Proposition d'approches de routage de requêtes dans les systèmes pair-à-pair non structurés

Taoufik Yeferny

École Doctorale de Mathématiques, Informatique, Sciences et Technologie des Matériaux

École Doctorale Sciences et Ingéniérie

THÈSE

présentée en vue de l'obtention du
Diplôme de Docteur en Informatique
par

Taoufik YEFERNY

Proposition d'approches de routage de requêtes dans les systèmes Pair-à-Pair non structurés

soutenue le 15 janvier 2014 devant le jury d'examen

MM.	**Thierry DELOT**	**Pr. Université de Valenciennes**	**Président**
	Philippe LAMARRE	**Pr. INSA Lyon**	**Rapporteur**
	Nadia ESSOUSSI	**M.C. FSEG, Nabeul**	**Rapporteur**
	Sadok BEN YAHIA	**Pr. FST, Tunis**	**Examinateur**
	Khedija AROUR	**M.A. INSAT, Tunis**	**Invitée**
	Yahya SLIMANI	**Pr. ISAMM, Manouba**	**Directeur de thèse**
	Amel BOUZEGHOUB	**Pr. Télécom SudParis, Evry**	**Directeur de thèse**

Thèse n° 2014$TELE$0002

Thèse préparée sous convention de cotutelle au sein du Laboratoire LISI (INSAT, Tunisie) et du Laboratoire SAMOVAR (Télécom SudParis, France)

Dédicace

Cette thèse représente l'aboutissement du soutien et des encouragements que mes parents m'ont prodigués tout au long de ma scolarité.

La patience et l'encouragement de ma femme m'ont aidé à surmonter toutes les difficultés rencontrées au cours de cette thèse.

A mon petit Dhia Eddine.

Remerciements

Je tiens à exprimer tout d'abord mes remerciements aux membres du jury, qui ont accepté d'évaluer mon travail de thèse : Monsieur Thierry DELOT pour tous ses efforts et sa contribution au bon déroulement des procédures administratives aboutissant à la soutenance de ma thèse qui de plus m'a fait l'honneur de présider le jury de cette thèse. Je tiens également à remercier Monsieur Philippe LAMARRE et Madame Nadia ESSOUSSI pour avoir accepté d'être rapporteurs de ce travail, Monsieur Sadok BEN YAHIA de m'avoir fait l'honneur de participer à ce jury de thèse et d'examiner mes travaux.

Je tiens à exprimer mes remerciements les plus sincères à Monsieur Yahya SLIMANI et Madame Amel BOUZEGHOUB, mes directeurs de recherche, pour m'avoir conseillé, encouragé et soutenu tout au long de la thèse avec patience et disponibilité, et pour la confiance qu'ils m'ont accordée. Je les remercie également de m'avoir donné l'opportunité d'assister à des congrès internationaux et d'exercer une activité de recherche en France.

J'adresse mes remerciements les plus chaleureux à mon co-directeur de thèse Madame Khedija AROUR pour tous les conseils techniques et scientifiques qu'elle m'a apportés et qui m'ont beaucoup aidé à accomplir mes travaux.

Merci aussi à tous mes collègues et amis des laboratoires LISI et SAMOVAR. Je leur exprime ma profonde sympathie et leur souhaite beaucoup de bien. Enfin, une pensée émue pour tous les étudiants avec qui j'ai partagé une salle, un café, un repas ou une console d'ordinateur pendant mes années de recherche à la FST et à Télécom SudParis.

Cette thèse m'a donné l'occasion de rencontrer et de travailler avec des personnes absolument épatantes. Il est difficile de leur dire ici à quel point j'ai été touché par tout ce qu'ils ont fait pour moi.

Résumé : Ces deux dernières décennies les systèmes P2P de partage de fichiers sont devenus très populaires grâce aux accès à des ressources diverses, distribuées sur Internet. Parallèlement à l'évolution de cette catégorie de systèmes, les dispositifs mobiles (téléphones cellulaires, PDA et autres appareils portatifs) ont eu un grand succès sur le marché. Équipés d'une technologie de communication sans fil (Bluetooth, et Wifi), ils peuvent communiquer sans nécessiter une infrastructure particulière en utilisant un réseau mobile adhoc (Mobile Adhoc NETwork -MANET). De la même manière, les systèmes P2P peuvent être aussi déployés sur ce type de réseau et deviennent des systèmes P2P mobiles (Mobile 2P systems). Dans le cadre de cette thèse, nous nous intéressons essentiellement à la recherche d'information dans les systèmes P2P et plus précisément au problème de routage de requêtes. La première partie de la thèse, s'est focalisée sur le routage de requêtes dans les systèmes P2P sur Internet. Nous avons proposé *(i)* un modèle de routage sémantique basé sur l'historique des requêtes. Ce modèle est ensuite instancié pour définir une nouvelle méthode de routage par apprentissage. Pour pallier le problème de démarrage à froid, *(ii)* nous avons proposé une méthode prédictive de l'intention de l'utilisateur qui construit une base de connaissances à priori pour chaque pair. Enfin, *(iii)* nous avons proposé une méthode de routage hybride pour traiter le problème d'échec de sélection. Cette méthode est basée sur l'historique des requêtes et le regroupement de pairs dans des groupes sémantiques. La deuxième partie de la thèse, s'est focalisée sur le routage de requêtes dans les systèmes P2P mobiles. L'apparition des MANETs, a soulevé de nouveaux challenges de routage. Ces réseaux souffrent de plusieurs contraintes liées aux supports de transmission ou bien aux dispositifs mobiles. Dans ce cadre, nous avons proposé une méthode de routage pour les systèmes P2P non structurés mobiles basée sur le contexte de l'utilisateur. D'un point de vue technique, toutes ces propositions ont été développées, validées et évaluées grâce aux simulateurs PeerSim et NS2.

Mots clés : P2P, routage de requêtes, contexte utilisateur, réseau MANET

Query Routing On Unstructured P2P Systems

Abstract : Peer-to-peer systems have emerged as platforms for users to search and share information over the Internet. In fact, thanks to these systems, user can share various resources, send queries to search and locate resources shared by other users. Nowadays, mobile and wireless technology has achieved great progress. These devices are also equipped with low radio range technology, like Bluetooth and Wi-Fi, etc. By means of the low radio range technology, they can communicate with each other without using communication infrastructure (e.g. Internet network) and form a mobile ad hoc network (MANET). Hence, P2P file sharing systems can be also deployed over MANET. A challenging problem in unstructured P2P systems is query routing. Researches' efficiency and effectiveness can be improved by making smart decisions for query routing. Our contributions, in this thesis, focus on two complementary axes. Firstly, our research work focalized on P2P systems over Internet. We introduced a novel semantic model for query routing based on past queries, thereafter we instantiated this model to define our specific routing method. In addition, we addressed two difficult challenging problems : (i) the bootstraping (ii) the unsuccessful relevant peers search. Secondly, we are focalized on P2P systems over MANET. Due the nature of MANET, mobile P2P systems suffer from several constraints of wireless medium and energy-limited. Indeed, query routing methods proposed for P2P system over Internet cannot be applied. In this context, we proposed a context-aware integrated routing method for P2P file sharing systems over MANET. The different contributions are developed, validated and evaluated with the network simulators PeerSim and NS2.

Keywords :

P2P, query routing, user's context, MANET network

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les deux dernières décennies ont vu naître et évoluer les réseaux informatiques, ce qui a permi aux utilisateurs de partager et rechercher différents types d'informations (i.e., des vidéos, musiques, des documents textuels, des images, etc.) via des systèmes de recherche d'information. Au début, ces systèmes ont été développés selon une architecture Client/Serveur. Un tel système est constitué d'un ou plusieurs serveurs qui sont chargés de l'indexation des documents partagés et de traiter les requêtes (i.e., liste de mots clés) de leurs clients. Ces dernières années, les systèmes basés sur cette architecture ont montré leurs limites. Le nombre de documents partagés augmente d'une manière exponentielle, ainsi, les serveurs doivent gérer des indexes de taille assez élevée. De plus, le nombre d'utilisateurs ne cesse d'augmenter ce qui entraîne une charge de traitement de requêtes importante au niveau des serveurs. Pour aborder ces problèmes, une première solution consiste à utiliser un grand nombre de serveurs puissants ayant de bonnes capacités de traitement et de stockage. Bien que cette solution soit faisable, elle est très coûteuse dans sa mise en place et donc restreinte à de grandes entreprises. Le système de recherche d'information le plus utilisé Google, par exemple, a adopté cette solution. En 2011, Google possédait un parc de plus de 900000 serveurs [Wikipedia 2011].

Une alternative à cette architecture centralisée consiste à développer des systèmes de recherche d'information selon une architecture décentralisée où l'indexation des ressources et le traitement des requêtes seront sous la responsabilité de toutes les machines du réseau. Ce mode de fonctionnement est appelé Pair-à-Pair (Peer to Peer ou P2P). Dans le modèle P2P, tous les participants sont des pairs qui ont un rôle équivalent, à la fois client et serveur [Bender 2007]. Cette solution permet d'indexer un volume massif de ressources partagées et supporte le passage à l'échelle avec un faible coût. Dans ce contexte, plusieurs systèmes basés sur ce mode de fonctionnement ont vu le jour et sont connus sous le nom de *systèmes de partage de fichiers en pair-à-pair sur Internet*, comme Gnutella [Gnutella 2009], KaZaA [Kazaa 2008], Napster [Napster 2008], etc.

De nos jours avec la veille technologique, les dispositifs mobiles (i.e., les téléphones cellulaires, les PDA et autres appareils portatifs) ont eu un grand succès sur le marché. En effet, ils ont une bonne capacité de stockage, une capacité de traitement plus élevée et ils offrent des fonctionnalités riches. Ils permettent à l'utilisateur de stocker des données de différente nature (audio, vidéo, texte et image). Ces mobiles sont équipés d'une technologie de communication sans fil comme Bluetooth et Wifi. De ce fait, ils peuvent communiquer sans besoin d'une infrastructure (i.e., le réseau Internet) en construisant un réseau mobile ad hoc (Mobile Adhoc NETwork : MANET). Par conséquent, les systèmes P2P peuvent être aussi déployés sur ce type de réseau. Ils sont appelés les systèmes P2P mobiles (Mobile P2P systems).

Contexte de travail

Dans cette thèse, nous nous focalisons sur les systèmes de partage de fichiers en P2P sur Internet et mobiles. Ces systèmes construisent une topologie de réseau logique (overlay) qui définit la manière avec laquelle les pairs sont connectés. En plus, un protocole de routage de requêtes est associé à cette topologie. Ce protocole définit le modèle d'échange de messages pour partager et localiser les pairs qui possèdent les ressources pertinentes à une requête donnée. La topologie du réseau et le protocole de routage associé ont une influence significative sur les propriétés de l'application P2P telles que les performances, la scalabilité (i.e., passage à l'échelle) et la fiabilité.

Nous pouvons classifier ces systèmes selon leurs topologies et les protocoles de routage associés en deux grandes catégories : Systèmes non-structurés et systèmes structurés. Dans les systèmes P2P non structurés, les pairs se connectent de manière adhoc et le placement de contenu est complètement indépendant de la topologie du réseau. La recherche dans ces systèmes se base sur la propagation aléatoire des requêtes dans le graphe de pairs (principe d'inondation). Une requête est résolue localement puis envoyée d'une manière récursive à un certain nombre de voisins choisis d'une manière aléatoire jusqu'à l'atteinte d'une limite de propagation (TTL, Time To Live). En effet, la localisation des ressources pertinentes aux requêtes des utilisateurs n'est pas effectivement contrôlée et aucune garantie pour le succès d'une recherche n'est offerte à l'utilisateur. Pour remédier à ce problème, quelques travaux de recherche ont introduit une topologie structurée du réseau. Dans ce réseau, une table de hachage distribuée (DHT-Distributed Hash Table) est utilisée pour placer les nouveaux pairs, en se basant sur leurs identifiants, dans le réseau. Lorsqu'une ressource est déposée sur ce réseau, une copie de celle-ci sera placée dans le pair ayant l'identifiant qui est le plus proche de celui de la ressource. Pour localiser une ressource, une technique de routage par contenu détermine la clé de hachage de celle-ci et l'identifiant du pair le plus proche.

Problématique et principales contributions

Les systèmes P2P proposent, selon leurs architectures, différentes techniques de localisation des données qui se traduisent par différentes méthodes de routage de requêtes. La méthode de routage la plus efficace est celle des systèmes structurés mais au prix de la perte d'autonomie. En effet, ces systèmes ne respectent pas l'autonomie des pairs, ni au niveau stockage, ni au niveau routage. A l'inverse, les systèmes P2P non structurés totalement décentralisés "purs" respectent bien l'autonomie des pairs au niveau du stockage et du routage. Cependant, la méthode de routage classique basée sur le principe d'inondation (propagation aléatoire des requêtes) est inefficace. Si on veut garder l'autonomie pour une bonne performance, il faut essayer de remplacer la propagation aléatoire par une propagation "guidée" (aller sur les "bons" pairs). En effet, le but de ce travail est d'introduire des nouvelles méthodes de routage de requêtes dans les systèmes P2P non structurés totalement décentralisés "purs" sur Internet ou mobiles.

Routage de requêtes dans les systèmes P2P non structurés purs sur Internet

Plusieurs travaux de recherche ont tenté d'améliorer les méthodes de routage classiques dans les systèmes P2P sur Internet, en introduisant de la sémantique dans le processus de routage des requêtes. Une première idée consiste à introduire des connaissances sur le contenu des pairs. Cela permet ainsi de choisir en fonction du contenu de la requête, les pairs les plus pertinents avant de la propager. Une autre idée consiste à construire des groupes (ou "clusters") les pairs partageant une "même connaissance". Le mécanisme de résolution d'une requête consiste alors à rechercher le ou les meilleurs clusters pour cette requête (c'est-à-dire les plus proches). Une autre idée possible consiste à utiliser l'historique des requêtes passées. Lors de la résolution d'une requête, on peut utiliser en priorité les pairs qui ont fourni précédemment des réponses à des requêtes émises. Ces dernières méthodes sont plus avantageuses que celles basées sur le contenu ou sur le regroupement de pairs car la construction des bases de connaissances ne nécessite pas un trafic réseau supplémentaire. Les méthodes basées sur l'historique de requêtes proposées dans la littérature améliorent l'efficacité et la performance de recherche des méthodes classiques. Cependant, elles souffrent de trois principaux inconvénients, à savoir :

1. Représentation de profil de l'utilisateur : Le profil de l'utilisateur est représenté par des statistiques sur les requêtes passées et les réponses retournées mais elles ne prennent pas en considération les relations entre les requêtes et les pairs qui agissent positivement à ces requêtes ;
2. Problème de démarrage à froid : Au démarrage du système, la base de connaissances est vide. Par conséquent, une technique de routage classique qui génère un nombre de messages très important et des réponses non satisfaisantes pour les utilisateurs est adoptée ;
3. Problème d'échec de sélection : La base de connaissances peut être non vide, mais l'algorithme de routage peut déterminer un nombre insuffisant de pairs pertinents. Dans ce cas, toutes les méthodes existantes ajoutent aux pairs sélectionnés un certain nombre de voisins choisis aléatoirement à partir de la table de routage. Par conséquent, le processus de routage est partiellement sémantique, ce qui dégrade les performances de la méthode de routage.

Dans un premier volet de notre thèse, nous proposons *(i)* un modèle de routage sémantique basé sur l'historique des requêtes. Contrairement aux méthodes de routage existantes qui représentent le profil utilisateur par des statistiques sur les requêtes passées et les réponses associées, dans notre modèle, le profil de l'utilisateur est représenté par un ensemble d'intérêts. Chaque intérêt représente des relations sémantiques entre les requêtes passées, leurs termes et les pairs positifs (i.e., les pairs qui ont agi positivement à des requêtes passées). Ensuite, ce modèle est utilisé pour définir notre propre méthode de routage par apprentissage (*Learning Routing Method* : *LRM*). De

plus, pour palier le problème de démarrage à froid, *(ii)* nous proposons une méthode prédictive de l'intention de l'utilisateur qui construit une base de connaissances initiale pour chaque pair. Cette base de connaissances initiale est construite au démarrage du système. Enfin, *(iii)* nous proposons une méthode de routage hybride (*Learning Routing Method based on Clustred Network* : *LRMCN*) pour traiter le problème d'échec de sélection. Cette méthode est basée sur l'historique des requêtes et le regroupement de pairs dans des groupes sémantiques.

Routage de requêtes dans les systèmes P2P non structurés mobiles

L'apparition des MANETs, a soulevé des nouveaux challenges de routage. Ces réseaux souffrent de plusieurs contraintes liées aux supports de transmission ou bien liées aux dispositifs mobiles. Les problèmes majeurs sont la mobilité des nœuds et la faible puissance d'énergie de batteries des nœuds mobiles. En effet, les pairs peuvent se déplacer librement. Par conséquent, le voisinage d'un nœud peut varier continuellement : à tout moment des stations peuvent être joignables directement ou non. La mobilité des nœuds engendre un changement fréquent de la topologie du réseau ce qui pose des problèmes de communication entre les nœuds. En outre, les batteries ont une durée de vie limitée. De ce fait, le temps d'utilisation d'une station est limité par la capacité de sa batterie et aussi par la puissance de son processeur.

Dans la littérature, une première approche consiste à adapter les méthodes de routage des requêtes de systèmes P2P déployés sur Internet pour les systèmes P2P mobiles. L'idée est de garder le même principe de la méthode de routage dans la couche application et d'utiliser un protocole de routage MANET existant de la couche réseau pour échanger les requêtes et les réponses. En effet, le protocole de routage dans la couche application sélectionne les bons voisins logiques vers lesquels la requête sera envoyée, puis il utilise un protocole de routage existant implanté dans la couche réseau pour localiser ces voisins. Les méthodes proposées pour les systèmes P2P sur Internet ne sont pas performantes due aux contraintes du réseau MANET. Elles ne prennent pas en considération des informations contextuelles, comme par exemple la charge de la batterie, la capacité de traitement du nœud, la charge du nœud mobile, la mobilité des nœuds, etc.

Une alternative à cette approche consiste à intégrer les fonctionnalités de la couche application avec celle de la couche réseau. L'idée consiste à envoyer d'une manière récursive la requête vers un certain nombre de voisins choisis en se basant sur le contexte de l'utilisateur, comme le contenu de la requête, la puissance de la batterie, la charge du calcul de dispositif mobile utilisé ainsi que la mobilité des utilisateurs, jusqu'à ce que des pairs pertinents soient trouvés. En effet, le choix des meilleurs voisins dépend de l'efficacité de la fonction de sélection et des informations contextuelles utilisées. Chacune des méthodes étudiées comprend des avantages et des inconvénients par rapport à l'autre. Afin de palier les inconvénients de ces méthodes, nous proposons une nouvelle méthode de routage pour les systèmes P2P non structurés mobiles basée sur le

contexte de l'utilisateur (*Context Aware Routing Method* : *CARM*) [Yeferny 2012b]. Les objectifs de notre méthode sont les suivants :

1. Sélectionner les meilleurs voisins pour une requête donnée, en se basant sur le contenu de la requête et les profils des voisins ;
2. Garantir que les pairs pertinents sont atteint en prenant en considération les contraintes de MANET. Nous avons considéré la puissance d'énergie de la batterie et la mobilité des nœuds pour garantir la stabilité de lien entre l'initiateur de la requête et le pair qui possède les documents pertinents. De plus, notre méthode traite le problème de congestion en évitant de router les requêtes vers les pairs chargés ou les lignes encombrées.

Organisation du mémoire

Cette thèse est composée de cinq chapitres, les deux premiers chapitres abordent le contexte dans lequel se situent nos travaux et les trois derniers présentent nos propositions et les expérimentations effectuées.

Dans le premier chapitre, nous présentons d'une manière générale les systèmes P2P et en particulier les systèmes de partage de fichiers en P2P et les différentes architectures sur lesquelles ils reposent. En second lieu, une revue critique de la littérature dans laquelle nous décrivons en détail les méthodes de routage de requêtes dans les systèmes P2P sur Internet est présentée.

Dans le deuxième chapitre, nous présentons en premier lieu les réseaux mobiles adhoc MANETs ainsi que les nouvelles contraintes de déploiement des systèmes P2P sur ces réseaux. En second lieu, nous décrivons les nouveaux défis de routage des requêtes dans les systèmes P2P non structurés mobiles. Ensuite, nous développons un état de l'art sur les méthodes de routage des requêtes dans ces systèmes.

Dans le troisième chapitre, nous détaillons les solutions proposées aux problèmes de représentation de profil d'utilisateur, de démarrage à froid et d'échec de sélection des pairs pertinents dans les méthodes de routage basées sur l'historique des requêtes pour les systèmes P2P sur Internet. Un modèle de routage sémantique basé sur l'historique des requêtes est présenté. Ensuite, ce modèle est utilisé pour définir notre propre méthode de routage par apprentissage, *LRM*. De plus, pour palier le problème de démarrage à froid, nous présentons une méthode de prédiction de l'intention de l'utilisateur qui construit une base de connaissances initiale pour chaque pair. Enfin, nous présentons une méthode de routage hybride *LRMCN* basée sur l'historique des requêtes et le regroupement de pairs pour traiter le problème d'échec de sélection.

Dans le quatrième chapitre, nous introduisons notre méthode de routage intégrée *CARM* pour les systèmes P2P non structurés mobiles. Nous décrivons les informa-

tions contextuelles considérées et les différentes métriques utilisées pour sélectionner les meilleurs voisins. Les algorithmes de routage et de sélection des meilleurs voisins sont détaillés.

Le cinquième chapitre est consacré aux expérimentations : nous décrivons et commentons les différentes évaluations utilisées pour analyser le comportement de nos solutions. Nous avons utilisé trois collections de tests pour comparer l'efficacité et la performance de nos méthodes de routages par rapport aux méthodes de la littérature.

Enfin, nous concluons et proposons différentes pistes de recherche.

Chapitre 1

Les systèmes pair à pair sur Internet et les méthodes de routage

1.1 Introduction

Les systèmes P2P présentent une nouvelle technologie d'accès aux diverses ressources, distribuées à large échelle. Les propriétés de ces systèmes leurs permettent de résoudre des problèmes d'utilisation optimale des ressources (i.e., cpu, capacité de stockage, etc.) disponibles dans les réseaux ainsi que le problème de tolérance aux fautes. Actuellement, ces systèmes sont utilisés dans plusieurs domaines comme les grilles de calcul, le partage de fichiers, les base de données distribuées et la communication (Voie sur IP, messagerie instantanée, etc.). Ces dernières années, de nombreux systèmes P2P de partage de fichiers sur Internet ont été développés, selon différents types d'architectures, et avec plus ou moins de succès pour l'usage à grand public tels que Gnutella [Gnutella 2009], KaZaA [Kazaa 2008], Napster [Napster 2008], etc. Un problème crucial dans ces systèmes est le routage de requêtes pour localiser les pairs qui possèdent des documents pertinents (i.e., fichiers) pour une requête donnée.

Dans ce chapitre, nous présentons en premier lieu d'une manière générale les systèmes P2P, leurs caractéristiques et quelques domaines d'application de ces systèmes. Ensuite, nous présentons en détail les systèmes de partage de fichier en P2P et les différentes architectures sur lesquelles ils reposent. Nous présentons ainsi une comparaison entre ces différentes architectures. En second lieu, nous décrivons en détail les méthodes de routage de requêtes dans les systèmes P2P déployés sur Internet. De plus, pour finir, nous présentons une synthèse des différentes méthodes existantes.

1.2 Les systèmes P2P

Conceptuellement, le modèle P2P est une alternative au modèle client/serveur, qui est constitué d'un serveur ou d'une grappe de serveurs et de clients. Dans le modèle P2P, tous les participants sont des pairs qui ont un rôle équivalent, à la fois client et serveur. Les systèmes qui adoptent ce mode de fonctionnement sont appelés : systèmes P2P. Ils apparaissent actuellement comme un moyen populaire pour partager et échanger des ressources (des fichiers le plus souvent, mais également des calculs, des données, etc.) dans un environnement distribué à large échelle.

1.2.1 Propriétés des systèmes P2P

Les systèmes P2P possèdent les caractéristiques suivantes :

- ***Partage de coût :*** Un système P2P partage les coûts de stockage et de traitement entre tous les pairs à l'opposé d'un système Client/Serveur qui répartit le coût sur un seul ou plusieurs serveurs [Bender 2007]. En effet, les systèmes P2P sont composés d'un ensemble de pairs qui acceptent volontairement de partager une partie de leurs ressources (fichiers, disque, cpu, etc). Chaque pair participe dans les traitements de requêtes ce qui permet de répartir le coût sur plusieurs pairs ;
- ***Grande échelle :*** L'absence d'une autorité centrale forte dans un réseau P2P pourrait lui permettre de se développer à grande échelle. Nous verrons plus loin que l'échelle d'un tel réseau P2P dépend du choix d'un modèle d'architecture et aussi d'un algorithme de routage appliqué sur ce réseau [Bender 2007] ;
- ***Autonomie :*** Cette caractéristique assume que le fonctionnement d'un pair ne repose sur aucun fournisseur de services centralisé. Tout travail d'un utilisateur doit pouvoir s'effectuer localement dans le pair ou avec un réseau partiel de pairs [Bender 2007] ;
- ***Dynamisme :*** Dans les systèmes P2P les utilisateurs participent volontairement en partageant une partie de leurs ressources. Ils peuvent rejoindre ou quitter le réseau à tout moment sans bloquer le fonctionnement du système [Bender 2007], ce qui crée un réseau fortement dynamique ;
- ***Hétérogénéité :*** Un système P2P peut être hétérogène en terme de capacité matérielle des pairs (i.e. Un pair peut être une petite machine et un autre peut être un super-ordinateur).

