers ou ils ont été trouvés.

Une réponse (fichier) est de la forme :

Fichiers	*nbrmot*	*Score*	*TTl*	*Taille réel*

- **Fichiers :** contient le nom de fichiers où apparait les mots de la requête
- **Score :** c'est la somme d'occurrence des mots de la requête choisie
- **TTL** : Chaque case du **TTL** informe l'utilisateur dans quelle profondeur de TTL le résultat a été trouvé
- **La taille réelle** : est le nombre de mots réel d'une requête choisie

Exemple : (la figure 5.22) donne un exemple de fichier possédant les mots de la recherche

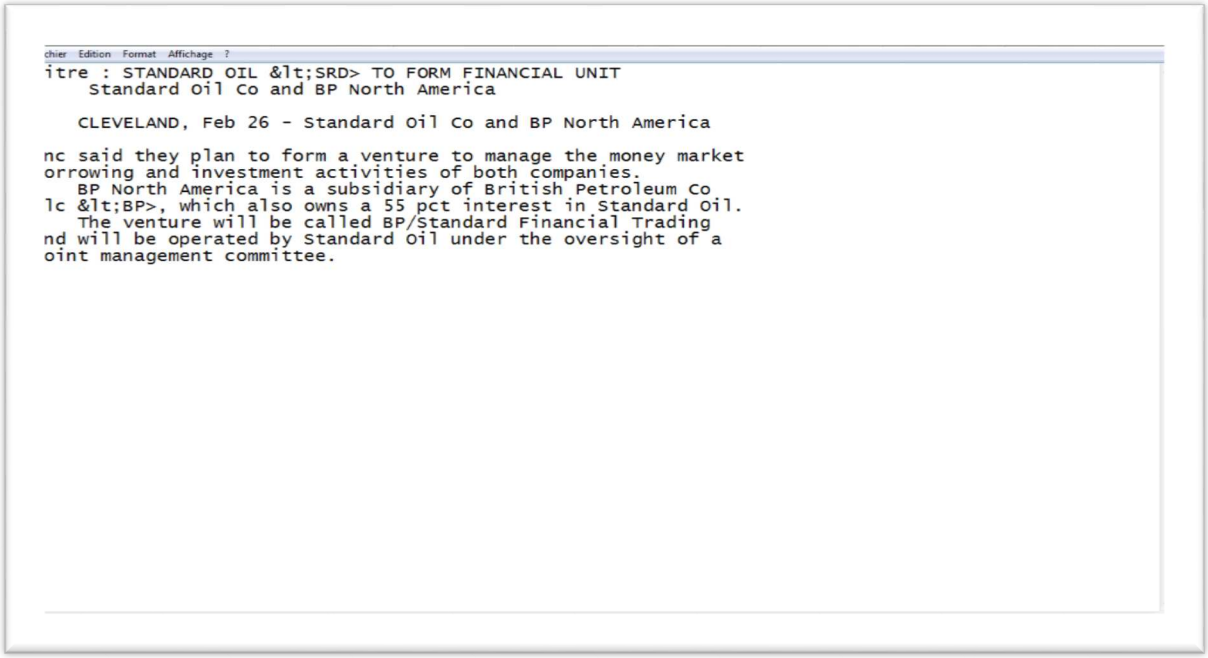


Figure 5.16 Fichier 128

6. Expérimentation à base de données réelles

Dans la partie suivante, nous exposerons les différents résultats obtenus ainsi que les méthodes utilisées. La figure 5.23 présente la répartition des termes sur l'ensemble des

fichiers. Nous remarquons qu'un ensemble réduits de termes (03 termes exactement) apparaissent dans beaucoup de fichiers alors que le reste apparaisse dans moins de documents ce que confirme la figure 5.24 qui présente la distribution des termes sur les documents ; ainsi on retrouve de façon explicite la distribution en loi de puissance . L