1.2.2 Domaines d'application

Les systèmes P2P ont émergé comme un moyen populaire et efficace pour partager un grand nombre de ressources, comme les cycles de calcul, des fichiers, des bases de données et des connaissances ou expertises. Ces systèmes sont utilisés dans plusieurs domaines. Nous citons par exemple, les grilles de calcul qui permettent de partager et d'agréger une variété de ressources de calcul distribuées géographiquement et de les présenter comme une seule ressource unifiée pour résoudre des problèmes de calcul et de stockage intensif. Les bases de données distribuées sont aussi basées sur le modèle P2P pour répartir le stockage de données et les traitements des requêtes sur plusieurs entités au lieu d'un seul serveur central. En outre, ces systèmes se sont imposés ces dernière années comme la technologie majeure de partage de fichiers sur Internet. Ils sont devenus très populaires car ils offrent la possibilité aux utilisateurs de partager et d'accéder à des ressources diverses, distribuées à large échelle.

1.3 Les systèmes de partage de fichiers en P2P

Dans cette thèse, nous nous focalisons sur les systèmes de partage de fichiers en P2P. Ces systèmes permettent à chaque utilisateur de partager des documents et de télécharger les documents partagés par les autres utilisateurs. Ils construisent une topologie de réseau logique qui définit la manière avec laquelle les pairs sont connectés. En plus, un protocole de routage de requêtes est associé à cette topologie. Ce protocole définit le modèle d'échange de messages pour partager et localiser les ressources. La topologie du réseau et le protocole de routage associé ont une influence significative sur les propriétés de l'application P2P telles que les performances, la scalabilité (i.e., passage à l'échelle) et la fiabilité.
Nous pouvons classifier ces systèmes selon leurs topologies et les protocoles de routage associés en deux grandes catégories : les systèmes non-structurés et les systèmes structurés.

1.3.1 Systèmes P2P non-structurés

La plupart des applications populaires de partage de fichiers en P2P constituent des réseaux logiques non structurés. Dans ces réseaux, les pairs se connectent de manière adhoc et le placement de contenu est complètement indépendant de la topologie du réseau. Chaque pair gère localement ses ressources partagées. Bien que les systèmes P2P non structurés soient supposés fonctionner de manière totalement décentralisée, dans la pratique, nous rencontrons divers degrés de centralisation dans les réseaux non structurés. En effet, selon le degré de centralisation trois catégories de réseaux P2P non structurés peuvent être identifiées : Architecture centralisée, totalement décentralisée et partiellement décentralisée.

Architecture centralisée

Dans cette architecture tous les pairs sont connectés à un serveur central qui se charge de l'indexation de tous les documents partagés sur le réseau. En général, la mise à jour de cet index s'effectue en temps réel, dès qu'un pair se connecte ou quitte le réseau. Dans les système P2P basés sur cette architecture, l'algorithme de recherche est très simple parce que toute recherche passe par ce serveur. En effet, lorsqu'un utilisateur lance une requête donnée, elle sera envoyée vers le serveur central. À son tour, le serveur cherche dans son index, les documents pertinents et les pairs associés puis il retourne la réponse à l'émetteur de la requête. Ce dernier peut établir des connexions directes avec les pairs qui possèdent les documents pertinents. L'avantage de l'indexation centralisée réside dans le confort et l'efficacité du processus de recherche, à condition que le serveur centralisé ne soit pas surchargé. Cependant, cette architecture introduit également deux inconvénients. D'une part, elle ne propose qu'une seule porte d'entrée, c'est le serveur central, ce qui peut bloquer le système en cas de panne de ce serveur. D'autre part, la taille de l'index est assez importante par conséquent, sa mise à jour est très coûteuse ce qui pose un problème de passage à l'échelle. Le représentant le plus connu de ce mode de P2P est Napster [Napster 2008].

Architecture totalement décentralisée

Les réseaux totalement décentralisés sont appelés "des réseaux P2P purs". Dans ce type d'architecture, il y a une symétrie complète au niveau de rôles de tous les pairs. Il n'existe aucune coordination centrale, tous les pairs sont égaux et jouent le même rôle (Client et Serveur à la fois). L'absence d'un serveur central permet aux systèmes totalement décentralisés d'être tolérant aux fautes. Cependant, dans ces systèmes chaque pair gère localement sa collection de documents partagés et il n'a pas des connaissances sur le contenu des autres pairs. Il ne connaît que les identifiants de ses voisins. Contrairement aux systèmes partiellement centralisés ou centralisés, où il suffit de se connecter au serveur pour avoir accès aux informations, la recherche dans ces systèmes se base sur la propagation aléatoire des requêtes dans le graphe de pairs (principe d'inondation). Une fois un pair possédant la ressource est trouvé, une connexion directe s'établit entre les deux pairs. Cette architecture, respecte au mieux l'autonomie des pairs et supporte des langages de requêtes plus expressifs. Cependant, la méthode de recherche classique conduit à générer un grand nombre de messages. Gnutella [Gnutella 2009] est le système le plus représentatif de ce type d'architecture.

Architecture partiellement décentralisée

Dans ce type de réseaux, il y a deux types de pairs : des super-pairs et des pairs ordinaires connectés à un super-pair. Le super-pair agit comme un répertoire centralisé pour le compte d'un ensemble fini de pairs. En effet, chaque pair ordinaire indexe sa collection locale puis il envoie son index local à son super-pair. En recevant les index locaux de ses pairs ordinaires, le super-pair construit un index global qui décrit les documents partagés par tous les pairs ordinaires qui sont connectés à lui. L'émetteur d'une requête donnée commence d'abord par interroger son super-pair, à son tour celui-ci cherche les documents pertinents à la requête et les pairs associés puis il la propage dans le graphe des super-pairs. Par conséquent, dans ces réseaux, le routage se limitera aux super-pairs. Cette topologie permet de diminuer le nombre de connexions sur chaque serveur, de diminuer le nombre des pairs à consulter dans le processus de recherche et de minimiser l'utilisation des bandes passantes. Ainsi, elle permet de résoudre le problème de l'approche totalement distribuée tout en gardant l'efficacité de la solution centralisée. Cependant, de tels systèmes supposent l'existence de pairs pouvant jouer le rôle de super pairs. Bien évidemment, de tels pairs existent mais les techniques de mises en œuvre sont complexes. Élire un pair comme super pair nécessite de disposer de nombreux paramètres sur ses capacités propres mais aussi et surtout sur les capacités, notamment en terme de bande passante et des réseaux l'entourant. Aussi, les systèmes basés sur cette architecture sont caractérisés par un langage de requêtes peu expressif vu que les super-pair décrivent d'une manière peu détaillée les collections des pairs ordinaires pour éviter d'avoir un index de taille importante, ce qui rend sa mise à jour très difficile. Plusieurs systèmes sont basés sur cette architecture le plus connu est le système KaZaA [Kazaa 2008].

1.3.2 Systèmes P2P structurés

Dans les systèmes P2P non structurés, la localisation des ressources pertinentes aux requêtes des utilisateurs n'est pas effectivement contrôlée et aucune garantie pour le succès d'une recherche n'est offerte à l'utilisateur. Pour remédier à ce problème, quelques travaux de recherche ont introduit une topologie structurée du réseau. Dans ce réseau, un identifiant est assigné à chaque pair. Par la suite, une table de hachage distribuée (DHT -Distrubuted Hash Table) est utilisée pour placer les nouveaux pairs, en se basant sur leurs identifiants, dans le réseau. De plus, lorsqu'une ressource est déposée sur ce réseau, un identifiant est assigné en appliquant une fonction de hachage sur son contenu. Une copie de la ressource sera alors gardée dans le pair ayant l'identifiant qui est le plus proche de celui de la ressource. Pour localiser une ressource, une technique de routage par contenu détermine la clé de hachage de celle-ci et l'identifiant du pair le plus proche.

Cette méthode de recherche est efficace puisqu'elle garantie la localisation des données qui sont disponibles dans le réseau en un temps logarithmique. En contrepartie, les problèmes majeurs sont la maintenance de la DHT et l'autonomie des pairs. Les systèmes les plus connus sont Chord [Chord 2008] et Pastry [Pastry 2008].

1.3.3 Comparaison des systèmes de partage de fichiers en P2P

Le tableau 1.1 représente une comparaison entre les architectures des systèmes P2P selon différents critères que nous avons définis, à savoir :

- *Autonomie :* nous voulons savoir si les pairs sont autonomes ou non ;
- *Qualité des réponses :* la qualité des réponses retournées ;
- *Expressivité du langage :* c'est à dire, voir si le langage de requêtes est expressif ou peu expressif ?
- *Tolérance aux pannes :* si un système est capable de fonctionner même s'il y a une panne au niveau d'un ou de plusieurs pairs on dit que l'architecture de ce système est tolérante aux pannes. Au contraire, si la panne d'un pair entraîne le dysfonctionnement du système on dit que l'architecture n'est pas tolérante aux pannes.

		Autonomie	Qualité des réponses	Expressivité du langage	Tolérance aux pannes
Système P2P non structuré	centralisé	Faible	Forte	Forte	Faible
	partiellement décentralisé	Forte	Forte	Forte	Forte
	totalement décentralisé	Forte	Faible	Forte	Forte
Système P2P structuré		Faible	Forte	Faible	Faible

TABLE 1.1 – Tableau comparatif des de systèmes P2P de partage de fichiers

Les systèmes P2P proposent selon leurs architectures différentes techniques de localisation des données, qui se traduisent par différentes méthodes de routage de requêtes. La méthode de routage la plus efficace est celle des systèmes structurés mais au prix de la perte d'autonomie. En effet, ces systèmes ne respectent pas l'autonomie des pairs, ni au niveau stockage, ni au niveau routage. Au contraire, les systèmes P2P non structurés totalement décentralisés "purs" respectent bien l'autonomie des pairs au niveau du stockage et du routage. Cependant, la méthode de routage classique basée sur le principe d'inondation (propagation aléatoire des requêtes) est inefficace. Si l'on veut garder l'autonomie pour une bonne performance, il faut essayer de remplacer la propagation aléatoire par une propagation "guidée" (aller sur les "bons" pairs). Dans la section suivante, nous détaillons les différentes méthodes de routage dans systèmes P2P non structurés proposées dans la littérature.

1.4 Routage de requêtes dans les systèmes P2P non structurés

Dans la littérature les premières méthodes de routage de requêtes dans les systèmes P2P non structurés purs sont appelées des méthodes aveugles ou classiques. Ces méthodes sont basées sur l'inondation des requêtes, une requête est résolue localement puis envoyée récursivement à un certain nombre de voisins choisis d'une manière aléatoire jusqu'à l'atteinte d'une limite de propagation (TTL, Time To Live). Plusieurs travaux de recherche ont tenté d'améliorer les méthodes classiques en introduisant de la sémantique dans le processus de routage des requêtes. Une première idée consiste à introduire des connaissances sur le contenu des pairs. Cela permet ainsi de "choisir" en fonction du contenu de la requête les pairs les plus pertinents avant de la propager. Une autre idée consiste à construire des groupes (ou " clusters ") regroupant les pairs partageant une "même connaissance". Le mécanisme de résolution d'une requête consiste alors à rechercher le ou les meilleurs clusters pour cette requête (c'est-à-dire ceux qui en sont le plus proches). Une autre idée possible consiste à utiliser l'historique des requêtes passées. Lors de la résolution d'une requête, on peut utiliser en priorité les pairs qui ont fourni précédemment des réponses à des requêtes émises.

1.4.1 Méthodes de routage classiques

Breadth-First Search : BFS

BFS est une technique de routage utilisée par le système Gnutella [Gnutella 2009]. Le principe de cette méthode est très simple, il consiste à propager récursivement la requête à tous les voisins. En effet, un pair qui lance ou reçoit une requête commence tout d'abord par faire une recherche dans sa collection locale, puis il la propage à tous ses voisins. Pour limiter la génération d'un nombre exponentiel de messages par ce procédé, la propagation des messages est limitée jusqu'à un horizon, défini par un paramètre de durée de vie de la requête (TTL : Time To Live), décrémenté à chaque pair. Une fois un pair possédant la ressource est trouvé, une connexion directe s'établit

entre les deux pairs. Cette approche est bien sûr très inefficace et conduit à la génération d'un grand nombre de messages.

Random Walk

Contrairement à BFS qui propage la requête à tous les voisins, dans la méthode marche aléatoire "*Random Walk*" [Lv 2002], quand un pair reçoit une requête, il choisit au hasard un seul voisin. Ce processus est répété jusqu'à ce que des documents pertinents soient trouvés. L'avantage de cet algorithme est qu'il réduit de manière significative le nombre de messages échangés. Cependant, il nécessite un délai important pour trouver les documents pertinents pour la requête. Pour pallier ce problème, les auteurs de [Lv 2002] ont suggéré que l'émetteur de requête l'envoie à k voisins au lieu d'un seul, chacun est donc un "*Walker*". Ensuite, quand un nœud voisin reçoit la requête, il utilise l'algorithme "*Random Walker*" afin de choisir aléatoirement un seul voisin et propage la requête vers celui-ci. Pour distinguer l'algorithme de base "*Random Walker*" de cette variante, le premier est appelé "*1-walker random walk algorithm*" tandis que le deuxième est appelé "*k-walker random walk algorithm*". Il y a deux mécanismes qui sont appliqués pour mettre fin à la propagation de la requête : Le (TTL) et le "*Checking*". Avec le "*Checking*", chaque Walker doit se référer régulièrement à l'émetteur et vérifie si le processus de propagation devrait être stoppé avant l'envoi de la requête au nœud suivant. De cette façon, la durée d'attente des réponses est diminuée. Les auteurs [Kalogeraki Vana 2002] ont proposé une autre méthode de routage appelée, "*random breadth first search (RBFS)*" qui est très similaire à "*K-walker*". Dans cette méthode, l'initiateur de la requête sélectionne aléatoirement k voisins et la propage vers ceux-ci. De même, chaque voisin répète le même processus jusqu'à ce que la condition de terminaison soit satisfaite, par exemple, profondeur maximale, TTL, est atteinte ou le résultat est trouvé.

L'approfondissement itératif (Iterative Deepening)

L'approfondissement itératif est une technique de recherche largement utilisée dans nombreux domaines plus ou moins liés à l'intelligence artificielle. Yang et al. [Yang 2002] ont appliqué cette technique pour le routage des requêtes dans les systèmes P2P. Dans cette approche, la requête est routée avec une séquence de plusieurs BFS en augmentant progressivement la profondeur de la recherche. Le processus de recherche se termine lorsque soit la profondeur maximale est atteinte ou que le résultat de recherche satisfait l'exigence de l'utilisateur. Pour ce faire, le système fournit une stratégie P pour spécifier une séquence de profondeur à chaque itération.
Formellement, la stratégie peut être modélisée comme suit : soit $P = (D1, D2, ..., Dn)$ une séquence de profondeurs, où D_i dénote la profondeur de l'$i^{ème}$ itération avec $D1 < D2 < ... < Dn$. En outre, une période de temps pour chaque itération, W, devrait également être spécifiée. Etant donnée, une stratégie de trois itérations, $P = (2, 4, 7)$ et une période de temps $W = 5$ secondes, le nœud source p_i envoi d'abord la requête en utilisant la méthode BFS avec une profondeur égale à 2. En effet, seuls les nœuds

de profondeur égale à deux (i.e., $P = 2$) reçoivent la requête. Cette dernière, est évaluée localement par ces nœuds, et les résultats de recherche sont retournés à p_i. Après W secondes, p_i met fin à la requête si les résultats satisferont l'utilisateur. Sinon, p_i envoi une autre fois la requête avec une profondeur égale à 4. Pour cette itération, les premiers nœuds de profondeur égale à deux seront de simples relais pour atteindre les nœuds de profondeur égale à quatre. Enfin, le même traitement sera répété avec une profondeur égale à 7.

1.4.2 Méthodes de routage basées sur le contenu des pairs

Ces méthodes sont basées sur les contenus partagés de pairs. Chaque pair maintient des indices de routage (ou des métadonnées) qui décrivent les contenus des autres pairs dans les réseaux. L'idée de base de ces méthodes est de choisir les pairs pertinents en fonction de la similarité entre le contenu de la requête et les métadonnées qui décrivent le contenu des autres pairs.

DFS : Depth-First Search

Contrairement à *Random Walk* qui sélectionne aléatoirement un seul voisin, dans DFS [Clarke 2001] le choix du voisin le plus pertinent pour la requête, se base sur la similarité de celle-ci avec les métadonnées qui décrivent le contenu de tous les voisins. Par la suite, si un pair n'a pas reçu une réponse de son voisin choisi, pendant une certaine période de temps, un autre voisin est sélectionné. Ce processus est répété jusqu'à ce que l'émetteur de la requête soit satisfait ou s'il n'a plus des voisins à sélectionner. Lorsqu'un pair possède une réponse à une requête, il retourne le résultat en suivant le chemin inverse de la requête.
Freenet [Clarke 2001] est un exemple de systèmes de recherche d'information en P2P (SRIP2P) basé sur DFS. Dans ce système, chaque pair maintient un index décrivant ses documents partagés et une table de routage contenant des informations sur ses voisins, comme par exemple : l'adresse IP et des clés décrivant chaque ressource partagée par les voisins. Ce système associe à chaque fichier une clé en appliquant une fonction de hachage sur les métadonnées de ce dernier. Pour rechercher un fichier dont la clé est égale à K, l'initiateur de la requête, un pair p_i, cherche tout d'abord les documents pertinents dans sa collection, s'il trouve un fichier qui possède la clé K, la recherche est terminée. Dans le cas contraire, il envoi la requête vers un pair voisin p_j qui possède une clé plus proche de K. En recevant la requête, p_j effectuera une recherche locale, si K est trouvée p_j retourne le résultat à p_i et la recherche est terminée. Sinon, p_j envoi la requête vers un pair voisin p_k qui possède une clé plus proche de K et ainsi de suite. Ce processus est répété jusqu'à ce que la valeur de TTL soit égale à zéro ou que le résultat de la recherche satisfasse l'utilisateur.

Routing Indices-Based Search

Dans Routing Indices-Based Search (Routing Indices : RI) [Crespo 2002], chaque pair maintient des indices de routage qui décrivent le contenu de chaque voisin. Cette

méthode classe les documents par thèmes. Pour chaque voisin p_j, elle garde le nombre de ses documents par thème ainsi que le nombre total de documents qui peuvent être récupérés à partir du "chemin" p_j. Un chemin est composé d'un voisin direct et les voisins qui se trouvent dans chaque saut jusqu'à une certaine profondeur h. Dans la littérature, il y a deux types de RI, à savoir :

1. Compound RI (CRI) [Crespo 2002]. Dans ce type de RI la "pertinence" d'un voisin p_j pour une requête donnée est évaluée sur la base du nombre de documents NUM_j qui pourraient être trouvés dans celui-ci. Avec l'hypothèse que les thèmes des documents soient indépendants, la valeur de pertinence NUM_j d'un voisin p_j pour une requête composée d'un ensemble de thèmes T pourrait être calculé comme suit :

$$NUM_i = n_j * \prod_{t \in T} \frac{CRI(n_{jt})}{n_j} \tag{1.1}$$

   Où $CRI(n_{jt})$ est le nombre de documents du thème t dans le voisin p_j et n_j est le nombre total de documents partagés par les pairs qui se trouvent dans le "chemin" p_j. CRI propage la requête vers les voisins de plus haut scores.

2. Hop-count RI (HRI) [Crespo 2002]. L'inconvenant de CRI, est que l'information indexée contient uniquement le nombre de documents qui peuvent être récupérés à partir d'un chemin, elle n'indique pas combien de sauts sont nécessaires pour trouver les documents. HRI a pour objectif de combler cette limite en ajoutant cette information. Dans cette méthode, chaque nœud maintient un CRI pour chaque saut en partant du voisin 1 au voisin h, où h est le nombre maximal de sauts. En effet, un HRI d'un nœud voisin est composé d'une série de CRI, dont chacune précise le nombre de documents qui peuvent être récupérés plus le nombre de sauts nécessaires. Étant donnée une requête, le calcul de la pertinence d'un voisin à une requête est basé sur le nombre de documents pertinents et le nombre de sauts nécessaires pour les trouver. L'inconvénient du HRI est qu'elle nécessite un espace de stockage important et un trafic réseau élevé.

Sunrise

Dans Sunrise [Mandreoli 2009], pour une représentation plus riche des données les pairs exposent leurs données partagées selon une ontologie locale (stockée au niveau de chaque pair). Chaque pair p_i établit un mappage sémantique de son schéma $MySchema_i$ avec les schémas de ses voisins. Supposons qu'un pair p_j soit un voisin du pair p_i, le mappage sémantique $Mp_i[p_j]$ de deux schémas $MySchema_i$ et $MySchema_j$ associe chaque concept dans $MySchema_i$ au concept correspondant dans $MySchema_j$ selon un score qui indique le degré de similarité sémantique entre ces deux concepts. En se basant sur les mappages sémantiques entre son schéma et les schémas de ses voisins, le pair p_i construit un index sémantique de routage $SRIP_i$. Chaque entrée $SRIP_i[p_j][c_k]$ contient un score de similarité de concept c_k dans les schémas des pairs p_i et p_j. Les lignes de cet index donnent une idée sur la pertinence des données qui peuvent être trouvées dans chaque direction. Lorsqu'un pair p_j reçoit une requête q, il

exploite son $SRIP_j$ pour classer ses voisins, afin d'identifier la direction la plus prometteuse à suivre. Sunrise introduit une sémantique de données en se basant sur une ontologie ce qui permet la reformulation des requêtes, par conséquent, elle présente de nouveaux challenges de routage de requêtes. En effet, le routeur peut reformuler la requête selon le schéma de son voisin afin que ce dernier puisse trouver des documents pertinents dans sa collection locale.

1.4.3 Méthodes de routage basées sur le regroupement des pairs

Ces méthodes organisent le réseau P2P en groupes de pairs qui partagent les mêmes préférences afin de construire un réseau logique sémantique (semantic overlay network). Les préférences de l'utilisateur sont déduites à partir de différentes sources d'informations (i.e., documents partagés, requêtes passées, etc.). En effet, dans un réseau sémantique chaque pair établit des liens avec les pairs ayant des intérêts proches. Dans le but de réduire le trafic sur le réseau, les requêtes sont d'abord acheminées directement au cluster correspondant, puis aux pairs de ce cluster. Dans la littérature plusieurs méthodes de routage basées sur le regroupement des pairs ont été proposées. Les différences majeures entre ces méthodes résident dans la construction de clusters.

GES

Dans *GES* [Zhu 2005], un nœud peut établir deux types de liens / connexions, à savoir des liens aléatoires et des liens sémantiques. Les liens aléatoires connectent des nœuds non proches (i.e. ne partagent pas des documents similaires), ces nœuds sont appelés des nœuds non pertinents. Tandis que les liens sémantiques organisent les nœuds proches en groupes sémantiques, ces nœuds sont appelés des nœuds pertinents. Pour ce faire, chaque nœud est représenté par un vecteur de termes (vecteur de nœud), qui résume ses documents. Ce vecteur est construit en se basant sur la technique (Vectoral Semantic Model : VSM) [Schütze 1997] qui affecte à chaque terme t dans le vecteur un poid w_t. Lorsqu'un nœud rejoint le système, il utilise d'abord un mécanisme de bootstrapping comme celui de Gnutella pour se connecter au reste du réseau. Au départ, il n'a aucune information sur le contenu des autres nœuds (i.e. les vecteurs des autres nœuds) ainsi que sur les groupes sémantiques existants. Alors, il lance une requête de recherche des pairs partageants des documents similaires à ses documents partagés. Cette requête est composée par son vecteur et un seuil de pertinence $REL\_THRESHOLD$, un seuil de réponses $MAX\_RESPONSES$ (les réponses sont un ensemble de nœuds avec leurs vecteurs) et la profondeur de recherche TTL. Cette requête sera propagée en utilisant la méthode *Random_ Walk*. En recevant cette requête, chaque nœud calcule un score de pertinence entre le vecteur de l'initiateur de la requête et son propre vecteur. Étant donnés deux nœuds p_i et p_j, le score de pertinence est calculé comme suit :

$$REL(p_i, p_j) = \sum w_{p_i,t} \cdot w_{p_j,t} \tag{1.2}$$

Avec t, un terme qui apparaît dans les vecteurs de deux nœuds p_i et p_j, $w_{p_i,t}$ le poids du terme t dans le vecteur du nœud p_i et $w_{p_j,t}$, le poids du terme t dans le

vecteur du nœud p_j. Après avoir calculé ce score, le récepteur de la requête retourne une réponse à l'initiateur de celle-ci contenant son vecteur et le score de pertinence. En recevant les réponses, l'initiateur de la requête établit des liens aléatoires avec les pairs ayant un score de pertinence inférieur à *REL_ THRESHOLD* (ces nœuds sont considérés comme non pertinents) et il établit des liens sémantiques avec les pairs jugés pertinents ayant un score de pertinence supérieur ou égal à *REL_ THRESHOLD*. Ces liens seront utilisés pour router les futures requêtes. En effet, le routeur utilise tout d'abord la technique *Biased walks*, plutôt que la technique *Random walks*, pour propager une requête à travers les liens aléatoires. Biased walks sélectionne un voisin parmi les nœuds aléatoires le plus pertinent par rapport à la requête. Le score de pertinence d'un voisin p_j à une requête q est calculé comme suit :

$$REL(p_j, q) = \sum w_{p_j,t} \cdot w_{q,t} \tag{1.3}$$

Avec t un terme qui apparait dans le vecteur du nœud p_j et la requête q, $w_{p_j,t}$ le poids du terme t dans le vecteur du nœud p_j et $w_{q,t}$ le poids du terme t dans la requête q. Une fois un pair p_j possédant des ressources pertinentes est trouvé (celui-ci est appelé "semantic group target node"), il propage la requête vers tous ses liens sémantiques. Chaque voisin sémantique évalue la requête, puis l'inonde à ses propres liens sémantiques. Pendant l'inondation, tous les nœuds au sein d'un groupe sémantique ou bien seulement un ensemble de nœuds peuvent être interrogés. Pour sélectionner uniquement un sous ensemble de nœuds, le "target node" peut fixer une profondeur de recherche TTL.

SKIP

Les auteurs de $SKIP$ [SHEN 2011] reprennent la même idée de regroupement de pairs proposée dans GES et développent un nouveau mécanisme de routage appelé $SKIP$. $SKIP$ remplace le principe d'inondation de requête dans le groupe de pairs sémantiques utilisé dans GES par une technique appelée K-itération préférence. En effet, dans $SKIP$ chaque nœud ordonne ses voisins selon leurs valeurs de pertinence (la valeur de pertinence de deux pairs est calculée selon la formule 1.2). Ensuite, les voisins sémantiques de ce nœud sont divisés en m sous-groupes, chaque sous-groupe contient $[M/m]$ voisins, M étant le nombre de voisins sémantiques. Après avoir localisé le "target node " avec la technique *Biased Walk*, le mécanisme $SKIP$ est utilisé pour propager la requête vers les voisins sémantiques du "target node". $SKIP$ route la requête aux voisins sémantiques du premier sous-groupe. Si un pair qui possède des documents pertinents est trouvé, il renvoie la réponse au "target node". Ce dernier renvoi la liste des documents à l'initiateur de la requête et décrémente la valeur de paramètre $MAX\_RESPONSES$ (ce paramètre est associé à la requête, il indique le nombre souhaité de réponses) par le nombre de documents trouvés. Si la valeur de $MAX\_RESPONSES$ est inférieure ou égale à zéro, le " target node " stoppe le processus de propagation. Sinon, il propage la requête au second sous-groupe de voisins sémantiques. Ce processus sera répété jusqu'à ce que le seuil de nombre de documents recherchés soit atteint ou bien que tous les groupes soient explorés.

Stratégies de regroupement pour les réseaux sémantiques

Les auteurs de [Raftopoulou 2009] ont présenté un framework pour l'étude de l'impact de la stratégie de regroupement sur la qualité du routage. Une stratégie générique de création des groupes et plusieurs variantes de celle-ci ont été présentées et évaluées. D'après [Raftopoulou 2009], dans un système basé sur SON (semantic overlay networks), lorsqu'un pair p_i rejoint le réseau, il commence d'abord par catégoriser ses propres documents dans une ou plusieurs catégories. Par conséquent, p_i peut avoir un ou plusieurs intérêts I_{ik}, avec k le nombre d'intérêts du pair p_i. Les intérêts d'un pair p_i sont stockés dans une liste d'intérêts L(p_i). Un intérêt est représenté par un index de termes si un système de référencement externe est utilisé pour classer les documents, ou par un vecteur qui représente le centroïde d'un cluster si les documents sont catégorisés par un algorithme de clustering. Pour chaque intérêt I_{in}, p_i maintient un index de routage RI_{in} où chaque entrée de cet index est de la forme $(ip(p_j), I_{jk})$, avec $ip(p_j)$ l'adresse IP d'un pair p_j qui possède I_{jk} proche de I_{jk}. Par la suite, p_i établit des connexions appelées "short-range links" avec les pairs du même cluster, il établit ainsi des connexions appelées "long-range links" avec des pairs qui appartiennent à d'autres clusters. Ce sont des pairs qui n'ont pas les mêmes intérêts que p_i. Pour simplifier la présentation des différents protocoles, on suppose que chaque pair p_i possède un seul intérêt I_i.

Protocole de regroupement

Ce protocole est utilisé pour remplacer les anciennes connexions par des nouvelles plus intéressantes. Chaque pair p_i exécute ce protocole à chaque changement d'intérêt en calculant la similarité intra-cluster avec la formule suivante :

$$NS_i = \sum_{\forall p_j \in RI_i} Sim(I_i, I_j) \tag{1.4}$$

Avec, I_j l'intérêt du pair p_j et $sim()$ une fonction de similarité appropriée. La similarité de voisinage (neighbourhood similarity) NS_i est utilisée comme une mesure de cohésion entre les clusters. Si NS_i est supérieure à un certain seuil τ alors p_i n'a pas besoin d'établir de nouvelles connexions. Dans le cas contraire, p_i lance une requête de recherche de nouveaux voisins appelée *FINDPEERS(*$ip(p_i)$*, *I_i*, L, R)*, L étant une liste d'intérêts et R le TTL du message. La liste L est initialement vide et elle va contenir des tuples de la forme $<ip(p_i), I_j>$ (chaque entrée de la liste contient l'adresse IP et l'intérêt d'un pair trouvé).

Lorsqu'un pair p_j reçoit le message *FINDPEERS()*, il ajoute son adresses IP, $ip(p_j)$, ainsi que son intérêt I_j (ou l'intérêt le plus similaire à I_i dans le cas où p_j possède plusieurs intérêts), décrémente R de 1 et propage le message vers m voisins qui ont les intérêts les plus similaires à I_i. La propagation de ce message est stoppée lorsque R atteint la valeur zéro, dans ce cas le message contenant la liste des pairs trouvés est retourné à l'émetteur en suivant le chemin inverse. Cette technique de propagation est appelée dans la littérature *gradient walk (GW)* [Sacha 2006]. À la réception de message, le pair émetteur p_i remplace les anciennes connexions avec les nouvelles connexions.

Les auteurs de [Raftopoulou 2009] ont constaté que la stratégie de propagation, du message *FINDPEERS()*, *Gradient Walk* n'est pas adéquate lors du démarrage du système où les clusters ne sont pas encore formés ce qui rend la probabilité de trouver des intérêts similaires très faible. Pour palier les limites de cette approche, ils ont proposé d'utiliser la stratégie *Random Walk(RW)* présentée dans la section 1.4.1.
En outre, ils ont proposé une 3^{eme} stratégie appelée *combined strategy* qui combine les deux stratégies RW et GW. En effet, un pair p_j qui reçoit le message *FINDPEERS()*, le propage à $m/2$ pairs choisis d'une manière aléatoire et à $m/2$ pairs ayant des intérêts similaires à I_i. Cette stratégie a pour objectif de bénéficier des avantages de deux autres, RW si les clusters ne sont pas formés et GW pour explorer les voisins possédants des intérêts similaires.
Également, une $4^{ème}$ amélioration appelée *informed strategy* a été présentée. Elle consiste à propager le message vers m pairs choisis d'une manière aléatoire si les clusters ne sont pas formés. Dans le cas contraire, elle propage le message vers m voisins qui ont les intérêts les plus similaires à I_i.

1.4.4 Méthodes de routage basées sur l'historique des requêtes

L'idée de base de ces méthodes est d'utiliser les informations relatives aux requêtes passées (i.e. les pairs qui ont agit positivement, les documents téléchargés, etc.) pour construire une base de connaissances au niveau de chaque pair du réseau P2P. Pour propager une requête donnée, le routeur (le pair qui va propager la requête) applique un algorithme d'apprentissage afin de sélectionner les pairs susceptibles de fournir des réponses à la requête. La requête sera ensuite envoyée aux pairs sélectionnés. Ce processus sera appliqué récursivement par les pairs choisis jusqu'à l'atteinte d'une limite de propagation (i.e., TTL).

BFS dirigée et Intelligent Search

Dans BFS dirigée [Yang 2002], chaque nœud maintient quelques statistiques sur ses voisins telles que le nombre de réponses d'un pair voisin aux requêtes précédentes, le nombre de résultats obtenus pour les requêtes précédentes et le temps de latence pour la réception de ces résultats. Sur la base de ces statistiques, le nœud peut choisir les bons voisins. Quelques heuristiques basées sur les hypothèses suivantes ont été proposées :

- Choisir les voisins qui ont retourné le plus grand nombre de résultats ;
- Choisir les voisins qui sont les plus proches. La distance entre les pairs est calculée en se basant sur le nombre de sauts ;
- Choisir les voisins qui ont propagé le plus grand nombre de messages précédemment ;
- Choisir les voisins qui ont la plus courte file de messages.

L'avantage de BFS dirigée, c'est qu'elle envoie la requête uniquement à un nombre limité de voisins, par conséquent, elle diminue le nombre de messages. De plus, si un nœud est capable de choisir les bons voisins, la qualité des résultats est assurée.

L'inconvénient de cette technique est que les statistiques maintenues au niveau de chaque nœud sont trop pauvres. En effet, ces statistiques ne contiennent pas d'informations relatives aux contenus des requêtes. Pour pallier ce problème, Kalogeraki et al. [Kalogeraki Vana 2002] ont présenté une approche similaire mais plus complexe appelée recherche intelligente IS (*Intelligent Search*). Dans cette méthode, chaque pair ordonne ses voisins en fonction de leurs pertinences par rapport à la requête et seuls les voisins qui ont une valeur de pertinence élevée seront choisis. Pour mettre en œuvre cette technique, chaque pair construit un profil pour chaque voisin. Le profil contient les requêtes les plus récentes ainsi que le nombre des réponses pour chaque requête. Afin de limiter l'espace du stockage, la stratégie Least Recently Used (LRU) est utilisée pour remplacer les anciennes requêtes. Lors de la réception d'une requête q, un pair p_i effectue un classement en ligne, en se basant sur une fonction de pertinence de ses voisins, afin de choisir les nœuds les plus adéquats vers lesquels la requête doit être transmise. Pour calculer le rang de chaque voisin p_j, p_i recueille les informations à partir des profils et calcule la pertinence de p_j par rapport à q, comme suit :

$$RR_{pl} = \sum_{q_k \ answered \ by \ p_j} simQ(q_k, q)^{\alpha} * S(p_j, q_k) \tag{1.5}$$

$simQ(q_k, q)$ dénote la fonction de similarité entre les deux requêtes q_k et q. La fonction de similarité utilisée par les auteurs est cosinus qui calcule le produit scalaire des vecteurs des deux requêtes. $S(p_i, q_j)$ est le nombre de résultats retournés par p_i pour la requête q_j. En outre, α est un paramètre configurable qui est utilisé pour ajouter plus de poids aux requêtes les plus similaires. En utilisant cette technique, la recherche intelligente prend en considération le contenu de la requête et elle permet de donner un classement meilleur que BFS dirigée.

Adaptive Probabilistic Search

La recherche probabiliste adaptative, *Adaptive Probabilistic Search (APS)* est une méthode de recherche qui combine les techniques *K-walker* et la recherche probabiliste [Tsoumakos 2003]. La principale différence entre APS et *K-walker*, c'est que la méthode *K-walker* envoie la requête vers un ensemble de voisins aléatoirement choisis tandis que APS envoie la requête vers un ensemble de nœuds voisins choisis en se basant sur des scores. Les scores sont des probabilités qui sont calculées en se basant sur l'historique des réponses de chaque nœud. Initialement, la probabilité associée à tous les voisins est la même. Il existe deux approches de mise à jour de ces valeurs de probabilité, à savoir :

- L'approche optimiste. Dans cette approche, le système augmente les probabilités pour les voisins choisis et diminue ces probabilités uniquement si la recherche se termine par un échec ;
- Approche pessimiste. Dans cette approche, le système diminue les probabilités pour les voisins choisis et augmente ces probabilités si la recherche se termine avec succès.

Learning-based query routing : LBQR

Dans LBQR [Shi 2008], chaque pair assigne des poids à chaque connexion établie avec un autre pair. Ces poids sont calculés en se basant sur l'historique des requêtes passées. Pour se faire, deux mécanismes ont été proposés : *(i)* le routage des requêtes, (*ii*) la construction et la maintenance des indices de routage. L'algorithme de routage s'exécute lors de l'émission ou la réception d'une requête, son rôle consiste en premier lieu à déterminer un nombre k de pairs vers lesquels la requête sera propagée, et en second lieu il sélectionne les k meilleurs pairs en se basant sur leurs poids. Contrairement à $RBFS$ où tous les pairs impliqués dans processus de routage envoient la requête à K voisins, avec K un certain seuil prédefini, $LBQR$ calcule ce seuil selon la formule suivante :

$$K = \frac{\alpha}{Hop + 1} \tag{1.6}$$

α est une constante prédéfinie et Hop est la distance entre le pair courant et l'émetteur de la requête. Le pair émetteur de la requête envoie α copies de la requête et le pair situé à une distance égale à α de l'émetteur stoppe le processus de propagation. Un pair p_i qui désire propager une requête q_j, devrait choisir les K connexions candidates ayant les probabilités les plus élevées en se basant sur la formule suivante :

$$Prj(Ci) = \frac{weight(C_i, q_j)}{\sum_i^M weight(C_i, q_j)} \tag{1.7}$$

$weight(C_i, q_j)$ indique le poids assigné par le pair p_i à la connexion C_i pour la requête q_j. Si la valeur du poids $weight(C_i, Q_j)$ est inférieure à un certain seuil ε elle sera remplacée par ε.

Route Learning

Route Learning [Ciraci 2009] exploite l'historique des requêtes pour sélectionner un sous ensemble de pairs voisins vers lesquels les requêtes seront propagées. Pour ce faire, chaque pair construit une base de connaissances à partir des requêtes envoyées vers les pairs voisins et les réponses de ces derniers. L'idée de base de cette approche est dérivée du problème de classification. En effet, chaque pair maintient un future espace (feature space), qui consiste en un tableau à deux dimensions, relatif à chaque voisin. Les colonnes de ce tableau représentent les termes des requêtes passées, la position (la colonne) de chaque terme est déterminée à travers une fonction de hachage qui place les termes sémantiquement proches dans des positions adjacentes. La première ligne du tableau représente le nombre de réponses retournées par le voisin p_j, qui contiennent le terme en colonne. La seconde ligne représente le nombre de requêtes envoyées au voisin p_j, contenant le terme en colonne. Durant la phase d'apprentissage et après la création des futures espaces relatifs à ses voisins, chaque pair met à jour ses futures espaces lors de l'émission d'une requête ou à la réception des réponses. Il est à noter que durant cette phase, l'algorithme utilise la méthode classique BFS pour router les requêtes. Dans la phase d'évaluation *Route Learning* applique une version adaptée de la technique d'estimation de densité *Parzen Windows* [Duda 2000] en se basant

sur les connaissances construites dans la phase d'apprentissage afin de sélectionner un nombre "*tosendPeers*" de voisins supposés pertinents. Pour ce faire, l'algorithme proposé calcule l'estimation *Parzen Windows* pour chaque voisin afin de choisir les *tosendPeers* voisins qui possèdent les valeurs d'estimation les plus élevées. Le calcul de cette estimation pour un voisin donné p_j à une requête q se fait de la manière suivante : pour chaque terme t de q l'algorithme recherche ce terme dans le future espace du voisin p_j puis il place une fenêtre de taille *Widowssize* déterminée empiriquement sur le future espace de telle sorte que l'indice du terme soit le centre de la fenêtre. Ensuite, il calcule une probabilité appelée "*containing-probabilty*" de terme t qui est égale à la somme des ratios du nombre de réponses retournées par le nombre de requêtes envoyées de toutes les cellules couvertes par la fenêtre. Enfin, l'estimation *Parzen Windows* pour le voisin p_j à la requête q est calculée selon la formule suivante :

$$P(V) = \frac{\sum_{t \in q} containing - probabilty\ de\ t}{\frac{Le\ nombre\ de\ termes\ dans\ le\ future\ espace}{Widowssize}} \tag{1.8}$$

La requête est routée vers les "*tosendPeers*" voisins qui possèdent les valeurs d'estimation les plus élevées. Si le nombre de pairs sélectionnés est inférieur à *tosendPeers*, l'algorithme ajoute un certain nombre de voisins de manière aléatoire.

1.4.5 Synthèse sur le routage de requêtes dans les systèmes P2P non structurés sur Internet

Les méthodes de routage basées sur le contenu donnent de bons résultats en termes de qualité et de nombre de documents trouvés. En plus, le processus de recherche des pairs pertinents n'est pas coûteux en nombre de messages échangés pour localiser ces pairs. En effet, les indices de routage au niveau de chaque pair synthétisent le contenu des autres, ce qui permet de choisir les bons pairs en fonction de la requête. Cependant, la construction de ces indices nécessite un coût supplémentaire en termes de trafic réseau et de traitement au niveau de chaque pair. Les pairs échangent un nombre assez important de messages pour construire et mettre à jour les indices de routage. La mise à jour des indices peut engendrer un temps de traitement élevé si la taille de l'index est importante. De plus, les utilisateurs peuvent ajouter ou supprimer des documents à tout moment ce qui rend les indices obsolètes ou inconsistants. Ces inconvénients freinent le passage à l'échelle des méthodes basées sur le contenu.

Les méthodes basées sur le regroupement de pairs organisent le réseau P2P en groupes de pairs qui partagent les mêmes préférences afin de construire un réseau logique sémantique (semantic overlay network). Dans un réseau sémantique chaque pair p_i établit des connexions appelées "short-range links" avec les pairs du même cluster (i.e., les pairs ayant des intérêts proches de ceux de p_i), ainsi, il établit des connexions appelées "long-range links" avec des pairs qui appartiennent à d'autres clusters (i.e., se sont des pairs qui n'ont pas d'intérêts proches de ceux de p_i). Les requêtes sont d'abord routées à travers les liens "long-range links" pour localiser les bons clusters, ensuite une méthode basée sur l'inondation est utilisée pour rechercher les pairs pertinents dans

chaque cluster. Contrairement aux méthodes basées sur le contenu, ces méthodes nécessitent un coût acceptable pour construire les clusters. Elles améliorent l'efficacité et le coût de la recherche si le pair lance des requêtes similaires aux documents partagés, mais elles donnent de mauvais résultats dans le cas contraire, car chaque pair possède peu de liens "long-range links" avec les pairs qui appartiennent à d'autres clusters.

L'idée sous-jacente de toutes les méthodes basées sur l'historique des requêtes est de remplacer la méthode classique *RBFS* qui propage la requête à K voisins choisis de manière aléatoire par une méthode qui propage la requête vers K pairs pertinents choisis en se basant sur des connaissances déduites à partir des requêtes passées. Le base commune de ces méthodes est la construction progressive des bases de connaissances par pair. Ces méthodes sont plus avantageuses que celles basées sur le contenu ou sur le regroupement de pairs car la construction des bases de connaissances ne nécessite pas un trafic réseau supplémentaire. Dans la littérature, peu de méthodes ont été proposées, elles améliorent l'efficacité et la performance de recherche des méthodes classiques. Cependant, elles souffrent de trois principaux inconvénients, à savoir :

- **Représentation du profil de l'utilisateur** : La représentation du profil de l'utilisateur varie selon les approches et les applications. Dans le domaine de la recherche d'information, le profil de l'utilisateur décrit le plus souvent son centre d'intérêt. Dans les méthodes de routage basées sur l'historique de requêtes étudiées, le profil de l'utilisateur est généralement représenté par des statistiques sur les requêtes passées et les pairs qui ont retournés des réponses pour celles-ci (représentation plate) [Kalogeraki Vana 2002, Christoph 2004, Shi 2008, Ciraci 2009]. Cette représentation est facile à mettre en œuvre, cependant, elle n'exploite pas les corrélations entre les requêtes passées et les pairs qui agissent positivement à ces requêtes. En effet, la représentation plate ne permet pas de déterminer les relations entre les requêtes passées (par exemple les requêtes qui ont des termes communs) et les pairs qui ont répondu à des requêtes connexes. Nous pouvons exploiter ces relations afin de mieux identifier les pairs pertinents pour une requête donnée ;
- **Problème du démarrage à froid** : Au démarrage du système, la base de connaissances est vide. De ce fait, l'algorithme de routage n'arrive pas à sélectionner des pairs pertinents pour les premières requêtes lancées par l'utilisateur. Les méthodes existantes ne traitent pas ce problème [Kalogeraki Vana 2002, Christoph 2004, Shi 2008, Ciraci 2009]. Lors du démarrage, elles propagent aléatoirement les requêtes dans le réseau ce qui génère un nombre de messages très important et des réponses non satisfaisantes pour les utilisateurs. IL serait intéressant de substituer le démarrage à froid par un démarrage à chaud en utilisant une base de connaissances initiale qui reflète les intentions de l'utilisateur ;
- **Problème d'échec de sélection** : Lorsque l'algorithme de routage sélectionne un nombre insuffisant de pairs, toutes les méthodes existantes ajoutent aux pairs sélectionnés un certain nombre de voisins choisis aléatoirement à partir de la table de routage. Par conséquent, le processus de routage est partiellement sémantique, ce qui dégrade les performances de la méthode de routage. Par contre,

d'autres pairs dans le réseau maintiennent des connaissances qui peuvent être utilisées pour extraire des meilleurs pairs pour la dite requête. Il serait intéressant de propager la requête vers ces pairs plutôt qu'aux voisins dont nous n'avons pas une idée ni sur leurs documents partagés ni sur leurs bases de connaissances [Kalogeraki Vana 2002, Christoph 2004, Shi 2008, Ciraci 2009].

Le tableau 1.2 compare les différentes méthodes étudiées selon différents critères, que nous avons défini :

- Passage à l'échelle : par passage à l'échelle, nous voulons savoir si la méthode permet de supporter un grand nombre de pairs (i.e + +), un nombre moyen (i.e + -) ou peu (i.e - -) ;
- Maintenance : fréquence des mises à jour des informations sémantiques. La fréquence peut être importante (i.e + +) ou plus ou moins importante (i.e + -) ;
- Espace de stockage utilisé : c'est l'espace mémoire utilisé pour stocker les informations sémantiques ;
- Partage de connaissances : c'est l'absence ou la présence de coopération des pairs pour construire des connaissances ;
- Efficacité de la recherche : une méthode de routage est efficace si elle peut localiser plus de documents pertinents ;
- Performance de la recherche : une méthode de routage est performante si elle propage un petit nombre de messages pour localiser les documents pertinents ;
- Coût de construction des indices de routage : les méthodes de routage sémantique maintiennent au niveau de chaque pair des indices de routage qui sont utilisés pour propager les requêtes. Si la construction de ces indices nécessite un grand trafic réseau, la méthode de routage ne permet pas le passage à l'échelle du système P2P.

1.5 Conclusion

Dans ce chapitre, nous avons présenté les systèmes P2P, leurs caractéristiques et quelques domaines d'application de ceux-ci. Ensuite, nous avons présenté en détail les systèmes de partage de fichiers en P2P et les différentes architectures sur lesquelles ils reposent, et pour chaque architecture nous avons présenté ses avantages et ses inconvénients. De même, nous avons présenté un état de l'art sur le routage de requêtes dans les systèmes non structurés purs déployés sur Internet. Dans le chapitre suivant, nous présentons un état de l'art sur les systèmes P2P mobiles de partage de fichiers ainsi que les méthodes de routage de requêtes dans ces systèmes.