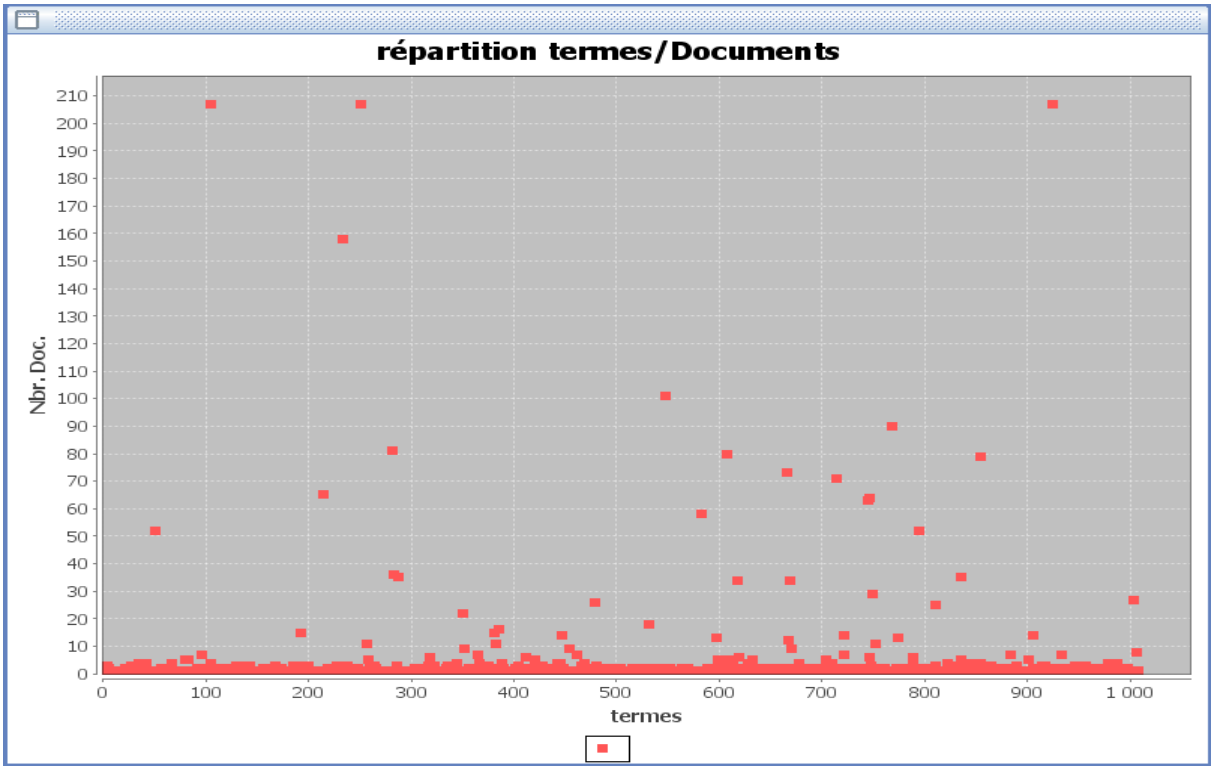


Figure 5.17 Répartition Termes-Documents

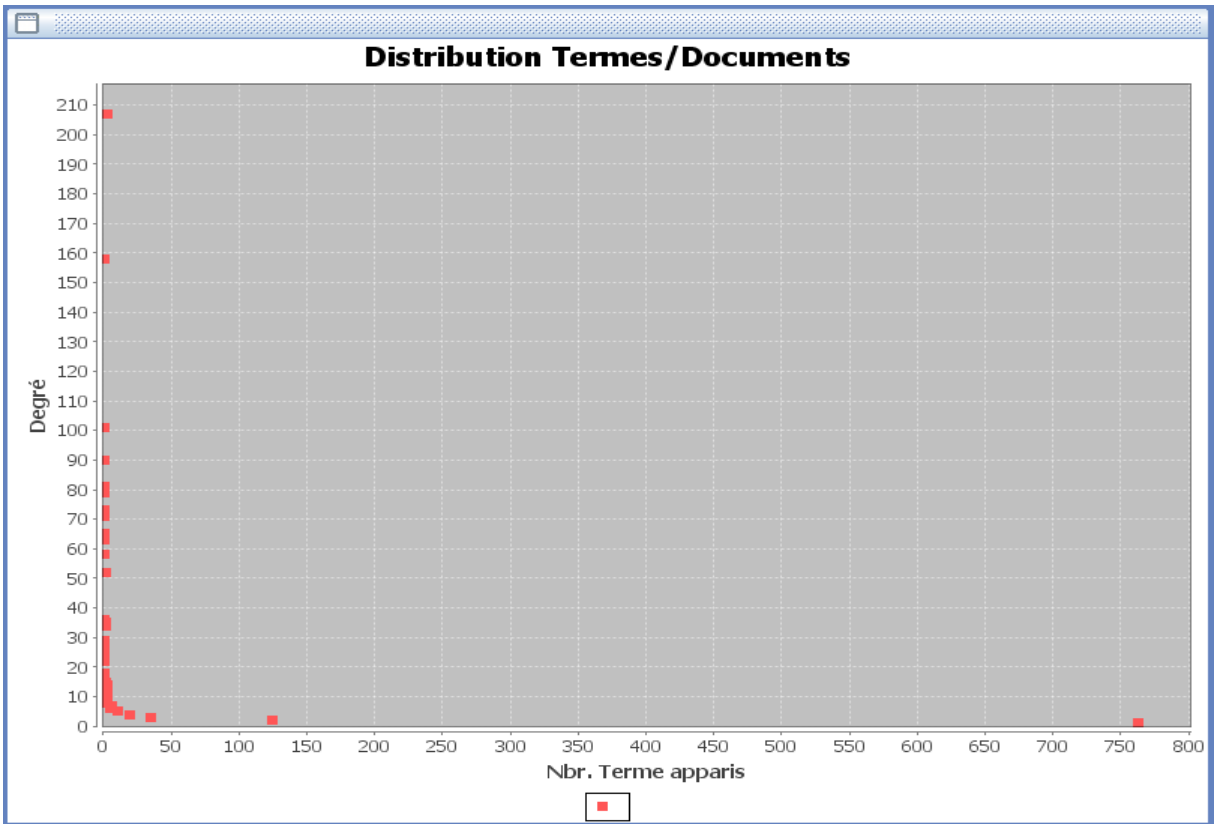


Figure 5. 18 Distribution Termes-Document

Le même constat est donné par les figures 5.25 et 5.26 qui donnent la répartition et la distribution des fréquences des termes sur les documents de l'expérimentation.