	Méthodes basées sur le contenu :	**Méthodes basées sur le regroupement de pairs**	**Méthodes basées sur l'historique de requête**
	CORI, GLOSS, PlanetP, Sunrise, etc.	**GES, CSS, PROSA, SKIP, etc**	**BFS, IS, REMINDIN, LBQR, Route Learning, etc.**
Passage à l'échelle	–	++	++
Maintenance	++	+-	+-
Espace de stockage utilisé	++	–	+-
Partage de connaissances	oui	oui	non
Efficacité de la recherche	Recherche efficace	Recherche inefficace si l'utilisateur lance des requêtes non similaires à ses documents partagés	Recherche inefficace au démarrage de système ou s'il y a un échec de sélection de pairs pertinents à partir de la BC locale
Performance de la recherche	Bonne performance de l'algorithme de recherche	Recherche non performante si l'utilisateur lance des requêtes non similaires à ses documents partagés	Recherche non performante au démarrage de système ou s'il y a un échec de sélection de pairs pertinents à partir de la BC locale
Coût de construction des indices de routage	Un coût elevé et non évitable	Coût acceptable	Aucun coût

TABLE 1.2 – Une étude comparative de méthodes de routage existantes

Chapitre 2

Les systèmes pair à pair mobiles et les méthodes de routage

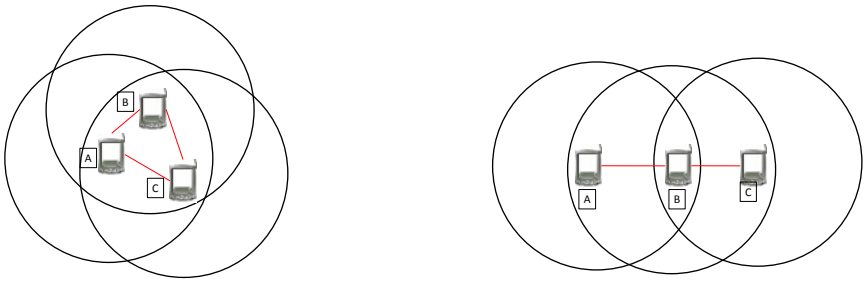
2.1 Introduction

Les systèmes P2P mobiles (Mobile P2P systems) soulèvent des nouveaux challenges de routage de requêtes. En effet, dans ces systèmes il y a plusieurs contraintes liées aux supports de transmission ou bien aux dispositifs mobiles. Par conséquent, l'utilisation des méthodes de routage développées pour les systèmes P2P sur Internet dans les systèmes P2P mobiles implique de mauvais résultats en termes d'efficacité et de performance. Dans la littérature, plusieurs travaux de recherche ont tenté de développer des nouvelles méthodes de routage de requêtes qui s'adaptent aux mieux à ce type de système.

Dans ce chapitre, nous présentons en premier lieu les réseaux mobiles adhoc MANETs ainsi que les nouvelles contraintes de déploiement des systèmes P2P sur ces réseaux. En second lieu, nous décrivons les nouveaux défis de routage des requêtes dans les systèmes P2P non structurés mobiles. Ensuite, nous développons un état de l'art sur les méthodes de routage de requêtes dans ces systèmes.

2.2 Les réseaux mobiles adhoc

Un réseau adhoc mobile (MANET) est principalement constitué par des dispositifs mobiles (i.e. PDA, des téléphones mobiles, etc.) qui communiquent entre eux en utilisant des technologies de communication sans fil comme Bluetooth [Bluetooth 2012], Wifi [Wifi 2012], etc. Contrairement aux réseaux avec infrastructure (i.e. Internet), la communication dans un réseau MANET se fait d'une manière totalement décentralisée sans avoir aucune infrastructure de communication préexistante et fixe. Ces réseaux sont particulièrement utiles dans les scénarios où l'infrastructure existante n'est pas fiable, trop cher ou tout simplement inexistante. Dans MANET, chaque nœud du réseau peut communiquer directement avec ses voisins, c'est-à-dire ceux qui sont à la portée de communication de sa propre interface. Pour communiquer avec un nœud qui est en dehors de la zone couverte par sa transmission radio, les messages de communication sont acheminés par des nœuds intermédiaires qui jouent le rôle de relais entre l'émetteur et le destinataire. La figure 2.1 (a) illustre un exemple d'un graph complet, c'est le cas idéal, où tous les nœuds peuvent communiquer directement. Dans la figure



Réseau MANET avec graph complet (a) Réseau MANET avec graph incomplet (b)

FIGURE 2.1 – Exemples de réseaux MANET

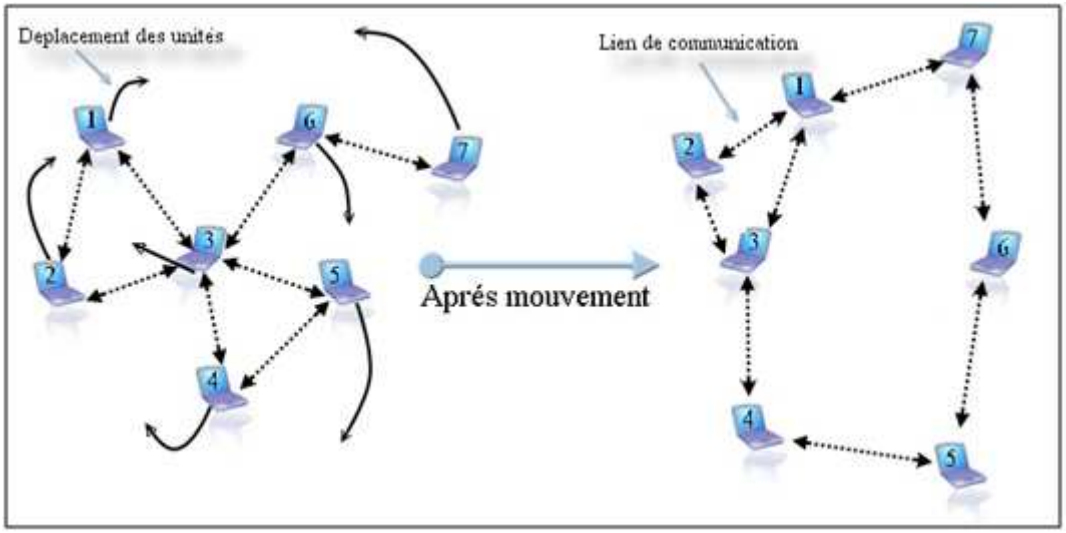


FIGURE 2.2 – Le changement de la topologie dans MANET

2.1 (b), le mobile A ne pourra pas échanger des informations avec le mobile C puisqu'ils ne sont pas connectés directement, alors le nœud B pourra relayer les messages du mobile A vers le mobile C.
Un aspect clé qui distingue les réseaux MANETs des réseaux traditionnels avec infrastructure (filaires ou sans fil), c'est qu'ils n'ont pas une topologie physique fixe, les nœuds sont mobiles et s'organisent de façon arbitraire. La mobilité crée également une topologie du réseau hautement dynamique, où les nœuds entrent et sortent de la couverture radio d'autres nœuds. La figure 2.2 illustre un exemple de changement de la topologie dans MANET.

2.2.1 Contraintes du réseau MANET

Bien que les réseaux MANETs soient rapidement déployables et ne nécessitent aucune infrastructure préexistante et fixe, ils souffrent de plusieurs contraintes liées

aux supports de transmission ou bien liées aux dispositifs mobiles. Les principales contraintes sont :

Limitations dues au support de transmission

- ***Partage du support de transmission :*** Les stations mobiles opèrent sur la même bande de fréquence. Le partage du support de transmission peut engendrer des collisions. Ce problème est lié également à la diffusion des signaux ;
- ***Taux d'erreur élevé :*** Les réseaux sans fil utilisent des ondes radio pour communiquer. Ces ondes peuvent subir des perturbations électromagnétiques, solaires, etc. Ces perturbations affectent les signaux transmis et constituent des sources d'erreurs ;
- ***Capacité variable des liens :*** Les caractéristiques et les performances des liens entre deux stations varient constamment. Ainsi, les performances de certains liens bidirectionnels peuvent chuter les réduisant ainsi en liens unidirectionnels ;
- ***Faible débit :*** Le débit dans le réseau MANET est faible en comparaison avec celui du débit dans les réseaux filaires. Par conséquent, la faiblesse de débit génère des problèmes de communication dans certaines applications.

Limitations dues aux stations mobiles

- ***Faible puissance :*** Les stations mobiles sont conçues pour une utilisation mobile. De ce fait, elles doivent être légères et surtout capables de fonctionner avec des batteries de petites tailles, par conséquent, leurs puissances de batteries sont limitées ;
- ***Durée d'utilisation restreinte :*** Les batteries ont une durée de vie limitée. De ce fait, le temps d'utilisation d'une station est limité par la capacité de sa batterie et aussi par la puissance de son processeur ;
- ***Zone de couverture :*** La zone de couverture est proportionnelle à la puissance d'émission que peut fournir une station mobile. Réduire la puissance d'émission, pour notamment économiser de l'énergie, peut engendrer des liens unidirectionnels ;
- ***Mobilité du nœud :*** Dans les réseaux MANETs, les pairs peuvent se déplacer librement. Par conséquent, le voisinage d'un nœud peut varier continuellement : à tout moment des stations peuvent être joignables directement ou non. La mobilité des nœuds engendre un changement fréquent de la topologie du réseau ce qui pose des problèmes de communication entre les nœuds.

2.2.2 Protocoles de routage dans les réseaux MANETs

Les protocoles de routage de paquets sont utilisés pour déterminer un chemin entre un nœud source et un nœud destination et assurer la transmission des données entre eux. Le routage de paquets dans les réseaux MANETs est un problème crucial vu les contraintes de ce réseau. En effet, dans un réseau MANET, il existe des liens asymétriques car les nœuds sont caractérisés par différentes puissances de transmission. Ainsi,

les informations de routage déterminées pour une direction donnée, peuvent être non valide pour la direction inverse. En outre, comme nous avons mentionné précédemment, la topologie est très dynamique. Les nœuds peuvent se déplacer à tout moment produisant des changements des routes en vigueur. Les informations de routage existantes deviennent alors incorrectes, ce qui force l'algorithme de routage à recalculer de nouveaux les itinéraires. De plus, les nœuds dans un réseau MANET ont des ressources limitées (i.e. la capacité de traitement, de mémoire, la puissance de la batterie, etc.). L'optimisation de l'utilisation des ressources devient une priorité.

Les réseaux mobiles adhoc ont attiré l'attention de la communauté des chercheurs et des dizaines de protocoles de routage ont été proposés pour MANET. Une description détaillée des protocoles de routage MANET peut être trouvée dans [Chlamtac 2003]. Dans la suite, nous allons mettre en lumière les principales caractéristiques de conception de ces protocoles, les avantages et les inconvénients plutôt que de décrire les algorithmes en détail.

D'après [Jochen 2003], les protocoles de routage pour MANET peuvent être classés en trois catégories : *routage ad hoc plat*, *routage hiérarchique* et *routage géographique*.

Les protocoles de routage ad hoc plats

Dans un protocole de routage ad hoc plat, il n'existe aucune hiérarchie de nœuds. En effet, tous les nœuds ont des rôles de routage équivalant au sein du réseau. Cette catégorie de protocoles est subdivisée en protocoles de routage réactifs et proactifs. Les protocoles proactifs maintiennent des tables de routage au niveau de chaque nœud et les mettent à jour indépendamment des besoins réels de communication du nœud courant. En d'autres termes, un protocole proactif maintient des routes vers des nœuds destinataires auxquels il n'a pas de paquets à envoyer. Des algorithmes basés sur l'état des liens sont couramment utilisés pour mettre en œuvre des protocoles proactifs. Dans ce contexte, les informations de routage sont périodiquement échangées entre les nœuds voisins via le principe de diffusion. Le principal avantage des protocoles proactifs est la configuration rapide de la connexion. Lorsqu'un nœud source donné doit transmettre un paquet à un nœud destinataire, il y a une probabilité élevée qu'un chemin vers ce nœud destinataire existe déjà dans la table de routage. Donc, les paquets ne prennent pas un long délai pour atteindre la destination à cause du processus de découverte de chemin. En outre, si les modifications de la topologie sont rares, chaque nœud aura une vue assez précise de la topologie du réseau, y compris les coûts (i.e., la latence, la bande passante) associés au chemin qu'il va choisir. Malheureusement, une surcharge inutile est nécessaire pour mettre à jour les tables de routage. En effet, les mises à jour au niveau des tables de routage sont échangées sans l'existence de besoins de communication. Compte tenu de la mobilité des nœuds et de leurs ressources limitées, le trafic supplémentaire va encore consommer l'énergie de batterie. De ce fait, Il est peut être difficile de définir un compromis adéquat entre la fréquence des mises à jour des tables de routage, afin de maintenir une vision précise de la topologie en cours, et la préservation des ressources qui sont déjà limitées. Les protocoles DSDV [Perkins 1994]

et OLSR [Jacquet 2001] sont les protocoles proactifs les plus connus pour MANET.

Les protocoles réactifs constituent une alternative aux protocoles proactifs. Un protocole réactif évite le trafic réseau nécessaire pour mettre à jour les tables de routage en découvrant les chemins sur demande. En d'autres termes, quand un nœud source désire envoyer un paquet à un nœud destination, le protocole de routage réactif initiera un processus de recherche pour découvrir un chemin entre les deux nœuds. Les chemins découverts sont mis en cache au niveau de chaque nœud et retirés de la table de routage, après une certaine période d'inactivité. L'avantage de ce type de protocoles est qu'ils limitent la maintenance des tables de routage uniquement aux routes demandées. Ceci offre une solution qui passe à l'échelle "scalable" si un nœud a des tendances à communiquer avec le même ensemble de nœuds et si la mobilité de nœuds est limitée. De plus, les nœuds peuvent mieux conserver la charge de leurs batteries à cause de l'absence de mises à jour périodiques des tables de routage. L'inconvénient majeur de ce type de protocole est le temps nécessaire pour configurer une nouvelle connexion lors de l'envoi du premier paquet à un voisin. Cela peut poser des problèmes pour les applications interactives. AODV [Perkins 2003] et DSR [Johnson 2001] sont les exemples les plus représentatifs des protocoles réactifs.

Les protocoles de routage ad hoc hiérarchiques

Les protocoles de routage ad hoc hiérarchiques regroupent les nœuds ayant les mêmes motifs de mouvement dans les mêmes clusters. Ces protocoles sont basés sur l'idée que les membres d'un groupe ont tendance à se rapprocher et donc un nœud restera probablement dans le même cluster. Après la construction des clusters, un ou plusieurs nœuds sont choisis comme nœud-maître, ces nœuds jouent le rôle des passerelles. Ils sont responsables de l'acheminement des paquets vers et à partir d'autres clusters. Le nœud-chef est élu suivant différents critères et informations sur le réseau : le niveau de l'énergie, la position géographique, etc. Ce type de protocole permet de réduire la taille de la table de routage au niveau de chaque nœud. De plus pour acheminer les paquets inter-clusters ou intra-clusters, il est possible d'utiliser plusieurs types de protocoles de routage. Les inconvénients de ce type de protocole résident dans la maintenance de la structure des clusters dans un environnement fortement dynamique. En outre, les nœuds maîtres consomment plus de ressources comme l'énergie de batterie et CPU. HSR [Kumar 2012] est le protocole de routage hiérarchique le plus connu.

Les protocoles de routage ad hoc géographiques

Les protocoles de routage ad hoc géographiques, utilisent les informations sur les positions géographiques des nœuds, pour améliorer les performances de routage. La position d'un nœud peut être obtenue par différents systèmes de localisation, le plus utilisé est le système de positionnement global (GPS). Un exemple de cette catégorie de protocoles est le protocole GPSR [Karp 2000]. Dans ce protocole, chaque nœud échange périodiquement des informations sur sa position avec ses voisins. Pour atteindre le nœud destination, les paquets sont propagés au voisin le plus proche.

2.3 Les systèmes P2P non structurés mobiles de partage de fichiers

En dépit de la popularité des deux technologies P2P et MANET plusieurs travaux de recherche ont récemment exploré les synergies qui existent entre les deux technologies. Plusieurs caractéristiques communes peuvent être identifiées entre les réseaux P2P non structurés purs et les MANETs. En effet, dans un réseau P2P non structuré pur et MANET il n'y aucun pair central chargé de faciliter l'échange d'information ou la coordination de l'activité du réseau. Tous les pairs doivent collaborer entre eux pour transmettre les informations de l'un à l'autre afin de garantir le fonctionnement de base du système. En outre, tous les participants assument les mêmes rôles dans le réseau : initiation, transmission et traitement des requêtes. Un autre aspect commun des deux réseaux est le principal problème qu'ils tentent de résoudre, c'est le routage efficace des requêtes.

De plus, pour intégrer les deux approches, il existe des différences significatives qui présentent des défis supplémentaires. En effet, les systèmes P2P construisent des réseaux logiques au niveau de la couche application, tandis que les MANETs ne concernent que les couches inférieures du modèle OSI. En raison de cette séparation claire des couches, la topologie du réseau logique P2P actuel est largement différente de celle du réseau physique sous-jacent. La figure 2.3 illustre un exemple d'une topologie de réseau P2P différente de celle du réseau physique. Les systèmes P2P classiques supposent l'existence d'un mécanisme de routage IP efficace permettant une communication performante entre les pairs du réseau logique qui peuvent être géographiques éloignés. Ces systèmes sont conçus pour Internet où les pairs sont souvent statiques, disposent des performantes ressources (i.e., la bande passante réseau, la puissance de traitement et la capacité de stockage) et des liaisons réseau très fiables. Aucune de ces hypothèses n'est valable pour les réseaux MANETs. En effet, dans MANET la communication directe n'est possible qu'entre voisins physiques et la communication entre les pairs distants peut être assurée par un protocole de routage opérant dans la couche réseau. Cependant, malgré la diversité des protocoles de routage de la couche réseau conçus pour MANET, tous les protocoles existants se basent sur la diffusion de messages. De ce fait, si on applique le principe d'inondation utilisé dans les systèmes P2P non structurés purs sur MANETs, cela se traduit par plusieurs diffusions de messages au niveau du réseau physique. Par exemple, dans la figure 2.3, si le pair 1 désire envoyer une requête à son voisin "logique" 11, cette requête sera diffusée par les pairs intermédiaires plusieurs fois jusqu'a atteindre le pair 11. La communication entre les nœuds distants en utilisant le routage "multi-sauts" dans MANET conduira a un débit beaucoup plus faible et une utilisation accrue du réseau. Cela ne pose pas un problème dans un système P2P déployé sur Internet où le coût de communication entre n'importe quels nœuds est supposé le même.

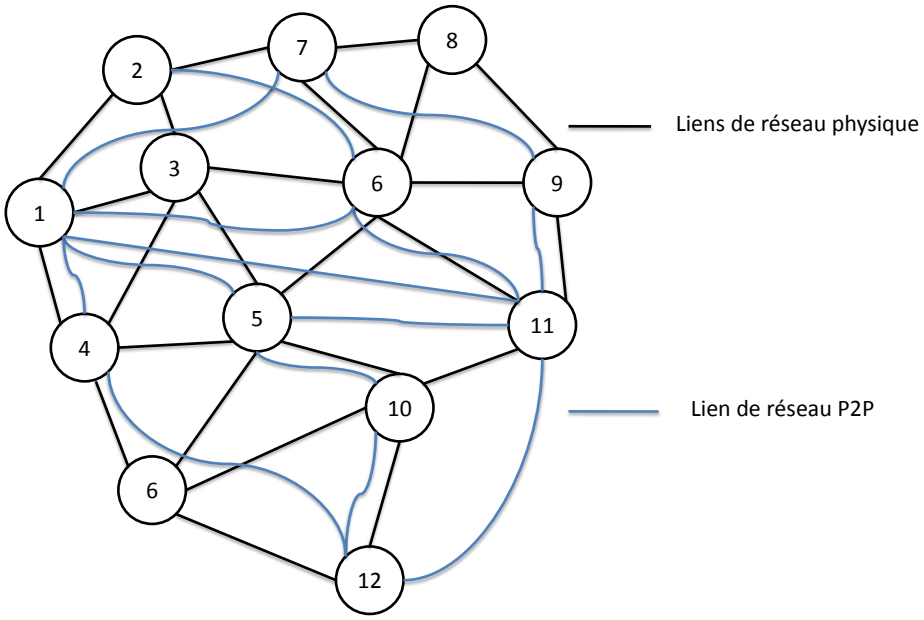


FIGURE 2.3 – Différence entre réseau P2P et réseau physique sous-jacent

2.4 Routage de requêtes dans les systèmes P2P non structurés mobiles

Dans la littérature, il existe deux approches de protocoles de routage de requêtes pour les systèmes de partage de fichiers en P2P mobiles. La première approche consiste à adapter les protocoles de routage de requêtes des systèmes P2P déployés sur Internet pour les systèmes P2P mobiles. L'idée est de garder le même principe de la méthode de routage dans la couche application et d'utiliser un protocole de routage MANET existant de la couche réseau pour échanger les requêtes et les réponses. En effet, le protocole de routage dans la couche application sélectionne les bons voisins logiques vers lesquels la requête sera envoyée, puis il utilise un protocole de routage existant implémenté dans couche réseau (i.e., DSDV [Perkins 1994], OLSR [Jacquet 2001], AODV [Perkins 2003], DSR [Johnson 2001], etc) pour localiser ces voisins. Les méthodes proposées ne sont pas performantes due aux contraintes du réseau MANET. Le découplage des fonctionnalités de la couche application et la couche réseau respecte bien le modèle OSI, ce qui permet un développement indépendant au niveau des deux couches. Cependant, cette séparation est coûteuse dans ce type d'application. En effet, au niveau de la couche application, les liens entre les voisins sont des liens logiques, mais un voisin logique n'est pas forcement un voisin physique. Par conséquent, lorsque la méthode de routage dans la couche application envoie une requête donnée à un voisin logique qui peut être physiquement hors de la portée radio de l'émetteur, cela se traduit par plusieurs dif-

fusions dans la couche réseau (en utilisant un protocole de routage Adhoc existant). Ce cas est très fréquent compte tenu de la mobilité des nœuds. En outre, les méthodes proposées pour les systèmes P2P sur Internet ne prennent pas en considération des informations contextuelles, comme par exemple la charge de la batterie, la capacité de traitement du nœud, la charge du nœud mobile, la mobilité, etc.

Une deuxième approche consiste à intégrer les fonctionnalités de la couche application avec celle de la couche réseau. Elle considère que les ressources dans MANET sont limitées, donc la performance de la méthode de routage est plus importante que la portabilité et la séparation des fonctionnalités. Dans ce contexte, des méthodes de routage de requêtes intégrées ont été proposées dans la littérature. Dans ce qui suit, nous présentons un état de l'art sur ces méthodes.

2.4.1 Optimized Routing Independent Overlay Network (ORION)

Dans cette méthode [Klemm 2003], le réseau logique est identique au réseau physique sous-jacent. En effet, *ORION* combine la couche application avec le protocole réactif *AODV* qui opère dans la couche réseau. Pour mettre en œuvre *ORION*, chaque dispositif mobile maintient une collection locale, constituée d'un ensemble de fichiers. Les auteurs supposent que chaque fichier est associé à un identifiant unique du fichier. En outre, *ORION* dispose de deux tables de routage, une table de routage de réponses et une table de routage des fichiers. La première table maintient le nœud à partir duquel un message de requête a été reçu. Elle sera utilisée pour router les réponses vers le pair qui a lancé la requête sachant que les réponses suivent le chemin inverse de la requête. La deuxième table est une sorte de cache qui stocke les identifiants des fichiers et les pairs associés. *ORION* met à jour à la fois la table de routage des réponses et la table de routage des fichiers lors du traitement de la requête. La consommation de mémoire pour les tables de routage est contrôlée en limitant la taille maximale et en appliquant la politique de remplacement Least Recently Used (LRU).

Pour rechercher les documents pertinents à une requête donnée, *ORION* utilise une technique de routage simple qui diffuse le message de requête à tous les voisins. Le processus de diffusion s'arrête lorsqu'une profondeur de recherche TTL est atteinte. Pour éviter les cycles, chaque pair diffuse une seule fois la même requête. Lorsqu'un pair reçoit la requête, il cherche, dans sa collection locale et sa table de routage de fichiers, les documents pertinents. S'il possède des documents pertinents, il renvoie une réponse à l'émetteur de la requête contenant les identifiants des fichiers. Le message des réponses suit le chemin inverse de la requête. Lorsqu'un nœud intermédiaire reçoit une réponse, il met à jour sa table de fichiers en ajoutant les identifiants de fichiers dans la réponse ainsi que la source de ceux-ci. De plus, il retire de la réponse, les identifiants des fichiers qu'il a déjà envoyé vers l'émetteur de la requête dans des réponses précédentes.

ORION utilise une technique de routage simple basée sur la propagation aléatoire de requêtes, comme BFS. Par conséquent, cette approche génère un trafic réseau très

important. En outre, dans *ORION*, il n'existe pas une méthode de sélection intelligente des voisins (i.e., chaque pair envoie la requête à tous ses voisins physiques). Par conséquent, la requête sera envoyée à plusieurs pairs non pertinents ce qui engendre une charge inutile à ces pairs. De plus, *ORION* ne considère aucune contrainte du réseau MANET comme la mobilité des pairs, la charge de batterie, etc.

2.4.2 On Localized Application-Driven Topology Control for Energy-Efficient Wireless Peer-to-Peer File Sharing : *TCP2P*

Dans [Leung 2008], les auteurs proposent un protocole de contrôle de topologie de systèmes P2P mobiles, appelé *TCP2P*. Ce protocole construit un réseau logique proche du réseau physique sous-jacent. En effet, le but de TCP2P est de construire une topologie logique qui permet de résoudre les problèmes de consommation d'énergie et la surcharge du réseau MANET. Pour ce faire, chaque nœud mobile s choisit un sous ensemble de voisins logiques, parmi ses voisins physiques, en se basant sur une fonction de préférence qui considère les quatre facteurs suivants : l'équité "fairness", "l'agressivité", la popularité des fichiers partagés par un pair mobile et le niveau de batterie restant d'un nœud mobile. Le premier facteur permet d'exclure le choix d'un voisin n qui a aidé à l'acheminement d'un nombre important de paquets aux autres pairs. Le deuxième facteur "l'agressivité" détermine si s est un nœud égoïste, demande toujours des fichiers mais il ne partage rien. Cela réduit la volonté d'un nœud voisin n à relayer les paquets de s. Le troisième facteur est la popularité de fichiers partagés par s. En effet, le voisin n doit tenir compte du nombre des fichiers qui seront téléchargés à partir de s. Si s possède un grand nombre de fichiers ou un des fichiers très populaires, alors il peut y avoir plusieurs pairs qui demandent le téléchargement de fichiers de s. Cela permettrait non seulement d'occuper la bande passante de s, mais aussi de charger ses voisins et consommer leurs énergies car ceux-ci contribuent au transfert de paquets transmis par ou vers s. Le dernier facteur considéré est le niveau d'énergie restant des nœuds. Être un nœud relais (qui transmet les fichiers) nécessite une consommation d'énergie plus importante que d'être simplement un nœud client (qui reçoit des fichiers) ou serveur (qui sert des fichiers). Il est évidemment non souhaitable de demander à un pair avec un très faible niveau d'énergie restant d'être un relais. Les auteurs proposent donc l'utilisation d'une métrique relative, c'est-à-dire, si un nœud voisin n a un niveau d'énergie supérieur à celui de s, il est souhaitable que n soit choisit comme voisin de s.

Dans *TCP2P*, la méthode de routage des requêtes est basée sur l'inondation comme la méthode *BFS*. Elle utilise le réseau logique pour propager les requêtes. Cette méthode propage la requête seulement vers un sous ensemble de voisins physiques (i.e., les voisins qui ont été choisis comme voisins logiques). De ce fait, elle réduit le nombre de messages. De plus les voisins sont choisis selon différentes métriques qui garantissent une utilisation optimale d'énergie et évitent la surcharge des pairs mobiles. Cependant, les requêtes sont propagées indépendamment de leurs contenus et le facteur de mobilité des pairs est non considéré.

	voisin 1	voisin 2	....	voisin i
type de fichier 1	$pv_{n1,type1}$	$pv_{n2,type1}$	....	$pv_{ni,type1}$
type de fichier 2	$pv_{n1,type2}$	$pv_{n2,type2}$	....	$pv_{ni,type2}$
....	....	....	....	....
type de fichier j	$pv_{n1,typej}$	$pv_{n2,typej}$	....	$pv_{ni,typej}$

FIGURE 2.4 – Exemple d'une table de phéromone

2.4.3 P2P file sharing system over MANET based on Swarm Intelligence (P2PSI)

P2PSI [Hwa 2009] repose sur deux processus : un processus de publication et un processus de découverte. Le processus de publication permet à chaque pair qui partage des ressources (HotSpot) d'annoncer périodiquement un message *seed* contenant des résumés des fichiers partagés. Lorsqu'un nœud reçoit ce message, il met en cache les informations associées. Le processus de découverte de ressources ou bien de routage de requêtes est basé sur une technique d'intelligence artificielle appelée *Swarm Intelligence* [Hwa 2009]. En effet, chaque nœud maintient une table de phéromones qui enregistre l'intensité de la phéromone de chaque lien avec ses voisins. Intuitivement, l'intensité de la phéromone d'un lien, est la probabilité d'acheminement d'une requête par l'intermédiaire de ce voisin. Une intensité de phéromone forte a une probabilité plus élevée. Un exemple d'une table de phéromones est illustré dans la figure 2.4, où chaque ligne est un type de fichier et chaque colonne est un voisin. Les entrées de la table de phéromones indiquent l'intensité de la phéromone de différents liens, appelées $pv_{n,t}$, où n désigne le nœud voisin et t représente le type de fichier. $pv_{n,t}$ exprime la chance de choisir n dans le prochain saut lors de la recherche des fichiers de type t. La table de phéromones maintient le type du fichier au lieu de son identifiant pour éviter d'avoir une taille assez importante de cette table en raison d'un grand nombre de fichiers partagés. Les auteurs se basent sur l'hypothèse suivante : si un nœud partage un fichier de type t, il peut contenir plusieurs fichiers appartenant à ce type. Un nœud peut obtenir des informations concernant les tables de phéromones de ses voisins en recevant des messages *hello*. En effet, chaque nœud diffuse un message *hello* toutes les T_{hello} secondes. Si un nœud reçoit un message *hello* d'un nouveau voisin n, il ajoute les informations de ce nœud dans sa table de phéromones. D'autre part, si le nœud ne reçoit pas de messages *hello* de son voisin pendant une période prédefinie, les informations de ce voisin seront supprimées de sa table de phéromones. Pour rechercher un fichier, le nœud mobile (demandeur) vérifie tout d'abord l'existence des informations valides mises en cache (n'ont pas expirées) concernant le dit fichier. S'il n'existe pas d'informations valides, il génère un certain nombre de requêtes pour rechercher le fichier désiré. Le nombre de requêtes à générer dépend de l'état actuel du réseau, le demandeur vérifie que si le

maximum de $pv_{n,t}$ est inférieur à un seuil MIN_{PV}, il génère NUM_{QUERY} requêtes. Dans le cas contraire, le demandeur crée une seule requête. Ensuite, le demandeur sélectionne les voisins auxquels la requête sera transmise en fonction de la probabilité de $P_{n,t}$, qui dénote la probabilité de transmission d'une requête à un voisin n lors de la recherche d'un fichier de type t. L'équation 2.1 décrit comment transformer l'intensité de phéromones $pv_{n,t}$ maintenue au niveau de la table de phéromones en une probabilité de transmission $P_{n,t}$:

$$P_{n,t} = \begin{cases} \frac{pv_{n,t}}{\sum_{i \in N - Vnodes} pv_{i,t}}, & si \quad n \notin Vnodes \\ 0, & Sinon \end{cases} \tag{2.1}$$

N représente l'ensemble des voisins et $Vnodes$ désigne l'ensemble des nœuds déjà visités par la requête. Notez que si aucune information n'est disponible pour les fichiers de type t, la requête est transmise à des voisins choisis d'une manière aléatoire.

Lorsqu'un nœud intermédiaire n reçoit une requête, il ajoute d'abord son identifiant à $Vnodes$ et incrémente le paramètre *hopCnt* (c'est un paramètre similaire à TTL) de un. Ensuite, le nœud n poursuit la transmission de la requête en se basant sur l'équation 2.1 si la valeur de hopCnt est inférieure à un certain seuil *limitQuery*. Si le nœud n possède le fichier demandé ou des informations concernant le fichier demandé existant dans son cache, il crée un message de réponse qui sera transmit en suivant le chemin inverse de celui de la requête. La réponse contient des informations concernant les fichiers trouvés et l'identifiant de leur source. Quand un nœud n reçoit un message de réponse d'un voisin m, il vérifie si le type de fichier "fileType" du message existe dans la table de phéromones. Si ce n'est pas le cas, le nœud n insère un nouveau type dans la table de phéromones. Ensuite, il transmet le message au prochain nœud. Si ce type existe, l'entrée $pv_{m,fileType}$ de la table de phéromones est renforcée par une intensité de phéromones δ calculée comme suit :

$$\delta = c \cdot (LIMIT_{query} - hopCnt) \tag{2.2}$$

Où c est un paramètre de conception et *hopCntr* représente le nombre de sauts que le message de réponse a traversé.

P2PSI route la requête à un ensemble de voisins choisis en se basant sur leurs contenus. De ce fait, cette méthode évite la diffusion de la requête à tous les voisins et la route uniquement aux pairs susceptibles de fournir une réponse. Cependant, plusieurs facteurs importants (i.e., charge de batterie, charge de pair, mobilité, etc.) ne sont pas considérés dans le choix des meilleurs voisins.

2.4.4 Protocol of Cross-layer Gnutella : *EAODV*

Ting Li et al [Li 2009] ont introduit un protocole de routage de requêtes basé sur un protocole de routage réactif pour les réseau MANET qui opère dans la couche réseau appelé *on-demand distance vector routing protocol (AODV)* [Perkins 2003]. L'idée

consiste à intégrer le protocole de la couche application Gnutella avec le protocole réseau *AODV*. Afin de réaliser l'interaction entre Gnutella et *AODV*, les auteurs ont étendu le protocole *AODV* pour définir un nouveau protocole appelé *EAODV* (enhanced *AODV*). Avec les extensions, *EAODV* est capable d'identifier les applications P2P et offre la possibilité de découvrir des fichiers ainsi que les routes. En effet, *EAODV* ajoute deux nouveaux identificateurs dans le message de demande de route standard (RREQ), "*flag tag*" et "*keyword*". "*flag tag*" est utilisé pour indiquer si ce message est une requête de découverte de fichiers ou tout simplement un message de demande de route. Si "*flag tag*" est égal à 1 le deuxième identificateur "*keyword*" doit contenir les mots clés de la requête. De même, ces deux identificateurs sont aussi ajoutés au message de réponse (RREP). L'identificateur "*flag tag*" spécifie le type du message RREP. Lorsqu'un fichier demandé est trouvé, l'application Gnutella informe le protocole *EAODV* qu'il doit répondre avec un message RREP, dans ce cas, le champ "*flag tag*" de RREP est égal à 1 et le champ "*keyword*" doit contenir l'identificateur du fichier trouvé. Les autres opérations de *EAODV* sont identiques à celles de *AODV* (comme le traitement et la transmission de messages RREQ, etc.).

L'intégration des protocoles *AODV* et Gnutella permet de construire un réseau P2P logique identique au réseau physique sous-jacent. Par conséquent, elle évite les problèmes des approches qui séparent les deux couches. Cependant, le mécanisme de propagation de requêtes utilisé par *EAODV* est basé sur la diffusion ce qui engendre un trafic réseau important. De plus il ne tient pas en compte la mobilité et l'énergie de batterie des pairs mobiles.

2.4.5 Context-aware routing for peer-to-peer network on MANETs

Dans cette approche, chaque nœud mobile maintient une collection locale de documents partagés, une table de réponses et une table de routage de fichiers [Shah 2009]. Chaque ligne de la table de réponses contient des informations concernant le nœud à partir duquel une requête a été reçue. Ces informations sont utilisées pour acheminer les réponses vers le nœud initiateur de la requête. La table de routage de fichiers stocke des métadonnées concernant des fichiers partagés et les pairs associés. La méthode de routage proposée est basée sur l'inondation. En effet, pour rechercher un fichier, le nœud mobile propage une requête appelée *RREQ* contenant les termes de recherche, le nœud source SRC et un nombre de séquences *SEQ* vers tous ses voisins. Ces deux derniers sont utilisés pour détecter les cycles. Lorsqu'un nœud reçoit la requête, il calcule la distance $D1$ entre lui et l'émetteur de la requête, en se basant sur la puissance du signal, et le temps $TS1$ de réception de la requête. Ces deux valeurs sont stockées dans la table de réponses. Ensuite, il recherche les documents pertinents dans sa collection locale (local repository). S'il possède des documents pertinents il renvoie une réponse *RREP* au nœud source, sinon, il utilise sa table de routage (une sorte de cache) pour rechercher des réponses avec une durée de vie minimum valide. S'il existe des réponses valides, il renvoie une réponse *RREP* et stoppe la recherche. Dans le cas où il n'y a pas de réponses dans la collection locale ou bien dans la table de routage, la requête est pro-

pagée de la même manière. La réponse $RREP$ suit le chemin inverse de la requête. Elle contient l'(s) identifiant(s) d'un ou de plusieurs fichiers pertinent(s) pour la requête, la distance $D1$ associée à l'émetteur de la réponse (cette valeur se trouve dans table de réponses du pair émetteur) et la valeur minimale de durée de vie MLT. Initialement, la durée de vie est égale à une valeur très grande. Lorsque un nœud intermédiaire reçoit la réponse, il calcule le temps de réception $TS2$ de celle-ci et la distance $D2$ entre lui et l'émetteur. Puis, il calcule la durée de vie du lien $Lifetime$ avec l'émetteur comme suit :

$$V = \frac{|D2 - D1|}{(TS2 - TS1)} \tag{2.3}$$

Dans cette équation, le temps $TS1$ est déterminé par le nœud qui a reçu la réponse (i.e., il dénote le temps de réception de la requête) et V dénote la vitesse de déplacement de l'émetteur de la réponse.

$$Lifetime = \frac{(TR - D2)}{V} \tag{2.4}$$

TR étant la portée de la transmission radio du nœud. En comparant la valeur de durée de vie calculée $Lifetime$ et la valeur minimale de durée de vie MLT qui se trouve dans la réponse, le minimum entre ces deux valeurs est choisi comme étant MLT du chemin. Par la suite, le nœud intermédiaire effectue l'une des décisions suivantes :

1. Si MLT est inferieur à RTT (Round Trip Time) la réponse sera annulée car il est impossible que l'émetteur de la requête récupère le fichier. De ce fait, cette méthode diminue la surcharge du réseau en éliminant les liens ayant une forte probabilité d'échec ;
2. Si le nœud courant est l'émetteur de la requête, la réponse $RREP$ et le MLT associé sont stockés dans la table de routage de fichiers. Sinon MLT et $D2$ seront remplacés dans la table de routage respectivement par la valeur de durée de vie calculée et $D1$. Ensuite, la réponse sera routée vers le nœud suivant.

L'utilisation de cache et le filtrage de réponses permettent de réduire le nombre de messages échangés, cependant la diffusion de requêtes à tous les voisins engendre une surcharge de réseau qui est caractérisé par un faible débit. De plus, tous les voisins sont impliqués dans le processus de routage et de traitement de la requête ce qui consomme plus d'énergie et temps CPU.

2.4.6 Gossiping-LB

Les auteurs de [daH 2009] ont proposé une méthode de routage adaptative basée sur la charge du réseau appelée *Gossiping-LB*. Contrairement aux méthodes [Klemm 2003], [Li 2009] et [Shah 2009] qui propagent les requêtes vers tous les voisins (broadcast), Gossiping-LB les route vers un sous ensemble de voisins les moins chargés. En effet, la sélection des voisins est basée sur la métrique "probabilité de propagation" (forwarding probability). La probabilité de propagation d'une requête à un voisin donné est calculée comme suit :

$$probabilité\ de\ propagation = p \times (1 - u) \tag{2.5}$$

Avec, p une valeur fixée par l'utilisateur et u $(0 \leq u \leq 1)$ dénote le taux d'utilisation du voisin. Cette métrique permet d'envoyer plus de messages aux voisins qui ont une faible charge, tandis que, moins de messages sont envoyés aux nœuds saturés.

Gossiping-LB réduit la consommation d'énergie et le trafic réseau. Cependant, il existe d'autres facteurs qui ont un effet sur le calcul de la charge des pairs et qui ne sont pas considérés (i.e., puissance de CPU). En outre, Gossiping-LB ne tient pas en compte la mobilité des pairs qui est un facteur très important. En effet, nous pouvons choisir un pair moins chargé mais très mobile dans le chemin de la requête. Par conséquent, il y a une forte probabilité que les réponses n'atteignent pas l'émetteur de la requête puisqu'ils vont suivre le chemin inverse de celle-ci qui est instable.

2.4.7 A Path-Quality-Aware Peer-to-Peer File Sharing Protocol for Mobile Ad-hoc Networks : Wi-Share

L'objectif de cette approche [Karasabun 2009] est de maintenir le débit global du réseau élevé en trouvant des chemins optimaux tout en minimisant l'utilisation de ressources d'énergies. En effet, lorsqu'un nœud reçoit une requête, il cherche dans sa collection locale les documents pertinents puis il retourne une réponse s'il existe des documents satisfaisants les critères de recherche. Par la suite, le nœud récepteur ajoute son identifiant et un paramètre de sa disponibilité "availability" dans le chemin de la requête. Ce paramètre de disponibilité est constitué de la puissance d'énergie courante disponible, le nombre de sauts entre lui et l'initiateur et la quantité de données envoyées, reçues ou transmises récemment. Les paramètres de disponibilité des nœuds dans un chemin sont utilisés pour calculer la qualité de celui-ci. Par conséquent, la qualité du chemin permet de prendre de meilleures décisions de routage. En effet, les auteurs combinent les paramètres de disponibilité des nœuds constituant le chemin pour calculer son coût, puis ils calculent la qualité du chemin qui est l'inverse de son coût. La formule 2.6 calcule le coût d'un chemin en se basant sur les disponibilités des nœuds :

$$PC = C_h \times HopCount + C_b \times (100 - Battery_{min}) + C_t \times Traffic_{max} \tag{2.6}$$

PC étant le coût du chemin, $HopCount$ le nombre de sauts, $Battery_{min}$ le niveau de batterie du nœud ayant la moins faible puissance dans le chemin, et $Traffic_{max}$ la charge de trafic du nœud le plus chargé. Ch, Cb et Ct sont des constantes représentant respectivement le nombre de sauts, le niveau de la batterie et la charge du trafic. Ces constantes sont utilisées pour ajuster la valeur de PC. Après le calcul de la qualité du chemin, le nœud décide de stopper la diffusion de la requête si elle a été diffusée précédemment et que la nouvelle qualité du chemin n'est pas meilleure que celle qui a causé la diffusion précédente.

Cette approche permet d'éviter la congestion des nœuds en favorisant les itinéraires les plus disponibles et en minimisant la surcharge du réseau par le contrôle de la diffusion de messages. De plus, elle prend en considération la contrainte de l'énergie de

batterie dans processus de routage de requêtes. Cependant, elle ne considère pas la mobilité des nœuds qui est un facteur très important dans MANET. En outre, la requête est diffusée à tous les voisins indépendamment de son contenu (i.e,. les mots clé de la recherche).

2.4.8 Synthèse sur le routage des requêtes dans les systèmes P2P non structurés mobiles

Dans la littérature peu de méthodes intégrées de routage de requêtes pour les systèmes P2P non structurés mobiles ont été proposées. Une première idée adoptée par les méthodes [Klemm 2003], [Li 2009] et [Shah 2009] consiste à utiliser le principe de diffusion pour propager les requêtes. En effet, chaque pair diffuse la requête à tous ses voisins physiques jusqu'à ce q'une certaine profondeur de recherche TTL soit atteinte. Dans ces méthodes, les réponses suivent le chemin inverse des requêtes. Les pairs intermédiaires peuvent mettre dans leur caches des informations concernant la requête et les fichiers trouvés ainsi que le nœud source pour avoir une idée sur le contenu des autres pairs. En effet, lorsqu'un pair reçoit une requête, il cherche dans son cache les fichiers pertinents et envoie une réponse à l'émetteur contenant la liste des fichiers et les chemins vers les pairs associés. Des mécanismes ont été proposés pour déterminer si les informations dans le cache sont valides. De plus, pour certaines méthodes, les pairs intermédiaires peuvent supprimer les réponses, si le chemin entre l'émetteur de la requête et le pair possédant le document pertinent devient instable, ce qui permet de diminuer la charge du réseau. L'utilisation de cache et le filtrage de réponses permettent de réduire le nombre de messages échangés. Cependant, la diffusion de requêtes à tous les voisins engendre une surcharge de réseau qui est caractérisé par un faible débit. De plus, tous les voisins sont impliqués dans le processus de routage et de traitement de la requête ce qui contribue à consommer plus d'énergie et de temps CPU. Egalement, lors de la propagation d'une requête, aucun choix n'a été fait sur les pairs voisins. De ce fait, le chemin entre l'émetteur de la requête et le pair qui possède des documents pertinents peut être instable, du à la mobilité des nœuds ou la faiblesse de la charge de la batterie, d'où une forte probabilité que la réponse qui va suivre le chemin inverse n'atteigne pas l'émetteur. Une deuxième idée, qui consiste à envoyer récursivement la requête vers un certain nombre de voisins choisis en se basant sur quelques statistiques, a été adoptée pour améliorer les méthodes basées sur la diffusion. En effet, des méthodes de routage comme [Hwa 2009], [Karasabun 2009] et [daH 2009] routent la requête vers les K meilleurs voisins en se basant sur le contexte de l'utilisateur comme le contenu de la requête, la puissance de la batterie, la charge du calcul du dispositif mobile utilisé ainsi que la mobilité des utilisateurs. Le choix des bons voisins dépend de l'efficacité de la fonction de choix de ses paramètres considérés. Contrairement aux méthodes basées sur la diffusion, ces méthodes réduisent le nombre de messages. Elles limitent le traitement des requêtes à un sous ensemble réduit de pairs ce qui permet d'éviter la surcharge du réseau et d'économiser ainsi les ressources d'énergie et de calcul. De plus, si la fonction de sélection prend en considération la mobilité des nœuds, nous pouvons garantir un chemin stable pour transmettre les réponses.

	Mobilité	Charge de pair mobile	Énergie de batterie	Contenu de la requête	Efficacité	Performance
ORION	non	non	non	non	non	non
TCP2P	non	oui	oui	oui	oui	+-
EAODV	non	non	non	non	non	non
[Shah 2009]	non	non	non	non	non	non
Gossiping – LB	non	oui	non	non	non	+-
P2PSI	non	non	non	oui	oui	non
WIShare	non	oui	oui	non	non	oui

TABLE 2.1 – Comparaison des méthodes de routage pour les systèmes P2P mobiles

Le tableau 2.1 compare les différentes méthodes étudiées selon différentes critères relatives au contexte de l'utilisateur, que nous avons défini :

- **Mobilité :** La mobilité des nœuds est un facteur très important. En effet, les méthodes de routage qui prennent en compte ce facteur peuvent garantir un chemin stable entre l'émetteur de la requête et la source de documents pertinents. Nous voulons savoir si la méthode de routage considère la mobilité des nœuds ou non ;
- **Charge de pair mobile :** La charge d'un pair mobile peut être déterminée en se basant sur plusieurs facteurs comme les messages en fil d'attente, les messages reçus ou transmis etc. Les méthodes qui considèrent ce facteur peuvent éviter le problème de congestion en envoyant les requêtes vers les pairs moins chargés ;
- **Énergie de la batterie :** L'énergie de la batterie nous permet de déterminer le temps de fonctionnent restant pour un dispositif mobile. Il est intéressant de router les requêtes vers les pairs ayant une énergie de batterie élevée ;
- **Contenu de la requête :** Le contenu permet de choisir les voisins qui possèdent des documents pertinents à la dite requête. Une méthode de routage qui prend en considération le contenu de la requête peut retrouver plus de documents pertinents ;
- **Efficacité de la recherche :** Une méthode de routage est efficace si elle peut localiser plus de documents pertinents ;
- **Performance de la recherche :** Une méthode de routage est performante si elle peut localiser rapidement les pairs pertinents.

2.5 Conclusion

Dans ce chapitre, nous avons présenté le réseau mobile adhoc MANET ainsi que les nouvelles contraintes de déploiement des systèmes P2P sur ce type de réseau. De même, nous avons décrit les difficultés de routage de requêtes dans les systèmes non

structurés mobiles et les différentes méthodes existantes dans la littérature. Dans le chapitre suivant nous allons présenter en détail les solutions proposées aux problèmes de représentation de profil d'utilisateur, de démarrage à froid et d'échec de sélection des pairs pertinents dans les méthodes de routage basées sur l'historique des requêtes pour les systèmes P2P sur Internet.

Chapitre 3

Solutions proposées pour les systèmes P2P sur Internet

3.1 Introduction

Dans le premier chapitre, nous avons déduit qu'aucune des approches existantes de routage supporte les trois exigences suivantes : efficacité et performance de la recherche et un coût acceptable de construction des indices de routage. En effet, nous avons montré que les méthodes de routage basées sur l'historique des requêtes sont plus avantageuses que les autres méthodes, car elles ne nécessitent pas un coût de construction des indices de routage. Cependant, (*i*) elles ne considèrent pas les relations entre les requêtes passées et les pairs associés pour représenter les profils des utilisateurs ; (*ii*) elles souffrent de problème du démarrage à froid et (*iii*) de problème d'échec de sélection des pairs pertinents à partir de la base de connaissances locale.

Dans la suite de ce chapitre un ensemble de points sont discutés, nous proposons :

- (*i*) un modèle de routage sémantique basé sur l'historique des requêtes. Contrairement aux méthodes de routage existantes, qui représentent le profil utilisateur par des statistiques sur les requêtes passées et les réponses associées, dans notre modèle, le profil de l'utilisateur est représenté par un ensemble d'intérêts. Chaque intérêt représente des relations sémantiques entre *les requêtes passées*, leurs *termes* et les *pairs positifs* (i.e., les pairs qui ont agi positivement à des requêtes passées). Ensuite, nous instancions le modèle pour définir notre propre méthode de routage par apprentissage (*Learning Routing Method* : *LRM*) ;
- (*ii*) une méthode prédictive de l'intension de l'utilisateur et qui construit une base des connaissances initiale pour chaque pair. Cette base des connaissances initiale est construite au démarrage de système ce qui permet de pallier le problème de démarrage à froid ;
- (*iii*) une méthode de routage hybride (*Learning Routing Method based on Clustred Network* : *LRMCN*) pour traiter le problème d'échec de sélection. Cette méthode est basée sur l'historique des requêtes et le regroupement de pairs dans des groupes sémantiques.