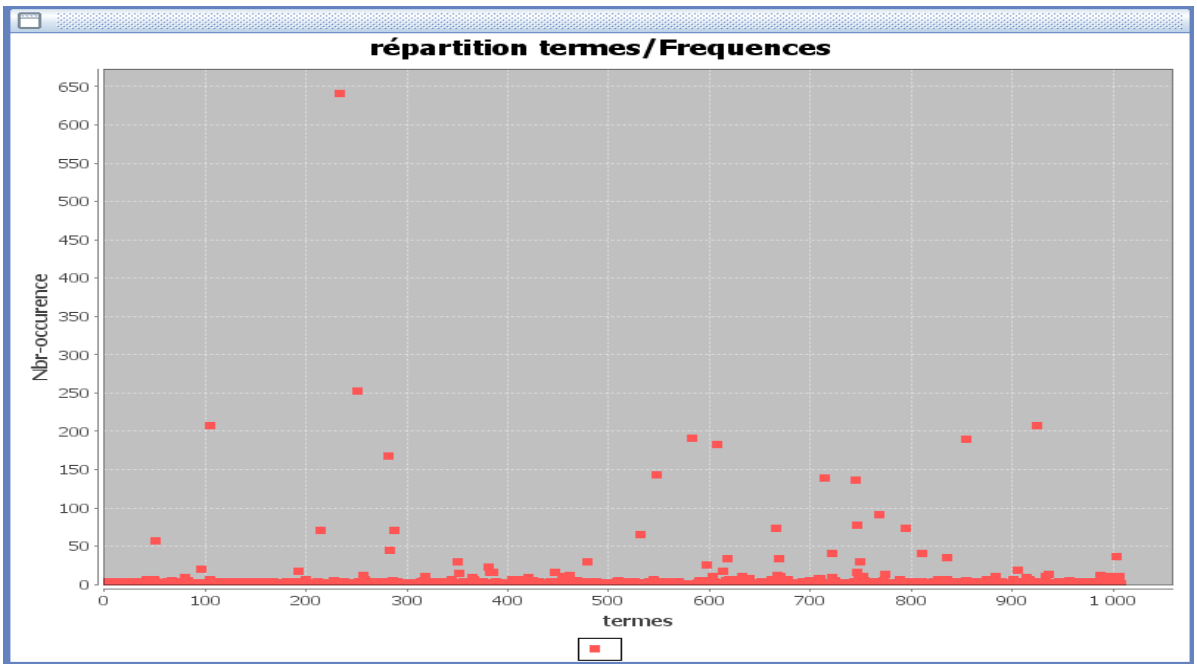


Figure 5. 19 Répartition Termes-Fréquences

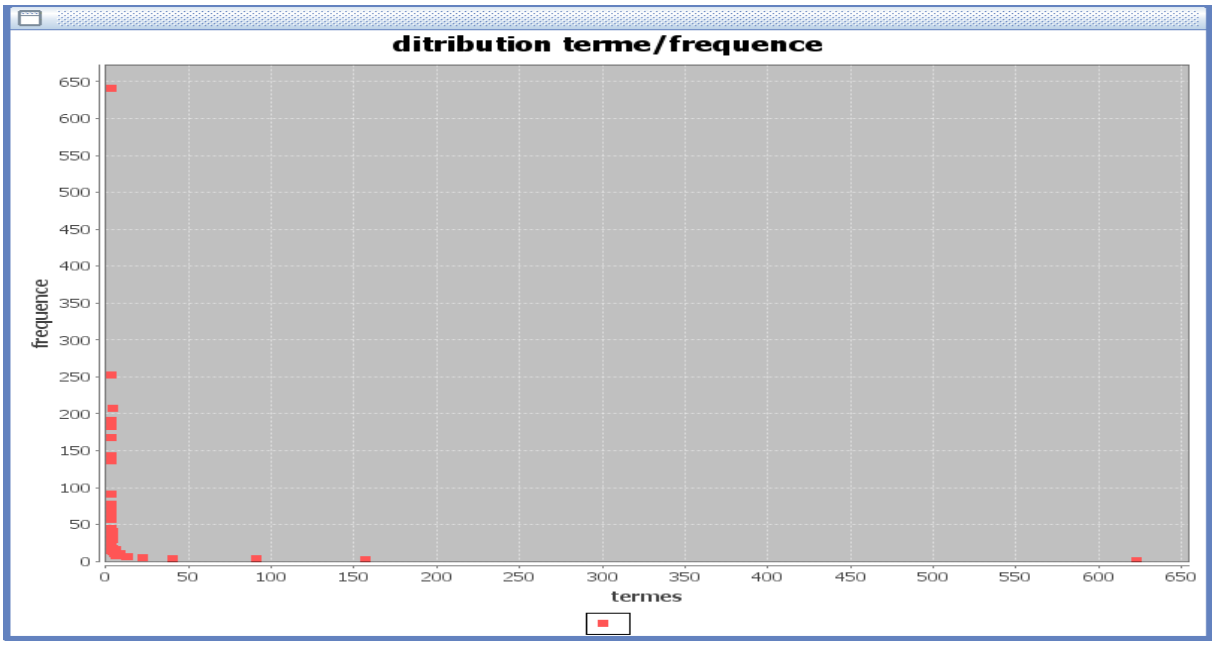


Figure 5.20 Distribution Termes-Fréquences

la figure 5.29 qui montre que pour un TTL=3 il est suffisant d'avoir un maximum de bonnes réponses et que pour deux pas du TTL, notre approche donne plus de 80% des résultats.