3.2 Modèle de routage par apprentissage

Notre modèle comprend trois principaux composants, à savoir : le composant "informations passées", le composant "gestion de profil" et le composant "sélection des

pairs pertinents".

Le composant "informations passées" décrit formellement les données collectées à partir des résultats de requêtes passées, comme les identifiants des requêtes, leurs termes, les documents téléchargés et les pairs associés, etc. Ces données peuvent être stockées dans un fichier journal ou une base de données. Les données brutes ne permettent pas d'exploiter le taux de répétition de termes dans les requêtes soumises et les relations entre eux. Par conséquent, l'évaluation d'une requête donnée par rapport aux données brutes ne permet pas de sélectionner la majorité des pairs pertinents. Pour exploiter les correlations entre les requêtes passées, nous devons construire une base des connaissances, en appliquant des techniques d'exploration de données, pour représenter les intérêts de l'utilisateur. En effet, la base de connaissances est décrite formellement par le composant "profil utilisateur". Enfin, le composant "sélection des pairs pertinents" présente un ensemble d'opérateurs qui permettent de sélectionner, à partir de la base des connaissances, les meilleurs pairs pour une requête donnée, vers lesquels elle sera routée.

Avant la description de notre modèle, nous définissons les notations suivantes :

- P : Un ensemble de pairs du réseau P2P ;
- Q : Un ensemble de requêtes envoyées par un pair $p \in P$.

3.2.1 Le composant "informations passées"

Nous modélisons une "information passée" par un ensemble de propriétés décrivant une requête. D'une manière formelle, nous définissons cette information par un n-uplet.

Définition 1 *Informations passées : nous définissons l'ensemble T des informations passées comme suit : $T=\{t_1, ..., t_n\}$ Où t_i est un n-uplet. Un n-uplet représente un groupe de champs connexes, il peut être défini par : $t = (champs_1 : type_1, champs_2 : type_2, ..., champs_n : type_n)$. Où $champs_i$ est une propriété de type $type_i$.*

Exemple 1 :

Dans un système P2P de recherche d'information, un n-uplet peut être vu comme suit : un n-uplet $t = (q : String, W_q : Set, Pos_q : Set)$, Où :

- $q \in Q$: Un identifiant d'une requête ;
- W_q : Un ensemble de termes de la requête q ;
- $Pos_q \subset P$: Un ensemble de pairs positifs qui ont répondu à la requête q.

3.2.2 Le composant "profil utilisateur"

Ce composant gère la base de connaissances locale de chaque pair. Nous présentons formellement la base de connaissances comme suit :

Définition 2 *Base de connaissances : une base de connaissances notée B est un ensemble d'intérêts de l'utilisateur, plus formellement : $B=\{I_1, ..., I_p\}$, avec I_i un intérêt de l'utilisateur.*

Définition 3 *Intérêt de l'utilisateur : Un intérêt de l'utilisateur $I_i \in B$ peut être modélisé par un triplet d'ensembles $I_i(E_i, F_i, G_i)$, avec :*

- *$E_i \subset Q$ est un sous-ensemble des requêtes passées qui ont des termes communs ;*
- *$F_i = \bigcap_{q \in E_i} W_q$ est l'ensemble des termes communs des requêtes qui appartiennent à E_i ;*
- *$G_i = \bigcap_{q \in E_i'}$ est un ensemble de pairs positifs qui ont répondu à toutes les requêtes d'un ensemble E_i', contenant la majorité des requêtes de E_i. Le cas idéal est de trouver un ensemble G_i des pairs positifs qui ont répondu à toutes les requêtes de l'ensemble E_i. Cependant, dans la pratique, il est possible de trouver des requêtes qui ont des termes partagés mais il n'existe pas de pairs en commun qui ont répondu à ces requêtes. Dans ce cas, G_i est vide. Pour palier ce problème, nous proposons de construire un ensemble G_i contenant les pairs qui ont répondu à la majorité des requêtes dans E_i. En d'autre terme, l'ensemble G_i contient les pairs qui ont répondu à un ensemble de requêtes E_i' proche de E_i.*

3.2.3 Le composant "sélection des pairs pertinents"

Nous définissons deux opérateurs et une fonction de similarité, qui sont utilisés pour sélectionner l'ensemble des pairs pertinents pour une future requête q.

Définition 4 *Opérateur de sélection noté $\sigma_{(c)}(B)$: Il est utilisé pour obtenir un sous-ensemble d'intérêts, à partir de la base des connaissances B, qui satisfont une condition de sélection c.*

Définition 5 *Opérateur de projection noté $\pi_{G_i}(I_i(E_i, F_i, G_i))$: Il permet d'extraire l'ensemble des pairs G_i à partir de l'intérêt I_i.*

Définition 6 *Fonction de similarité notée Sim : La fonction $Sim : Q \times B \longmapsto [0, 1]$ donne la similarité sémantique entre une requête $q \in Q$ et un intérêt $I_i \in B$ en se basant sur les termes communs. Une valeur élevée de la fonction Sim indique une grande similarité entre q et I_i. Si la valeur est égale à 0, q et I_i ne sont pas similaires.*

Dans la section suivante, nous allons instancier ce modèle pour définir notre méthode spécifique. Nous précisons comment nous définissons les informations passées, comment nous générons des intérêts sans recours à une coopération des pairs ce qui permet d'éviter des surcoûts de communication. Enfin, nous utilisons les opérateurs définis dans le modèle afin de sélectionner les pairs appropriés pour une requête donnée.

3.3 Méthode de routage par apprentissage : *LRM*

L'idée de notre démarche est de définir une méthode de routage sémantique basée sur un ensemble d'intérêts de l'utilisateur [Yeferny 2010]. L'objectif de notre méthode est double. Nous abordons les problèmes de l'efficacité et de la performance de la

recherche d'information. Par conséquent, nous essayons de réduire les coûts de communication afin d'avoir une méthode efficace et performante de routage pour les systèmes pair-à-pair non structurés.

3.3.1 Architecture globale

Dans notre proposition nous avons repris l'architecture du système Gnutella [Gnutella 2009] qui est le système P2P non structuré le plus connu. Dans le but d'améliorer l'efficacité de la méthode de routage de ce système, nous avons remplacé le processus de propagation aléatoire de requêtes par autre processus "guidé" qui utilise une base des connaissances afin de router les requêtes vers les pairs pertinents. L'architecture globale de notre approche se compose de trois couches :

- La couche de gestion des informations passées : Elle gère les informations relatives aux requêtes envoyées et les réponses associées ;
- La couche de gestion de profils : Elle permet de construire la base des connaissances à partir des informations extraites de l'historique des requêtes. Cet historique se trouve dans un fichier journal (log file) ;
- La couche de propagation de requêtes : Elle utilise la base de connaissances pour sélectionner les pairs pertinents. Ensuite, elle propage la requête vers les pairs sélectionnés à condition que la profondeur maximale de recherche ne soit pas atteinte.

3.3.2 La couche de gestion de informations passées

Chaque pair stocke des informations relatives aux requêtes passées dans un fichier journal local. À la reception des réponses d'une requête, cette couche met à jour le fichier log en ajoutant des informations relatives à celle-ci. Chaque ligne de ce fichier est un n-uplet du composant "informations passées" de notre modèle. Dans notre cas, nous avons besoin de trois propriétés : $\mathsf{t} = (q, W_q, Pos_q)$, avec q est un identificateur de requête, W_q un ensemble de termes de la requête et Pos_q l'ensemble des pairs positifs (i.e., les pairs qui ont agi positivement à la requête q).

3.3.3 La couche de gestion de profil

La base des connaissances est une instance du composant "profil utilisateur". En effet, notre objectif est de générer des ensembles d'intérêts qui représentent des relations sémantiques entre *les requêtes passées*, leurs *termes* et les *pairs positifs*. Rappelons que dans notre modèle, un intérêt utilisateur est un triplet $I_i(E_i, F_i, G_i)$, qui exprime des correlations entre un sous ensemble de requêtes E_i, leurs termes communs F_i et les pairs positifs associés G_i. Il y a une forte relation entre les trois ensembles E_i, F_i et G_i. Pour générer un intérêt de l'utilisateur, la première étape regroupe les requêtes qui ont des termes communs dans un ensemble E_i. La deuxième étape consiste à construire l'ensemble G_i qui contient les pairs positifs qui ont répondu à toutes les requêtes de l'ensemble E_i' (par hypothèse E_i' est l'ensemble le plus proche de E_i).

Maintenant, la question clé est comment trouver ces correlations ?

Pour répondre à cette question, nous avons adapté une approche formelle basée sur l'analyse formelle de concepts (Formal Concept Analysis : FCA) [Ganter 1997] pour générer ces correlations. En effet, due au volume important des données disponibles, il est nécessaire d'éliminer la redondance de données et de trouver les correlations intéressantes. Dans ce qui suit, nous présentons une brève description de la FCA.

Analyse Formelle de Concepts

l'Analyse Formelle de Concepts, utilise comme point de départ un «*contexte formel*» qui définit une relation binaire reliant les objets à leurs propriétés [Ganter 1997].

Contexte formel

Un contexte formel est un triplet $(\mathcal{O}, \mathcal{P}, \mathcal{R})$, décrivant deux ensembles finis $\mathcal{O}$ et $\mathcal{P}$ et une relation (d'incidence) binaire, $\mathcal{R}$, entre $\mathcal{O}$ et $\mathcal{P}$ telle que $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{P}$. L'ensemble $\mathcal{O}$ est habituellement appelé ensemble d'objets (ou instances) et $\mathcal{P}$ est appelé ensemble de propriétés (ou attributs). Chaque couple (o,p) $\in \mathcal{R}$ désigne que l'objet o $\in \mathcal{O}$ possède la propriété $p \in \mathcal{P}$ (noté aussi $o\mathcal{R}p$) [Ganter 1997].
Le tableau 3.1 illustre un exemple de contexte binaire, avec $\mathcal{O}$=$\{\mathcal{P}, \mathcal{C}, \mathcal{A}, \mathcal{F}, \mathcal{G}\}$ et $\mathcal{P}$=$\{v, n, p, m, b, me\}$. Ce contexte représente un échantillon de données décrivant différents types d'animaux et leurs caractéristiques.

$\mathcal{O}$-$\mathcal{P}$	vole (v)	nocturne (n)	plume (p)	migrateur (m)	bec-plat (b)	membrane (me)
palatouche ($\mathcal{P}$)	x					x
chauve-souris ($\mathcal{C}$)	x	x				
autruche ($\mathcal{A}$)			x			
flamant-rose ($\mathcal{F}$)	x		x	x		
goéland ($\mathcal{G}$)	x		x		x	

TABLE 3.1 – Illustration d'un contexte formel.

Correspondance de Galois (Connexion de Galois)

Supposons A $\subseteq \mathcal{O}$ et B $\subseteq \mathcal{P}$ deux ensembles finis. Pour les deux ensembles A et B, les opérateurs $\varphi(A)$ et $\delta(B)$ sont définis comme suit [Ganter 1997] :

- $\varphi(A)$:= { p | $\forall$ o, o $\in$ A et (o,p) $\in \mathcal{R}$ }.
- $\delta(B)$:= { o | $\forall$ p, p $\in$ B et (o,p) $\in \mathcal{R}$ }.

L'opérateur φ définit les attributs partagés par tous les éléments de A. L'opérateur δ définit les instances qui partagent les mêmes attributs de l'ensemble B. Les deux opérateurs φ et δ définissent la correspondance de Galois entre les deux ensembles A et

B. Les opérateurs de fermeture sont définis par $B"= \delta \circ \varphi(B)$ et $A"=\varphi \circ \delta(A)$. Enfin, le couple d'ensemble fermé (A,B) est défini par $B=\delta \circ \varphi(B)$ et $A=\varphi \circ \delta(A)$ [Ganter 1997].

Concept formel

Un concept associe un ensemble maximal d'objets à l'ensemble d'attributs que ces objets partagent. Un concept formel C du contexte $(\mathcal{O},\mathcal{P},\mathcal{R})$ est un couple (A,B) avec A$\subseteq\mathcal{O}$, B$\subseteq\mathcal{P}$ et φ(A)=B, δ(B)=A. Les ensembles A et B sont appelés respectivement l'extension (domaine), notée $Ext(C)$ et l'intention (co-domaine), notée $Int(C)$ du concept formel C [Ganter 1997].

Adaptation de la FCA pour construire la base de connaissances locale

Pour appliquer la FCA, nous nous sommes basés sur deux contextes qui sont des projections sur le fichier log. La première projection représente le lien entre les requêtes passées et les termes associés, appelé contexte $C1$. La seconde représente le lien entre les requêtes passées et les pairs positifs, appelé contexte $C2$. En effet, les attributs du contexte $C1$, (respectivement $C2$), sont les termes des requêtes (respectivement les pairs ayant répondu à ces requêtes). Un algorithme de génération des concepts formels est appliqué pour générer deux ensembles de concepts, notés $E1$ et $E2$. Par la suite, nous utilisons ces deux ensembles pour construire la base des connaissances.

Construction de l'ensemble $E1$

Les éléments de l'ensemble $E1$ sont des concepts formels qui représentent des corrélations sémantiques entre les requêtes passées. Pour construire cet ensemble, nous appliquons un algorithme d'extraction de concepts formels (par exemple Godin [Godin 1995]) sur le contexte $C1$. Les concepts générés sont sous la forme $(\{R_1,..,R_i\},\{T_1,..,T_j\})$ avec $\{R_1,..,R_i\}$ un ensemble d'identifiants de requêtes et $\{T_1,..,T_j\}$ un ensemble de termes.

Construction de l'ensemble $E2$

Les éléments de l'ensemble $E2$ sont des concepts formels qui représentent des corrélations sémantiques entre les pairs positifs. Pour construire cet ensemble, nous appliquons cette fois le même algorithme sur le contexte $C2$. Les concepts générés sont donc sous la forme $(\{R_1,..,R_i\},\{P_1,..,P_j\})$ avec $\{R_1,..,R_i\}$ un ensemble d'identifiants de requêtes et $\{P_1,..,P_j\}$ l'ensemble de pairs qui ont répondu à ces requêtes.

Construction de la base des connaissances

Rappelons que notre objectif est de générer un ensemble d'intérêts de l'utilisateur qui constituent la base de connaissances locale d'un pair donné. Chaque intérêt est un triplet $I_i(E_i,F_i,G_i)$, qui exprime des correlations entre un sous ensemble de requêtes E_i, leurs termes communs F_i et les pairs positifs associés G_i. Avec, G_i contenant un

ensemble de pairs qui ont répondu à un ensemble de requêtes $E_i{}'$ proche de E_i. Formellement, nous définissons la base de connaissances comme suit :

$$B = \left\{ \begin{array}{l} I_i(E_i, F_i, G_i) \quad / \exists c_i(E_i, F_i) \in E_1 \quad et \quad \exists c_j(E_i{}', G_i) \in E_2 \\ avec \quad Sim(c_i, c_j) = Max(Sim(c_i, c_k)) \\ \qquad\qquad\qquad\qquad\qquad\qquad c_k \in E_2 \end{array} \right\}$$

La similarité entre un concept $c_i \in E_1$ et un concept $c_j \in E_2$ est calculée comme suit :

$$Sim(c_i, c_j) = \frac{|Ext(c_i) \cap Ext(c_j)|}{|Ext(c_i) \cup Ext(c_j)|} \tag{3.1}$$

Avec : $Ext(c_i)$ et $Ext(c_j)$ représentent respectivement les extensions des concepts c_i et c_j.

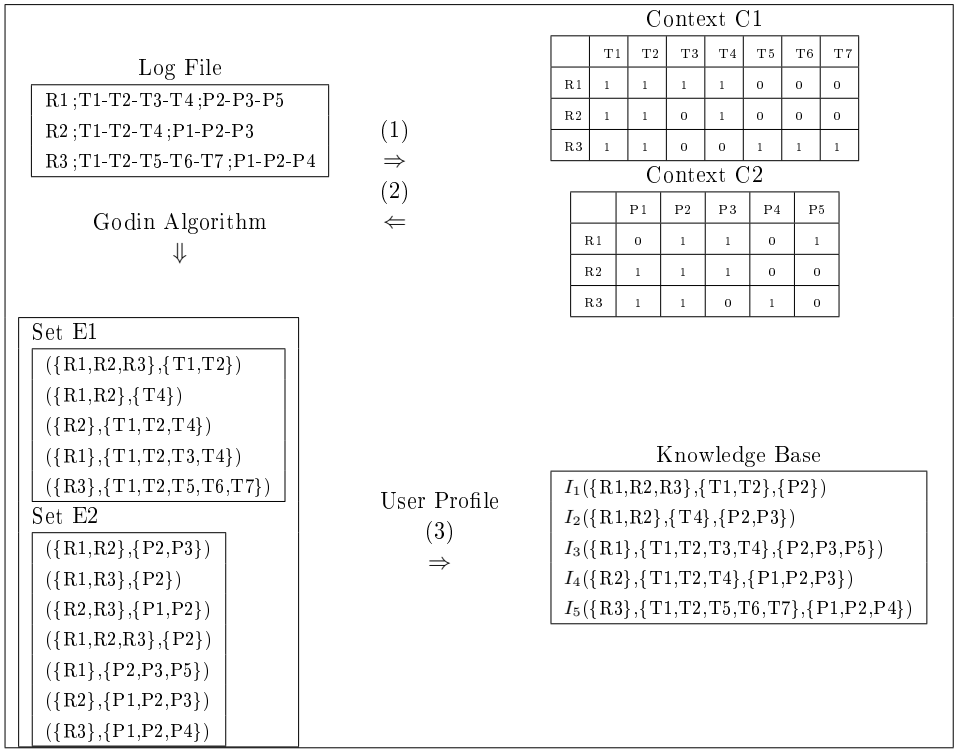
Avec cette approche adaptée de la FCA, nous assurons que chaque intérêt $I_i(E_i, F_i, G_i) \in B$ satisfait l'hypothèse précédente. En effet, les requêtes appartenant à l'ensemble E_i ont un ensemble F_i de termes en communs et G_i contient un ensemble de pairs qui ont répondu à un ensemble de requête $E_i{}'$ sémantiquement proche de E_i. La figure 3.1 représente les différentes étapes de construction de la base de connaissances B. La première étape de cette figure représente la génération de contexte à partir de fichier journal. La deuxième étape consiste à appliquer un générateur de concepts formels pour obtenir les deux ensembles E_1 et E_2. Enfin, la troisième étape consiste à générer la base des connaissances selon la spécification précédente en utilisant l'algorithme *User Profile* (voir Algorithme 1).

En effet, les entrées de cet algorithme sont les ensembles E_1 et E_2 et son objectif est de générer les intérêts de l'utilisateur qui seront stockés dans la base des connaissances B. La fonction $getCloseConcept(c_1, E_2)$ dans l'algorithme 1 (voir ligne 3 de l'algorithme 1) retourne le concept le plus similaire à c_1 à partir de l'ensemble des concepts de E_2. La similarité entre un concept c_i de E_1 et un concept c_j de E_2 est calculée selon la fonction de similarité 3.1

Algorithme 1: User Profile

Input:
E_1, E_2 : Ensembles de concepts
Output:
B : Base de connaissances

```
1 begin
2     for each c1 ∈ E1 do
3         c2 = getCloseConcept(c1, E2)
4         B = B ∪ {I(Ext(c1), Int(c1), Int(c2))}
5     return (B)
6 end
```



	T1	T2	T3	T4	T5	T6	T7
R1	1	1	1	1	0	0	0
R2	1	1	0	1	0	0	0
R3	1	1	0	0	1	1	1

	P1	P2	P3	P4	P5
R1	0	1	1	0	1
R2	1	1	1	0	0
R3	1	1	0	1	0

FIGURE 3.1 – Les étapes de construction de la base des connaissances

Mise à jour de la base des connaissances

Pour prendre en compte les nouvelles informations sur les requêtes passées, les bases des connaissances sont mises à jour périodiquement. Pour ce faire, nous avons défini deux stratégies (méthodes) de maintenance de ces bases :

- *Stratégie par rafraîchissement* : consiste à régénérer la base de connaissances de chaque pair à partir de l'historique de toutes les requêtes émises. Elle recalcule la base de connaissances périodiquement ;
- *Stratégie incrémentale* : consiste à générer une base B^+ à partir de l'historique des nouvelles requêtes : les requêtes émises après la dernière opération de mise à jour. Par la suite, nous construisons la nouvelle base de connaissances en fusionnant B^+ à l'ancienne base. Avec cette stratégie, le nombre d'objets et d'attributs nécessaire pour la construction de la base B^+ est largement inférieur à celui de la stratégie par rafraîchissement. Par conséquent, le temps de mise à jour sera inférieur à celui de la stratégie par rafraîchissement.

3.3.4 La couche de propagation de requêtes

Lorsqu'un pair désire router une requête donnée, cette couche appelle l'algorithme LPS (LearningPeerSelection) (voir ligne 2 de l'algorithme 2) pour sélectionner une liste des pairs pertinents. Si le nombre de pairs sélectionnés est inférieur à un certain seuil

prédéfini P_{max}, nous ajoutons un ensemble aléatoire de pairs à partir de la table des voisins (ligne 5 de l'algorithme 2).

Algorithme 2: ALGORITHME DE ROUTAGE

Input:
B : Base des connaissances
q : Une requête à propager
Nl : Liste de voisins
P_{max} : Le nombre de pairs à sélectionner pour une requête donnée

```
1 begin
2   finalList =LearningPeerSelection(B, q, P_max)
3   if (|finalList| < Pmax) then
4     N = P_max − |finalList|
5     addRandom(Nl, finalList, N)
6   Forward(finalList) ;
7 end
```

Algorithme de selection de pairs pertinents : LearningPeersSelection (LPS)

L'objectif de l'algorithme LPS est de sélectionner un ensemble d'intérêts de l'utilisateur, à partir de la base de connaissances, similaires à la nouvelle requête q. La similarité entre la requête q ayant une liste de termes Wq et un intérêt $I_i(E_i, F_i, G_i)$ est caractérisée par les termes en commun. Plus formellement, cette similarité est définie comme suit :

$$\begin{array}{rlll} Sim: & Q \times B & \longrightarrow & [0,1] \\ & q \times I_i(E_i, F_i, G_i) & \longmapsto & \frac{|Wq \bigcap F_i|}{|Wq \bigcup F_i|} \end{array} \tag{3.2}$$

À partir des intérêts sélectionnés, l'algorithme génère l'ensemble des pairs pertinents vers lesquels la requête est routée.

Presentation de l'algorithme LPS

Pour sélectionner les pairs pertinents, LPS (voir algorithme 3) se base sur la base de connaissances générée par la couche de gestion de profils. En effet, pour une requête donnée q, l'algorithme LPS extrait, à partir de la base de connaissances B, l'intérêt $I_i(E_i, F_i, G_i)$, le plus proche de la requête q en utilisant la fonction de similarité 3.2 (ligne 3 de l'algorithme 3). Par la suite, à partir de l'ensemble G_i nous déterminons la liste des pairs pertinents. Nous répétons ce procédé jusqu'à ce que le nombre de pairs sélectionnés soit égal à P_{max} ou il n'existe plus d'intérêts similaires à la requête dans la base des connaissances (lignes de 4 à 7 de l'algorithme 3).

Algorithme 3: Algorithme de sélection des pairs

Input:
B : La base des connaissances
q : L'identificateur de la requête
P_{max} : Le nombre de pairs à sélectionner
Output:
$peers$: Liste de pairs sélectionnés

```
1 begin
2     peers = ∅
      I_i   = σ(B)
3           Sim(q, I_i) = Max(Sim(q, I_k)) > 0, I_k ∈ B
4     while I_i and |peers| < P_max do
5         peers = peers ∪ π_(G_i)(I_i)
6         B = B − {I_i}
          I_i   = σ(B)
7               Sim(q, I_i) = Max(Sim(q, I_k)) > 0, I_k ∈ B
8     Return (peers)
9 end
```

3.4 Traitement de problème du démarrage à froid

Pour améliorer l'efficacité et la performance de recherche des méthodes basées sur l'historique des requêtes lors du démarrage du système ou durant la phase d'apprentissage, nous proposons de construire une base initiale de connaissances pour chaque pair [Yeferny 2012a, Yeferny 2013a]. Cette base initiale des connaissances est construite avant que l'utilisateur ne lance sa première requête de recherche. En effet, une phase d'apprentissage est implicitement exécutée au niveau de chaque pair qui rejoint le réseau. Par conséquent, nous substituons le démarrage à froid du système par un démarrage à chaud. Notre méthode est générique, elle peut être utilisée non seulement pour palier le problème de démarrage à froid de la méthode LRM, mais elle peut être également adaptée aux autres méthodes de routage par apprentissage existantes comme [Kalogeraki Vana 2002, Christoph 2004, Shi 2008, Ciraci 2009].

Pour construire une base de connaissances initiale B_{i_0} d'un pair p_i, nous devons prédire le profil de l'utilisateur avant qu'il ne lance ses premières requêtes. Pour cela, nous inondons un certain nombre de requêtes qui ont une relation sémantique avec les futures requêtes de l'utilisateur (intension de l'utilisateur), ensuite, nous utilisons les réponses à ces requêtes pour construire B_{i_0}. La question qui se pose est comment générer ces requêtes ?

Pour générer ces requêtes, nous exploitons la collection locale de documents partagés pour en extraire les requêtes représentatives. En effet, dans le monde réel, il existe

une forte relation entre les requêtes de recherche d'un utilisateur donné et ses documents partagés. Par exemple, un utilisateur qui partage des documents du domaine de mathématiques a de forte chance de lancer des requêtes sur des thèmes du même domaine.

Dans ce quit suit, nous présentons les différentes étapes de construction de la base des connaissances initiale, à savoir, l'extraction de requêtes représentatives à partir de la collection locale de documents et la construction de la base en se basant sur les réponses retournées.

3.4.1 Génération de requêtes

Algorithme 4: Extraction de requête

Input:
d : Document de la collection locale
t_{max} : Nombre maximum de termes de la requête
Output:
$query$: requête extraite

```
1  begin
2      t : terme
3      w_t : poids de terme t
4      index : liste contenant des tuples de la forme suivante < t, w_t >
5      index =<>
6      for each t ∈ d do
7          w_t = computeWeight(t, d, C)
8          index.add(< t, w_t >)
9      query = generateQuery(index, t_max)
10     return (query)
11 end
```

L'idée de notre proposition est de générer à partir d'un document partagé $d_i \in C$, (C dénote la collection locale d'un pair), une requête représentative q_i. Les termes de la requête doivent être représentatifs du document d_i. En d'autre termes, nous cherchons à maximiser la valeur de similarité $Sim(q_i, d_i)$, entre le document d_i et la requête q_i où Sim est une fonction de similarité appropriée. Dans notre cas, nous avons choisi la fonction de similarité cosinus [Makhoul 1999]. Pour un document $d_i \in C$, nous calculons le poids w_t de chaque terme $t \in d_i$ en utilisant la fonction $computeWeight(t, d, C)$ de l'algorithme 4. Le poids de chaque terme est calculé selon la mesure *tf-idf* [Makhoul 1999]. Ensuite, les premiers t_{max} termes ayant les poids le plus élevés forment une requête q_i représentative du document d_i (voir ligne 9 de l'algorithme 4).

3.4.2 Construction de la base de connaissances initiale

Pour construire la base de connaissances initiale d'un pair donné, nous sélectionnons au hasard K doncuments de sa collection locale, K est proportionnelle au nombre de documents partagés. Ensuite, une une requête est extraîte à partir de chaque document en utilisant l'alogorithme 4.

Les requêtes sélectionnées sont envoyées par la suite implicitement en utilisant la méthode classique de routage $RBFS$ [Lv 2002]. En recevant les résultats, le pair peut alors lancer la construction de sa base de connaissances initiale.

3.5 Méthode de routage hybride basée sur l'historique des requêtes et le regroupement de pairs ($LRMCN$)

3.5.1 Présentation

Dans le premier chapitre, nous avons déduit qu'aucune des approches de routage existantes ne supporte les trois exigences suivantes : *(1)* efficacité, *(2)* performance de la recherche et *(3)* coût acceptable de construction des indices de routage. En effet, nous avons montré que les méthodes de routage basées sur l'historique des requêtes sont plus avantageuses car elles ne nécessitent pas ce coût de construction. Cependant, elles ne prennent pas en considération les relations entre les requêtes passées et les pairs associés pour représenter les profils des utilisateurs. De plus, elles souffrent de problèmes de démarrage à froid et d'échec de sélection des pairs pertinents à partir de la base de connaissances locale. Nous avons traité le problème de représentation de profils et de démarrage à froid en proposant un modèle de routage qui considère les corrélations entre les requêtes et les pairs positifs, aussi nous avons proposé une méthode générique qui traite le problème de démarrage à froid.

Pour supporter les trois exigences prédéfinies, nous devons trouver une solution au problème d'échec de sélection des pairs pertinents. Dans ce contexte, nous proposons de combiner l'approche de routage basée sur l'historique des requêtes et celle basée sur le regroupement des pairs. Notre objectif est de proposer une méthode de routage hybride (*Learning Routing Method based on Clustered Network* : $LRMCN$) [Yeferny 2013b, Yeferny 2013a] qui tire partie de ces deux approches. En effet, dans LRM, chaque pair maintient une base de connaissances locale qui contient un ensemble d'intérêts de l'utilisateur déduits à partir de ses requêtes passées. Nous avons gardé le même principe de construction de la base de connaissances de notre méthode de routage par apprentissage LRM. De plus, un algorithme de regroupement est utilisé pour organiser le réseau P2P en groupes logiques de pairs qui partagent des connaissances similaires. Chaque pair p_i du réseau, établit des liens avec un ensemble de pairs amis dénoté par Fr_i ayant des bases de connaissances similaires. Par conséquent, si un pair p_i sélectionne à partir de sa base de connaissances locale un nombre insuffisant de pairs pertinents, il transmet la requête selon son contenu aux meilleurs pairs amis, plutôt

que de la transmettre à des voisins choisis au hasard. Les pairs amis choisis sont en mesure de trouver dans leurs bases de connaissances des pairs pertinents. Nous avons fondé notre idée sur l'hypothèse suivante de réseaux sociaux : *il est plus fréquent que deux de vos amis aient une forte probabilité de se connaître l'un l'autre, en raison de leur connaissance commune de vous, que deux autres personnes choisies au hasard de la population* [Yeferny 2013b].

3.5.2 Modélisation

L'idée sous-jacente de notre approche est de représenter la base de connaissances B_i d'un pair donné p_i par un ensemble de vecteurs représentatifs V_i pour décrire son contenu. Chaque vecteur $v_{i_k} \in V_i$ est un centroïde d'un cluster de requêtes passées. En effet, il représente un intérêt de l'utilisateur. Par la suite, les pairs qui partagent les mêmes intérêts (bases de connaissances) établissent des relations d'amitié entre eux. Pour illustrer cette idée, considérons la figure 3.2 qui montre un réseau P2P avant d'appliquer notre méthode de regroupement. Dans cet exemple, les pairs sont reliés de manière aléatoire et chacun d'eux a une base de connaissances décrite par deux vecteurs représentatifs. Chaque vecteur représentatif est représenté par une forme (par exemple, un triangle, carré, etc.). Après l'application de notre méthode de regroupement, chaque pair établit des connexions avec des pairs qui partagent des connaissances similaires (voir figure 3.3). De cette façon, nous construisons un réseau P2P, dans lequel les pairs partageant des connaissances similaires sont regroupés.

Avant de détailler cette méthode de regroupement, nous présentons les définitions suivantes :

Définition 7 *Ensemble de vecteurs représentatifs V_i : Chaque pair $p_i \in P$ sélectionne un ensemble de vecteurs représentatifs V_i pour décrire le contenu de sa base de connaissances B_i. Nous définissons, le centroïde d'un cluster de requêtes passées comme un vecteur représentatif v_{i_j} de la base des connaissances B_i.*

Définition 8 *$Distance(v_{i_k}, v_{j_v})$: Soient deux ensembles de vecteurs représentatifs V_i et V_j qui décrivent respectivement deux différentes bases de connaissances B_i et B_j. La métrique $Distance(v_{i_k}, v_{j_v})$ calcule la similarité entre deux vecteurs représentatifs $v_{i_k} \in V_i$ et $v_{j_v} \in V_j$. Nous utilisons la distance euclidienne pour calculer la similarité entre ces deux vecteurs. Cette distance est définie comme suit :*

$$Distance(v_{i_k}, v_{j_v}) = \sqrt{\sum (x_i - y_i)^2} \tag{3.3}$$

Avec x_i et y_i les $i^{ème}$ composants des vecteurs v_{i_k} et v_{j_v} respectivement. Il est à noter que le $i^{ème}$ composant représente le poids du $i^{ème}$ terme dans le vecteur.

Sur la base des précédentes définitions, nous présentons notre algorithme de regroupement de pairs. Il comporte trois phases *(i)* Le calcul des vecteurs représentatifs, *(ii)* La recherche des amis et *(iii)* La maintenance de l'ensemble des pairs amis.

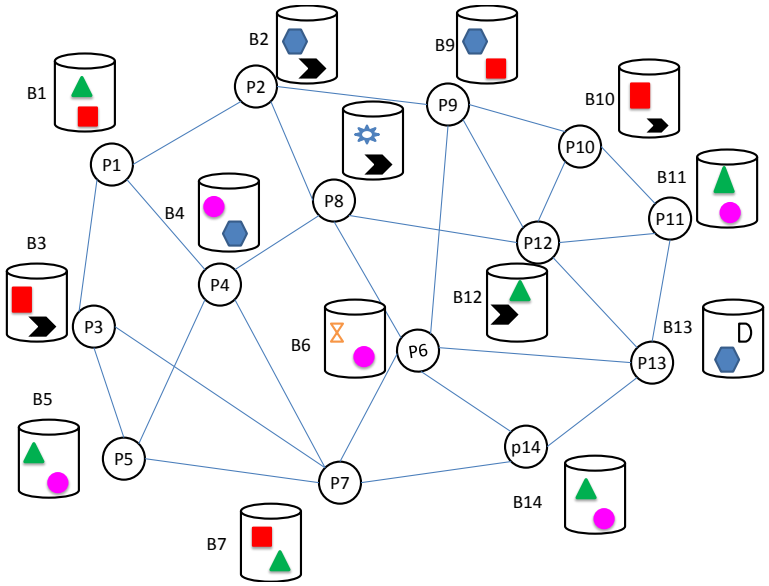


FIGURE 3.2 – Réseau P2P aléatoire (le réseau P2P avant le regroupement)

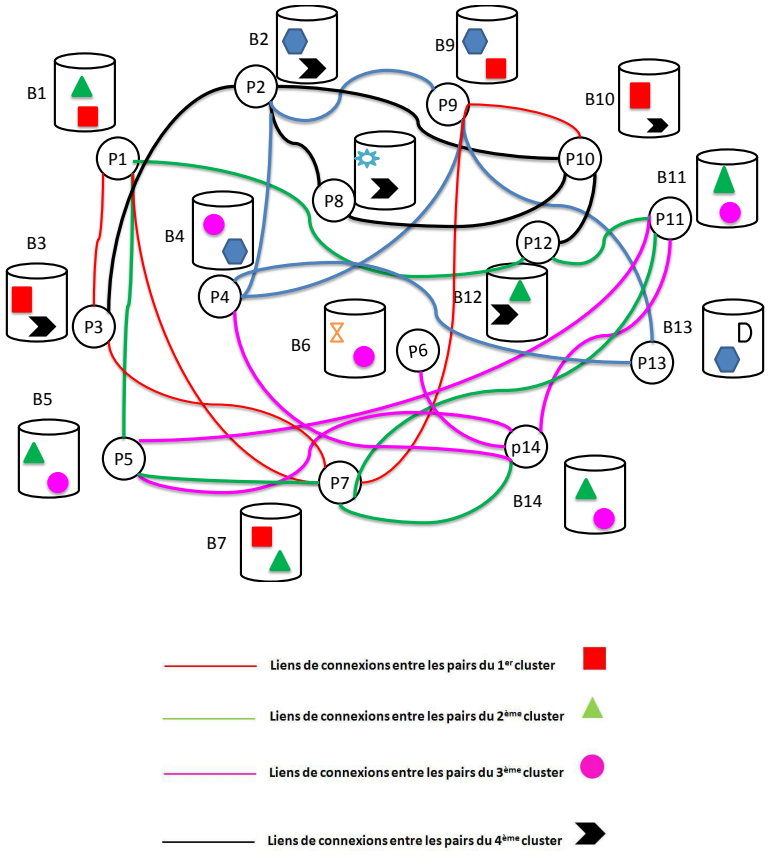


FIGURE 3.3 – Réseau P2P sémantique (le réseau P2P après le regroupement)

3.5.3 Calcul des vecteurs représentatifs

Dans notre cas, chaque vecteur représentatif $v_{i_k} \in V_i$ est un centroïde d'un cluster de requêtes passées appartenant à la base B_i. En effet, nous avons utilisé la méthode de regroupement Kmeans [Kanungo] avec comme données les requêtes passées. Chaque requête est un vecteur de termes avec les poids associés.

3.5.4 Recherche des amis

Après le calcul de l'ensemble V_i de ses vecteurs représentatifs, le pair p_i doit chercher leur pairs amis dans le réseau. Les amis de p_i forment un ensemble Fr_i. Nous définissons formellement cet ensemble comme suit :

$$\begin{aligned} Fr_i = \{p_j \in P \ \ / \exists v_{i_k} \in V_i, \ \ v_{j_v} \in V_j \\ avec \ \ i \neq j \ \ et \ \ Distance(v_{i_k}, v_{j_v}) < \theta\} \end{aligned} \tag{3.4}$$

θ est un seuil predéfini. Pour construire l'ensemble Fr_i, le pair p_i lance une requête de recherche des pairs amis $search\_Friends(V_i, TTL)$ contenant l'ensemble de vecteurs représentatifs de sa base des connaissances avec un certain (TTL). Cette requête est propagée avec la méthode classique BFS. Lorsqu'un pair p_j reçoit cette requête, il calcule la distance entre les vecteurs représentatifs de p_i et ses propres vecteurs (voir l'algorithme 5 *ComputeSimilarity*). Ensuite, p_j répond en envoyant à la fois son vecteur représentatif le plus proche, s'il existe, et la distance. Lorsque p_i reçoit les vecteurs représentatifs des pairs ayant des profils proches, il sélectionne les premiers K pairs ayant les plus basses valeurs de distance comme pairs amis.

Algorithme 5: ComputeSimilarity

Input:
V_i : vecteur représentatif de pair p_i.
V_j : vecteur représentatif de pair p_j.
θ : un seuil prédefini.

Output:
$< r, d >$: couple, avec r un vecteur représentatif et d une distance.

```
1 begin
2     < r, d >=< null, +∞ >
3     for each v1 ∈ Vi do
4         for each v2 ∈ Vj do
5             s = Distance(v1, v2)
6             if s < θ and s < d then
7                 r = v2
8                 d = s
9 end
```

3.5.5 Maintenance de l'ensemble des pairs amis

La base de connaissances de chaque pair est mise à jour périodiquement (i.e., après l'émission d'un certain nombre de requêtes, dans notre cas, après l'émission de 20 requêtes) pour prendre en compte les nouvelles requêtes. Cela induit la mise à jour des vecteurs représentatifs. Dans la pratique, nous avons représenté chaque pair par deux vecteurs (i.e., le nombre de clusters est égal à 2). Après le re-calcul des nouveaux vecteurs représentatifs, l'algorithme de recherche des amis est exécuté pour mettre à jour l'ensemble d'amis.

3.5.6 Algorithme de routage : Learning Peer Selection over Clustered Network $LPSCN$:

Pour mettre en œuvre notre méthode de routage de requêtes hybride $LRMCN$ nous avons développé l'algorithme $LPSCN$ qui utilise à la fois la base de connaissances locale d'un pair et la liste de ses pairs amis. Il exploite en premier lieu la base des connaissances locale pour sélectionner les pairs pertinents pour une requête donnée. On utilise le même algorithme de sélection de pairs par apprentissage LPS (*LearningPeerSelection*) présenté dans la section 3.3.4 (ligne 2 de l'algorithme 6). Par la suite, si le nombre de pairs pertinents est inférieur à un certain seuil P_{max}, $LPSCN$ appelle l'algorithme *addSemantic*() (ligne 5 de l'algorithme 6) pour ajouter les meilleurs pairs amis, plutôt que des pairs choisis de manière aléatoire. Nous sommes sûrs que les pairs amis sont en mesure de sélectionner des pairs pertinents à partir de leurs bases de connaissances. Par conséquent le processus de sélection est totalement sémantique, ce qui permet de palier au problème d'échec de sélection dans les méthodes basées sur l'historique de requêtes.

Algorithme 6: Learning Peer Selection over Clustered Network Algorithm

Input:
B : base de connaissances
Q : requête
F_r : liste de pairs amis
P_{max} : le nombre de pairs à sélectionner

```
1 begin
2     finalList = LearningPeerSelection(B, Q, P_max)
3     if (|finalList| < P_max) then
4         N = P_max − |finalList|
5         addSemantic(finalList, N, F_r, Q)
6     Forward(Q, finalList)
7 end
```

L'algorithme *addSemantic()*

Notre algorithme de sélection de pairs amis pertinents Algorithme (voir 7) implique trois étapes :

1. *Calcul de similarité* : dans cette étape, nous calculons la similarité entre le vecteur représentatif de chaque pair ami et la requête q (lignes 3 - 5 de l'algorithme 7) ;
2. *Trie de la liste d'amis* : cette phase ordonne la liste des pairs amis F_r selon la valeur de similarité entre le vecteur représentatif de chaque pair ami et la requête (ligne 6 de l'algorithme 7) ;
3. *Ajout des meilleurs pairs amis* : cette phase ajoute à la liste finale $finalList$ les meilleurs N pairs amis ayant les plus hautes valeur de similarité (lignes 7 - 9 de l'algorithme 7).

Algorithme 7: ADD SEMANTIC ALGORITHM

Input:
$finalList$: liste des pairs pertinents
N : nombre de pairs à ajouter
F_r : liste des pairs amis
Q : requête

```
1  begin
2      cpt = 0
3      for each friend ∈ F_r do
4          r = friend.getRepresentativeVector()
5          friend.similarity = similarity(r, Q)
6      sort(F_r)
7      while cpt < |F_r| and cpt < N do
8          finalList = finalList + F_r.getElement(cpt)
9          cpt++
10 end
```

3.6 Conclusion

Dans ce chapitre, nous avons présenté notre modèle de routage sémantique basé sur l'historique des requêtes. Ensuite, nous avons instancié le modèle proposé pour définir notre propre méthode de routage par apprentissage LRM. De plus, pour palier le problème de démarrage à froid, nous avons présenté une méthode de prédiction de l'intension de l'utilisateur pour construire une base de connaissances initiale à chaque pair. Enfin, nous avons présenté une méthode de routage hybride $LRMCN$ basée sur l'historique des requêtes et le regroupement de pairs pour traiter le problème d'échec de sélection. Dans le chapitre suivant, nous proposons une solution de routage pour les systèmes $P2P$ mobiles.

Chapitre 4

Solution proposée pour les systèmes P2P mobiles

4.1 Introduction

Dans le troisième chapitre, nous avons montré que les méthodes de routage intégrées dans les systèmes P2P mobiles sont basées sur le contexte de l'utilisateur. En effet, le choix des meilleurs voisins dépend de l'efficacité de la fonction de sélection et des informations contextuelles utilisées. Par conséquent, chaque méthode a des avantages et des inconvénients par rapport à l'autre. Dans ce cadre, nous proposons une méthode de routage pour les systèmes P2P non structurés mobiles basée sur le contexte de l'utilisateur (*Context-Aware Routing Method* : *CARM*) [Yeferny 2012b]. Les objectifs de notre méthode sont :

- Sélectionner les meilleurs voisins pour une requête donnée en se basant sur le contenu de la requête et les profils des voisins ;
- Garantir que les pairs pertinents sont atteint en prenant en considération les contraintes de MANET. Nous avons considéré la puissance d'énergie de la batterie et la mobilité des nœuds pour garantir la stabilité des liens entre l'initiateur de la requête et le pair qui possède les documents pertinents. De plus, notre méthode traite le problème de congestion en évitant de router les requêtes vers les pairs chargés ou les lignes encombrées.

Dans le reste de ce chapitre, nous présentons notre approche d'une manière générale. Ensuite, nous décrivons les informations contextuelles considérées et les différentes métriques utilisées pour sélectionner les meilleurs voisins. Les algorithmes de routage et de sélection des meilleurs voisins sont détaillés. Enfin, nous présentons un exemple illustrant notre proposition.

4.2 Méthode de routage proposée

Dans le deuxième chapitre, nous avons montré que dans les méthodes de routage intégrées les plus récentes [Hwa 2009], [Karasabun 2009] et [daH 2009], lorsqu'un pair lance une requête de recherche, il commence tout d'abord par faire une recherche dans sa collection locale. Ensuite, il la route vers les K meilleurs voisins choisis en se basant sur des informations contextuelles liées au contenu de la requête, la puissance de la batterie, la charge de réseau ou de calcul d'un pair mobile et le même processus est réitéré par chaque voisin sélectionné. Pour limiter la génération d'un nombre important de messages par ce procédé, la propagation des messages est limitée jusqu'à un

horizon, défini par un paramètre de durée de vie de la requête (TTL), décrémenté à chaque itération. Une fois qu'un pair possédant des ressources pertinentes est trouvé, il retourne une réponse à l'initiateur contenant la liste des documents pertinents. La réponses suit le chemin inverse de la requête pour atteindre l'initiateur.

L'étude que nous avons présenté dans le deuxième chapitre a permis de déduire que chaque méthode possède des avantages et des inconvénients. En effet, le choix des meilleurs voisins dépend de l'efficacité de la fonction de sélection et des paramètres considérés. Dans ce contexte, nous proposons une méthode de routage pour les systèmes P2P non structurés mobiles basée sur le contexte de l'utilisateur.

Dans ce qui suit, nous présentons les informations contextuelles considérées pour sélectionner les meilleurs voisins pour une requête donnée q. Nous présentons la manière d'acquisition de ces informations et nous définissons différente métriques utilisées dans notre méthode de sélection. Par la suite, nous présentons notre méthode de routage.

4.2.1 Modélisation d'un système P2P mobile

Formellement, un réseau MANET peut être représenté par un graphe non-orienté $G = (V, E)$ où V désigne l'ensemble des nœuds et $E \in V^2$ dénote l'ensemble des arcs correspondants aux communications directes possibles. Soit n_i et n_j deux nœuds de V, l'arc (i, j) ou bien le lien l_{ij} existe, si et seulement si, n_i peut envoyer directement un message à n_j ; on dit alors que n_j est un voisin de n_i. Les couples appartenant à E dépendent de la position des nœuds et de leur portée de communication. Si on retient l'hypothèse que tous les nœuds ont une portée R identique, et si $d(i, j)$ désigne la distance entre les nœuds n_i et n_j, alors l'ensemble E peut-être défini comme suit :

$$E = \{(i, j) \in V^2 | d(i, j) \leq R\} \tag{4.1}$$

Chaque nœud n_j peut partager un ensemble de documents $D_j = \{d_1, ..., d_k\}$, où d_i dénote un identifiant d'un document.

4.2.2 Notion de contexte

L'informatique sensible au contexte est apparue dans le milieu des années quatre-vingt dix impulsée par les travaux de [Schilit B.N. 2002]. Ce terme fait référence à des systèmes capables de percevoir un ensemble de conditions d'utilisation "le contexte" afin d'adapter en conséquence leur comportement en termes de délivrance d'informations et de services [Cheverest K. 2002]. On comprend donc qu'avec l'avènement des technologies sans fil, la sensibilité au contexte est devenue un caractère incontournable des systèmes qui permettent une utilisation de type nomade. Un utilisateur est dit nomade s'il peut se connecter au système depuis différents lieux, en utilisant un dispositif d'accès le plus souvent léger, qu'il peut changer à tout moment, parfois sans même se déconnecter. La notion de contexte est extensible à volonté. En pratique, elle n'englobe qu'un nombre limité et variable de caractéristiques. La plupart de ces

systèmes se contentent de ne traiter qu'une portion du contexte [Burrell J. 2001]. Vis-à-vis de l'acquisition, on distingue l'information contextuelle [Mostefaoui K. 2004] selon qu'elle soit être détectée, dérivée ou explicite. L'information détectée provient de capteurs physiques ou logiciels (température, niveau sonore, pression, altitude, lumière, etc.). La localisation de l'utilisateur entre dans cette catégorie. Elle peut être détectée par GPS en extérieur et par diverses techniques d'approximation en intérieur (par exemple, par croisement de signaux perçus par des bornes Wifi et d'étiquettes électroniques [Anne M. 2005].

4.2.3 Informations contextuelles utilisées

Plusieurs dimensions de contexte peuvent être exploitées. Dans notre méthode nous avons considéré la mobilité, la puissance de batterie et la charge de calcul.