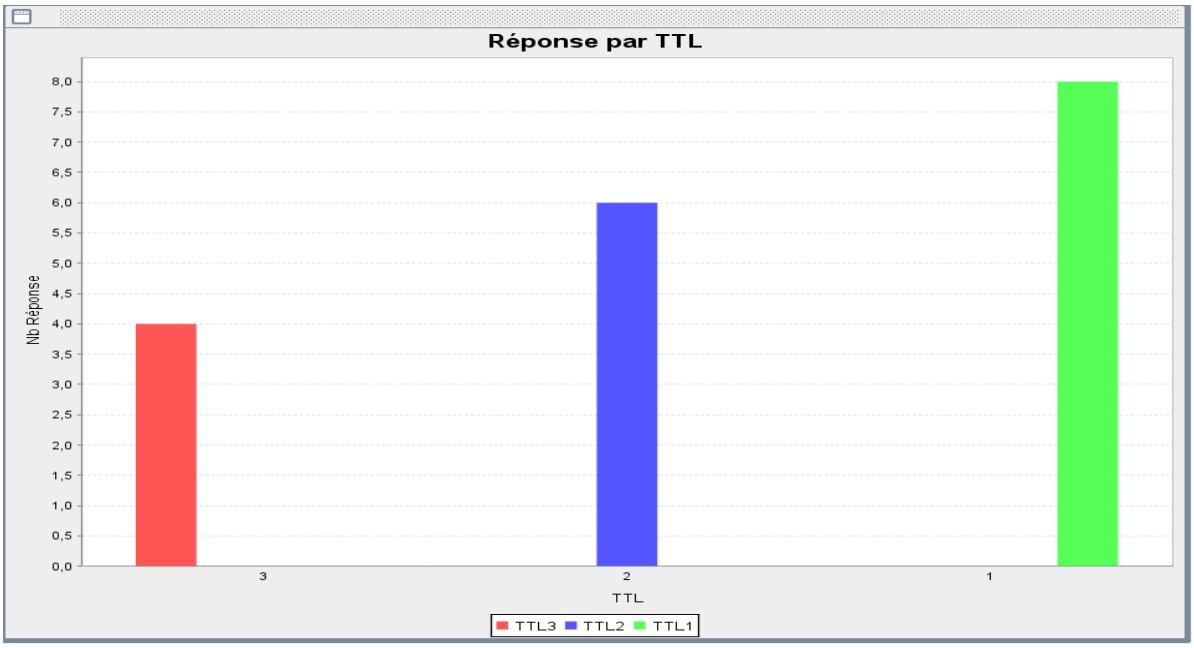


Figure 5.21 TTL (time to live)

7. Conclusion

Dans ce chapitre, nous avons présenté la description de l'application. En présentant les étapes nécessaires pour la création de notre projet avec les différents modes d'exécution. Notre application permet la création du réseau et étudié les propriétés de ce même réseau.

Et perspectives :

La localisation de données dans les systèmes pair-à-pair hybrides est un défi vu la quantité de fichiers et le nombre large des pairs. Il faudra trouver un compromis entre la recherche aveugle qui découvre les profils des pairs et la recherche sémantique qui touche directement aux pairs détenteurs des réponses.

Dans ce mémoire, nous avons présenté une méthode de recherche sémantique qui a pour but d'améliorer les profils sémantiques des pairs et de localiser les données. Les gains en temps de calcul et en espace sont bien prouvés. Nous avons réalisé un système pair-à-pair hybride sur lequel nous avons réparti des fichiers réels. Les résultats de l'expérimentation ont montré qu'il est possible de maximiser les recherches ont ne faisons que deux pas du TTL chose qui n'a pas été obtenue auparavant.

Plusieurs perspectives se présente pour ce travail, car concevoir une plate forme de simulation nécessite plusieurs étapes, notre travail représentait que la partie construction du réseau, donc parmi ses perspective on peut citer :

- Prendre l'aspect dynamique du réseau en compte (connexion et déconnexion des pairs).
- Echanges de requête entre les pairs.
- Ajout des Index de recherche pour chaque pair.
- Implémenter une couche de réseau logique.

Résumé

L'idée des systèmes pair-à-pair est d'offrir de nouvelles opportunités pour concevoir des réseaux purement distribués. L'arrivée de la notion du Web Sémantique a donné lieu à de nouvelles catégories des systèmes p2p. Chaque pair peut utiliser, contrôler, gérer ces données, ainsi il peut maintenir sa propre autonomie,

Nous considérons ces réseaux, dont le but général est de faciliter le partage des données, bases de connaissances, flux d'informations, …etc. Les systèmes p2p impliquent l'efficacité de l'échange d'information avec une consommation modérée de la bande passante.

Notre méthode semble la meilleure avec un bon compromis entre la complexité d'espace (mémoire, messages de communication, objets manipulés et nombre des opérations effectuées) et la complexité de temps de recherche pour l'évaluation des requêtes dans un système hybride.

Mots Clés : *P2P, systèmes hybride.*

Abstract

The idea of Peer-to-Peer (P2P) computing, offers new opportunities for building highly distributed data systems; the advent of Semantic Web gives rise to new categories of p2p systems. Each peer can use, control, manage its data, so it can maintain its own autonomy,

We consider these networks, which the main goal is to provide the easiest data sharing, knowledge bases, information flow, etc. The p2p network implies the efficiency of the exchange of information without overly consuming bandwidth.

The introduction of the concept of the Semantic Web has given rise to new categories and strategies in p2p systems. Each peer can use, control, manage data, so it can maintain its own autonomy,

Our method looks best with a good compromise between the complexity of space (memory, communication messages, manipulated objects and number of transactions) and the time complexity of research for the evaluation of queries in a hybrid system.

Bibliographie

[1] T. W. Malone, K. R. Grant, F. A. Turbak, S. A. Brobst et M. D. Cohen : Intelligent information-sharing Systems. Commun ACM, 30(5):390–402, 1987.

[3]Gharzouli Mohamed, Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer, le 25 Septembre 2011.

[4] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4):331–370.

[5] B. Durrenberger : « De la recherche de données dans des systèmes pair-à-pair » ; Looking up data in P2P Systems, Université de la Réunion, DESS RMI 2004/05.

[6] B. Zhao, J. Kubiatowicz, A. Joseph : « Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing » ; Computer, 2001.

[7] Castagnos S., Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information, Thèse de doctorat, LORIA - Université Nancy 2, 2008.

[8] Derbali Mohammed, Embouazza Fethi : « Etude des réseaux Peer to Peer » ; Université Paris 13 Nord, 2002.

[9] Ricci F., Rokach L., Shapira B., Kantor P. B., Recommender Systems Handbook, Springer, 2011.

[10]http://www.google.fr//P2P\p2p - Documentation Ubuntu Francophone.html**.**Denière date de consultation 18 Novembre 2014**.**

[11] R. Burke: Hybrid web recommender systems. In the adaptive web, pages 377–408. Springer, 2007.

[12] Baeza-Yates, R. Ribeiro-Neto, B. Modern Information Retrieval. Addison-Wesley.1999

[13] Belloui, A. (2008). L'usage des concepts du web sémantique dans le filtrage d'information collaboratif. Master's thesis, Institut National d'Informatique d'Alger.

[15] Jian Liang, Rakesh Kumar, Keith W. Ross : « Understanding KaZaA » ; 2004. Available at http://cis.poly.edu/ross/papers/UnderstandingKaZaA.pdf.

[17] "kazaa" : http://interstices.i_nfo/jcms/c_8622/les-reseaux-de-pair-a-pair?part=1

[20] "napster" http://www.napster.com .

[30] Stoica I., Morris R., Karger D., Kaashoek M. F., Balakrishnan H : « Chord: A scalable peer-to-peer lookup service for internet applications » ; proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, ACM Press, 2001.

[36] Hakik abdelaziz, Gharnati Fathallahe, Traitement de la réplication dans un système réparti. Gestion de la tolérance aux défaillances, PFE 2005, Institut d'informatique, université d'oran, es-sénia

[37] Kacimi Karima, Kadri walid, Recherche sémantique basée sur les mots clés, Dans les réseaux Peer-To-Peer, PFE 2009, département d'informatique.

[40] Zekri Lougmiri, Beldjilali Bouziane, Compression des index dans les systèmes p2p, proceeding of 1st International Conférence on Information Systems and Technologies.

[42] Kenniche Ahlem, « Indexation de Données à Large Echelle », le diplôme de magister Spécialité : Informatique Option : Informatique et Automatique Intitulé, Université d'Oran Es- Senia.Juin 2010

[47] J.Lu. J. Callan. Content-Based Retrieval in Hybrid Peer-to-Peer Networks. Proceedings of ACM CIKM'03. New Orleans. LA. pp. 199-206.2003.

[48] K.Aberer. P-Grid. A self-organizing access structure for P2P information systems. In Proc. of the 6th International Conference on Cooperative Information Systems. 2001

[49] F.Dabek. M.Kaashoek. D.Karger. R. Morris. I. Stoica. Wide-area Cooperative Storage with CFS. In Proc. of the18th ACM Symposium on Operating Systems Principles, 2001.

[50] S.Ratnasamy. P. Francis. M. Handley. R. Karp. S. Shenker. A Scalable Content-Addressable Network. In Proc. of the ACM SIGCOMM'01 Conference. 2001

[51] C.Tang. Z. Xu. M. Mahalingam. Efficient Information Retrieval in Peer-to-Peer Networks. In Proc. of HotNets-I,ACM SIGCOM., 2002.

[52] P.Haase. R.Siebes. F.v.Harmelen. Peer Selection in Peer-to-Peer Networks with Semantic Topologies, In:Proc. of the Semantics of a Networked World, Semantics for Grid Databases, First intl. IFIP conf., ICSNW, Paris, France, 17–19, pp 108–125. June 2004.

[53] W.Nejdl. B .Wolf. S. Staab. Edutella: Searching and Annotating Resources within an RDF-Based P2P Network", In: Proc. of semantic web workshop 2002.

[54] C. Raja. D.Bruno. H.Georges. Définition et Diffusion de Signatures Sémantiques dans les Systèmes Pair-à-Pair. 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.

[55] B.Defude .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.

[56] Touati Ilies, Taibi Djamel : « Etude et Analyse Spectraledes Grands Graphes et Exploitation des Ultra-Pairs pour la Recherche de données »; Faculté des sciences, département d'informatique. Université d'Oran Es-sénia, 2008.