A. Stabilité de lien

La stabilité de lien entre le routeur de la requête, un nœud n_i, et son voisin n_j est un facteur très important. En effet, si le lien l_{ij} entre les deux nœuds mobiles est instable, il n'est pas intéressant de router la requête vers n_j. Pour mesurer la stabilité du lien l_{ij}, nous définissons une fonction *Link_stablity* qui combine les deux facteurs : énergie de la batterie et la mobilité des deux nœuds n_i et n_j. Avant la description de cette fonction, nous présentons deux principales métriques. La première prend en considération le facteur de mobilité pour prédire la durée de vie du lien entre les deux mobiles en seconde. La deuxième calcule le temps restant de batterie d'un nœud donné.

a. Mobilité du pair : Dans l'environnement MANET, les pairs mobiles se déplacent librement et à tout moment. Par conséquent, les liens entre les pairs mobiles ont une durée de vie limitée. Dans notre approche, nous considérons la mobilité des pairs pour choisir le chemin le plus stable en prédisant la durée de vie du lien entre le routeur de la requête et ses voisins. Pour prédire la durée de vie d'un lien l_{ij} entre un pair n_i et son voisin n_j, nous allons adapter des fonctions définies dans le protocole de routage de la couche réseau RABR [Agarwal 2000]. Le protocole RABR se base sur les forces des signaux de deux pairs voisins pour déterminer la durée de vie du lien entre eux. En effet, pour calculer la durée de vie d'un lien l_{ij}, le pairs n_i calcule périodiquement à chaque $\triangle_t$ secondes la puissance de signal S_{ij} de son voisin n_j. Due à la mobilité des nœuds, la puissance du signal change dans le temps. Lorsque la force du signal S_{ij} associée à un lien l_{ij} est inférieure à un certain seuil S_t, les deux pairs n_i et n_j sont supposés déconnectés. Dans MANET, les puissances des signaux des pairs voisins ne sont pas égales. Par exemple, si un nœud n_i possède un voisin n_j qui se trouve dans la périphérie de sa zone de transmission, alors il est faiblement connecté à n_i par rapport à un autre voisin n_k qui est proche de n_i. Par conséquent, le risque que n_j sorte de la portée de transmission de n_i (en raison de la mobilité de deux nœuds n_j et n_i) est supérieure à celui du nœud n_k. Les auteurs ont développé une fonction appelée *link_ affinity* $a_{ij}(t)$ qui prédis le temps nécessaire (à partir

de l'instant t) pour que le nœud n_j sorte de la portée de transmission radio du nœud n_i. Pour ce faire, ils calculent tout d'abord le taux de changement de la puissance du signal $\Delta S_{ij}(t)$ à l'instant t comme suit :

$$\Delta S_{ij}(t) = \frac{S_{ij}(t) - S_{ij}(t - \triangle_t)}{\Delta_t} \tag{4.2}$$

Avec, $S_{ij}(t)$ la puissance du signal du nœud n_j reçu par le nœud n_i à l'instant t, $S_{ij}(t - \triangle_t)$, la valeur de la puissance de signal à l'instant $(t - \triangle_t)$ et $\triangle_t$, la période d'échantillonnage. Le temps prévu pour que le nœud n_j sorte de la portée de transmission radio du nœud n_i est éstimé par la fonction $a_{ij}(t)$ comme suit :

$$a_{ij}(t) = \begin{cases} High \quad if \quad \Delta S_{ij}(ave)(t) \geq 0 \\ \\ \frac{S_t - S_{ij}(t)}{\Delta S_{ij}(ave)(t)} \quad Sinon \end{cases} \tag{4.3}$$

Avec, $\Delta S_{ij}(ave)(t)$ la moyenne du taux de changement du signal dans les dernières périodes d'échantillonnage et S_t le seuil de puissance du signal à partir duquel les deux nœuds sont supposés être déconnectés.

Si $\Delta S_{ij}(ave)(t)$ est positif alors les deux nœuds se rapprochent. Par conséquent, $a_{ij}(t)$ est égale à $High$ à l'instant t. La valeur de $High$ est approximativement égale au temps pris par le nœud n_j pour sortir de la portée de la transmission radio du nœud n_i avec une vitesse moyenne. Elle est calculée comme suit :

$$High = \frac{la\ portee\ de\ transmission\ du\ noeud\ n_i}{vitesse\ moyenne\ du\ noeud\ voisin\ n_j} \tag{4.4}$$

Comme nous avons mentionné précédemment, si le voisin n_j se trouve dans la périphérie de la zone de transmission du nœud n_i, alors n_j est faiblement connecté à n_i par rapport à un autre voisin n_k proche de n_i. Ainsi, le risque que n_j sorte de la portée de transmission de n_i est supérieur à celui du nœud n_k. Cette hypothèse est toujours valide même si les deux nœuds se rapprochent (i.e., $\Delta S_{ij}(ave)(t) \geq 0$). De ce fait, pour prendre en compte cette hypothèse dans le calcul de la durée de vie du lien, un facteur de correction μ est utilisé pour modérer la valeur de $High$, donc, $a_{ij}(t) = \mu \times High$. Ce facteur de correction est calculé comme suit :

$$\mu = 1 - \frac{S_t}{S_{ij}(t)} \tag{4.5}$$

Ceci indique que si $S_{ij}(t)$ est très proche de S_t, μ sera proche de zéro et par conséquent $a_{ij}(t)$ tend vers zero, même si $\Delta S_{ij}(ave)(t)$ est positif.

b. Énergie de batterie :

Un voisin qui a un temps de batterie restant faible n'est pas intéressant même si la durée de vie du lien entre lui et le routeur de la requête est élevée. Pour calculer ce facteur, nous nous sommes basés sur les niveaux de batteries des voisins associés. Nous supposons que le niveau de la batterie d'un pair mobile diminue

lorsqu'il transmet, reçoit et traite des paquets. Un pair est déconnecté si sa puissance de batterie devient très faible.

Pour prédire le temps restant de batterie, nous supposons comme dans [Gupta 2011] que l'énergie nécessaire pour chaque opération de reception, transmission, diffusion d'un paquet est fixe. Chaque nœud calcule le temps restant de sa batterie comme suit : on suppose que les niveaux de batterie d'un nœud n_i aux instants t_1 et t $(t_1 < t)$ sont respectivement $Rengy1(n_i)$ et $Rengy(n_i)$. Le temps de batterie restant du nœud n_i à partir de l'instant t est calculé par la formule suivante :

$$Rtime(n_i)(t) = Rengy(n_i) \times [\frac{(t - t1)}{(rengy1(n_i) - rengy2(n_i))}] \tag{4.6}$$

Nous combinons la durée de vie du lien entre l'émetteur de la requête et son voisin et le temps restant de batterie du voisin pour définir notre fonction de stabilité du lien $Link\_stability$. La fonction $Link\_stability_{ij}(t)$ estime le temps restant, à partir de l'instant t, durant lequel un nœud n_i peut communiquer directement avec son voisin n_j. En effet, $Link\_stability_{ij}(t)$ c'est le minimum entre le temps nécessaire pour que le voisin n_j se déplace hors de la portée de transmission radio du pair n_i et le temps de batterie restant du voisin n_j. Formellement, $Link\_stability_{ij}(t)$ est calculé comme suit :

$$Link\_stability_{ij}(t) = Min(Rtime(n_j)(t), a_{ij}(t)) \tag{4.7}$$

Avec, $Rtime(n_j)(t)$, le temps de batterie restant du pair n_j à partir de l'instant t et $a_{ij}(t)$, le temps pris par le nœud n_j pour sortir de la portée de transmission radio du nœud n_i calculé selon l'equation 4.3.

B. Charge du pair

Pour une utilisation optimale des resources du réseau P2P, il est nécessaire d'équilibrer la charge de traitement entre les différents pairs en fonction de leurs capacités de calcul. Dans ces systèmes, il y a une possibilité de déséquilibre de charge due à l'uniformité de capacité de calcul des pairs mobiles. Un pair qui a une capacité de calcul élevée termine son propre travail rapidement, par conséquent, la plupart du temps il n'a pas de travaux à traiter ou bien il est moins chargé que les pairs ayant une faible capacité de calcul. Cependant, si on envoie les requêtes uniquement aux pairs ayant une capacité de calcul élevée, les paquets seront acheminés par les mêmes routes ce qui implique un temps de latence important et un problème de congestion. En effet, une file d'attente du serveur se forme car le degré élevé de variation dans les intervalles entre les paquets arrivés et dans les temps de service cause des congestions temporaires. En général, ces variations sont représentées par des distributions théoriques de probabilités. Dans les principaux modèles utilisés, on suppose que le nombre d'arrivées dans un intervalle donné suit la loi de Poisson, alors que le temps de service suit une loi exponentielle. En évitant le problème de congestion, nous pouvons garantir un débit fiable et moins de temps de latence. Un paramètre important qui indique la congestion d'une ligne est le

taux d'utilisation du voisin vers lequel on désire router la requête (i.e., ce voisin joue le rôle d'un serveur). Le taux d'utilisation $u(t)$ d'un noeud donné représente le rapport entre la demande (mesurée grâce au taux d'arrivée moyen de paquets, λ) et la capacité de service (le taux moyen de service, μ). En effet, $u(t)$ est la probabilité que le serveur de la file soit occupé.

$$u(t) = \frac{\lambda}{\mu} \tag{4.8}$$

$$\mu = \frac{1}{Temps\ moyen\ de\ service} \tag{4.9}$$

Les taux d'arrivée λ et de service μ sont exprimés en paquêts/seconde.

Pour résoudre les problèmes de congestion et de temps de latence, nous avons défini la fonction $Peer\_Load(n_j)(t)$ qui calcule la charge du voisin n_j en fonction de sa capacité de calcul *cpu* et son taux d'utilisation. La charge d'un voisin n_j à l'instant t est calculée comme suit :

$$Peer\_Load(n_j)(t) = cpu \times (1 - u(t)) \tag{4.10}$$

Avec :

- *cpu* dénote la puissance de processeur. Dans la pratique, la fréquence d'horloge est la caractéristique la plus importante pour définir la puissance d'un processeur. La règle est relativement simple : plus la fréquence est élevée, plus le processeur est rapide. Actuellement, les appareils mobiles possèdent des processeurs avec une fréquence d'horloge entre 0.5 et 3 GHZ ;
- $(1 - u(t))$ est la probabilité pour que le serveur soit libre (aucun paquet dans la file d'attente).

Une valeur élevée de la fonction $Peer\_Load(n_j)(t)$ indique que le pair est moins chargé, par contre, si la valeur est faible, elle indique que le pair est à éviter car il est trop chargé. En effet, cette fonction permet d'envoyer plus de messages aux voisins les moins chargés et peu de messages aux pairs saturés. Ainsi, l'équilibre de charge sera garantie.

C. Le profil de l'utilisateur et le contenu de la requête

Choisir des voisins moins chargés et ayant des liens stables avec l'initiateur de la requête ne suffit pas pour retrouver les ressources pertinentes. Pour améliorer l'efficacité de la recherche, nous devons choisir les voisins susceptibles de fournir des réponses adéquates à la requête. En effet, nous devons considérer le contenu de la requête et les profils des voisins, en plus de leurs charges et la stabilité des liens avec eux, afin de localiser les pairs pertinents. La représentation du profil de l'utilisateur varie selon les approches et les applications. Dans le domaine de la recherche d'information,

le profil de l'utilisateur décrit le plus souvent son centre d'intérêt et, de ce fait, est souvent confondu avec la requête de l'utilisateur. Ce profil est généralement défini à l'aide d'un vecteur de mots clés avec éventuellement un poids associé à chaque mot [Bouzeghoub 2005]. Pour intégrer le profil utilisateur dans notre méthode de routage, nous présentons tout d'abord la manière de l'acquérir, de le représenter et enfin son intégration.

a. Acquisition des profils : L'acquisition du profil peut être faite d'une manière implicite ou explicite. Dans une acquisition explicite, un utilisateur doit exprimer ses informations personnelles ou ses préférences à travers la spécification de ses centres d'intérêts, le marquage de services ou la réponse a des questions sur ses intérêts. L'inconvénient de cette méthode d'acquisition réside dans l'implication de l'utilisateur dans le processus de recommandation, ce qui lui impose une charge additionnelle à son utilisation du système [Bouzeghoub 2005]. En plus, souvent l'utilisateur, qui se trouve obliger à remplir un formulaire, ou à répondre à une question, peut introduire des informations erronées. Pour ces raisons, plusieurs travaux de recherche préfèrent utiliser une approche implicite pour extraire les profils des utilisateurs. Dans notre cas, nous avons recouru vers une acquisition implicite des intentions de l'utilisateur. En effet, nous construisons le profil de chaque voisin à partir de l'historique de requêtes et les réponses associées. Notre méthode d'acquisition de profil ne nécessite pas un coût de communication entre les pairs pour collecter les informations relatives à chaque voisin.

b. Représentation des profils : Dans la littérature, différents modèles de représentation de profils ont été présentés. Nous citons, le modèle vectoriel, graphique, sémantique, etc. [Bouzeghoub 2005]. Dans notre cas, nous avons adopté le modèle vectoriel grâce à sa simplicité de mise en œuvre. En effet, dans notre méthode, le profil de l'utilisateur est représenté par un ensemble de vecteurs de termes pondérés indépendants, pour prendre en compte des centres d'intérêt multiples. Chaque vecteur correspond à une requête passée où le nœud voisin a retourné une réponse. Nous avons utilisé une table de profils au niveau de chaque pair pour stocker les profils de tous ses voisins. Chaque entrée de la table est de la forme (pair p_j , liste de requêtes auxquelles p_j a agit positivement). En effet, chaque ligne représente le profil d'un voisin. Pour éviter d'avoir une taille importante de cette table, la stratégie Least Recently Used (LRU) est utilisée pour remplacer les anciennes requêtes par des nouvelles requêtes.

c. Intégration de profils utilisateurs : Pour déterminer le degré de pertinence d'un voisin n_j pour une requête donnée q, le routeur de la requête n_i calcule la similarité entre son voisin n_j et la requête q en utilisant la fonction de pertinence suivante :

$$Psim_{n_i}(n_j, q) = \sum_{q_k \in Q_j} Cosine(q_k, q) \tag{4.11}$$

Avec, Q_j, un ensemble de requêtes auxquelles le pair n_j a retourné des réponses, $Cosine(q_k, q)$ dénote la fonction de similarité cosinus qui calcule le produit scalaire des vecteurs associés aux requêtes q_k et q. Chaque vecteur contient une liste de termes avec leurs poids TF normalisés.

4.2.4 Algorithme de routage

l'objectif de l'algorithme de routage est de router les requêtes vers les meilleurs voisins en utilisant les différentes informations contextuelles et les métriques proposées. Lorsqu'un pair mobile reçoit une requête, l'algorithme de routage décrémente la valeur de TTL par un. La propagation de la requête s'arrête si la valeur de TTL est égale à zéro. Si la valeur de TTL est positive l'algorithme de routage (voir Algorithme 8) utilise la fonction $getPertinentPeers()$ pour retourner les K meilleurs voisins vers lesquels la requête est routée (voir ligne 9 de l'algorithme 8), avec K un certain seuil predéfini.

Algorithme 8: ALGORITHME DE ROUTAGE

```
1  Input :
2      q : Une requête
3      N : Liste de voisins
4      K : Le nombre de pairs à sélectionner
5      TTL : Profondeur de la recherche
6  begin
7      TTL = TTL − 1
8      if (TTL > 0) then
9          voisinsChoisis = getPertinentPeers(q, N, k)
10         propager(voisinsChoisis)
11 end
```

La fonction $getPertinentPeers()$ est détaillée dans la section suivante. Elle se base sur les métriques déjà définies.

4.2.5 Algorithme de sélection des meilleurs voisins

L'algorithme de sélection des meilleurs voisins (voir Algorithme 9)) implique trois étapes. En premier lieu, nous calculons le score de pertinence de chaque voisin à la requête q (lignes 2 - 4 de l'algorithme 9). Pour calculer ce score, nous utilisons notre fonction de calcul de pertinence 4.12. Ensuite, nous ordonnons les voisins selon leurs scores de pertinence à la requête (ligne 5 de l'algorithme 9). Enfin, nous sélectionnons les K meilleurs voisins ayant les plus hautes valeurs de pertinence (ligne 6 de l'algorithme 9).

Algorithme 9: getPertinentPeers

Input:
q : requête.
N : liste de voisins.
K : le nombre de pairs à sélectionner.
t : le temps courant.

```
1 begin
2     for each v_i ∈ N do
3         pertinence = Pertinence(v_i, q)(t)
4         liste.add(< v_i, pertinence >)
5     liste.sort()
6     return(liste, k)
7 end
```

Fonction de pertinence

Notre fonction de pertinence calcule le score de pertinence d'un voisin n_j à une requête q à l'instant t comme suit :

$$Pertinence(n_j, q, t) = \begin{cases} MaxTime\times & [PL \times Peer\_Load(n_j)(t) + PS \times Psim(n_j, q)] \\ & Si\ Link\_stability(ij)(t) > MaxTime \\ Link\_stability(ij)(t)\times & [PL \times Peer\_Load(n_j)(t) + \\ PS \times Psim(n_j, q)] & Sinon \end{cases} \tag{4.12}$$

Avec, $Link\_stability(ij)(t)$, $Peer\_Load(n_j)(t)$, $Psim(n_j, q)$ dénotent respectivement les fonctions de stabilité du lien l_{ij} (4.7), la charge du voisin n_j (4.10) et la similarité entre le voisin n_j et la requête q (4.11). PL et PS sont des constantes respectivement de charge et similarité d'un pair. Ces constantes sont utilisées pour ajuster la valeur de la fonction de pertinence. $MaxTime$ est une constante déterminée expérimentalement qui dénote le maximum de temps d'attente d'une réponse d'un voisin.

Notre fonction de pertinence donne plus d'importance au facteur $Link\_stability$. En effet, lorsque la durée de vie du lien entre l'émetteur de la requête et son voisin est très faible, la valeur de la fonction de pertinence est également faible, même si le voisin est moins chargé et susceptible de fournir des documents pertinents à la requête. Par conséquent, nous favorisons les liens stables, car il n'y a aucune garantie que les réponses, qui vont suivre le chemin inverse de la requête, atteignent l'émetteur si on choisit un lien instable. En outre, si plusieurs pairs ont des liens stables avec l'émetteur de la requête, la fonction de pertinence donne plus d'importance aux pairs moins chargés et susceptibles de fournir des documents pertinents à la requête.

	n_2	n_3	n_4
t_1	1	20	30
t_2	5	10	9
t_3	3	15	20
t_4	12	4	3
t_5	5	20	25
t_6	14	15	20
t_7	12	18	25
t_8	10	5	7
t_9	14	25	15
t_{10}	10	26	10

TABLE 4.1 – Variation de la puissance de signal de chaque voisin

Pair	Liste de requêtes
n_2	$q_1 = \{w1, w2, w3\}, q_2 = \{w2, w4, w5\}, q_3 = \{w2, w7, w6\}$
n_3	$q_3 = \{w2, w7, w6\}, q_4 = \{w1, w7, w8\}, q_5 = \{w9, w10\}, q_6 = \{w9, w10\}$
n_4	$q_1 = \{w1, w2, w3\}, q_5 = \{w9, w10\}, q_4 = \{w1, w7, w8\}$

TABLE 4.2 – Table de profils

4.3 Exemple d'exécution de l'algorithme de routage

Dans cette section, nous illustrons un exemple d'application de notre algorithme de routage. Dans ce qui suit, nous présentons tout d'abort quelques hypothèses puis nous détaillons les différentes étapes d'exécution.

4.3.1 Hypothèses

Nous présentons les différentes hypothèses que nous avons considéré dans ce travail.

- Le pair n_1 désire propager à l'instant t_{10} la requête $q = \{w2, w4, w6\}$, où chaque $wi \in q$ est un terme de la requête q ;
- La liste $N = \{n_2, n_3, n_4\}$ contient les voisins physique du nœud n_1 à l'instant t_{10} ;
- Le tableau 4.1 présente les variations des signaux de chaque voisin de l'instant t_1 à t_{10} ;
- Le pair n_1 maintient la table de profils (voir Table 4.2) ;
- La portée de transmission des différents nœuds est $R = 20m$;
- Le tableau 4.3 présente la vitesse moyenne, le temps de batterie restant et la charge de chaque voisin à l'instant t_{10} ;
- Le nombre de voisins à sélectionner est $K = 1$;
- Le seuil du puissance de signal à partir duquel les nœuds sont supposés être déconnectés est $S_t = 2$;
- Les valeurs des constantes PL et PS sont fixées à 1.

Pair	Vitesse moyenne ($m.s^{-1}$)	Temps de batterie restant (s)	Charge du pair
n_2	10	10	3
n_3	15	25	1
n_4	10	100	0.3

TABLE 4.3 – La vitesse moyenne, le temps de batterie restant et la charge de chaque voisin à l'instant t_{10}

4.3.2 Exécution

Pour propager la requête q, l'algorithme de routage fait appel à la fonction $getPertinentPeers()$ de l'algorithme 9) qui sélectionne le meilleur voisin (i.e., par hypothèse $k = 1$) parmi n_2, n_3 et n_4. La fonction $getPertinentPeers()$ retournera le voisin qui possède le plus haut score selon la fonction de pertinence 4.12. Dans ce qui suit, nous illustrons les étapes de calcul du score de pertinence de chaque voisin.

Calcul de la stabilité des liens

Le calcul de la stabilité des liens comprend deux facteurs : la mobilité et la charge de la batterie. En premier lieu nous commençons par le calcul de la durée de vie des liens "*Link_ affinity*" avec les pairs voisins. En second lieu, nous calculons le temps de batterie restant de chaque voisin. Enfin, nous déterminons la stabilité de chaque lien "*Link_ stability*".

Pour calculer la durée de vie du lien l_{12} entre les pairs n_1 et n_2, nous commençons par le calcul du taux de changement de signal du pair n_2. Ce taux est calculé en se basant sur la formule (4.3).

Instant	*Taux de changement de signal*
t_{10}	$\Delta S_{12}(t) = \frac{S_{12}(t_{10}) - S_{12}(t_9)}{1} = \frac{10-14}{1} = -4$
t_9	$\Delta S_{12}(t) = \frac{S_{12}(t_9) - S_{12}(t_8)}{1} = \frac{14-10}{1} = 4$
t_8	$\Delta S_{12}(t) = \frac{S_{12}(t_8) - S_{12}(t_7}{1} = \frac{10-12}{1} = -2$
t_7	$\Delta S_{12}(t) = \frac{S_{12}(t_7) - S_{12}(t_6)}{1} = \frac{12-14}{1} = -2$
t_6	$\Delta S_{12}(t) = \frac{S_{12}(t_6) - S_{12}(t_5)}{1} = \frac{14-5}{1} = 9$
t_5	$\Delta S_{12}(t) = \frac{S_{12}(t_5) - S_{12}(t_4)}{1} = \frac{5-12}{1} = -7$
t_4	$\Delta S_{12}(t) = \frac{S_{12}(t_4) - S_{12}(t_3)}{1} = \frac{12-3}{1} = 9$
t_3	$\Delta S_{12}(t) = \frac{S_{12}(t_3) - S_{12}(t_2)}{1} = \frac{3-5}{1} = -2$
t_2	$\Delta S_{12}(t) = \frac{S_{12}(t_2) - S_{12}(t_1)}{1} = \frac{5-1}{1} = 4$

Le taux moyen de variation du signal $\Delta S_{12}(ave)(t_{10})$ du pair n_1 est calculé comme suit :

$$\Delta S_{12}(ave)(t_{10}) = \frac{-4+4-2-2+9-7+9-2+4}{9} = \frac{9}{9} = 1$$

Puisque $\Delta S_{12}(ave)(t_{10})$ est positif, alors $a_{12}(t_{10}) = High$.
Avec, $High = \frac{la\ portée\ de\ transmission\ du\ noeud\ n_2}{vitesse\ moyenne\ du\ noeud\ voisin} = \frac{20}{10} = 2s$

La valeur de $High$ est modérée par un facteur de correction μ qui est calculé comme suit : $\mu = 1 - \frac{S_t}{S_{12}(t_{10})} = 1 - \frac{2}{10} = 0.8$, donc
$High = 0.8 \times 2 = 1.6s$ donc, le nœud n_2 peut quitter la portée de transmission radio du nœud n_1 après $a_{12}(t_{10}) = 1.6s$
De même, après le calcul des taux de changement des signaux des pairs n_3 et n_4, nous obtenons les résultats suivants :
$\Delta S_{13}(ave)(t_{10}) = 0.66$, donc $a_{13}(t_{10}) = \mu \times High = (1 - \frac{2}{26}) \times \frac{20}{15} = 1.23s$
$\Delta S_{14}(ave)(t_{10}) = -2.2$, donc $a_{13}(t_{10}) = \frac{S_t - S_{14}(t_{10})}{\Delta S_{14}(ave)(t_{10})} = \frac{2-10}{-2.2} = 3.6s$

Enfin, la stabilité de chaque lien à l'instant t_{10} est calculée comme suit :

$$Link\_stability_{12}(t_{10}) = Min(Rtime(n_2)(t_{10}), a_{12}(t_{10})) = Min(10, 1.6) = 1.6s$$
$$Link\_stability_{13}(t_{10}) = Min(Rtime(n_3)(t_{10}), a_{13}(t_{10})) = Min(25, 1.23) = 1.23s$$
$$Link\_stability_{14}(t_{10}) = Min(Rtime(n_4)(t_{10}), a_{14}(t_{10})) = Min(100, 3.6) = 3.6s$$

Calcul de similarité *voisin-requête*

En se basant sur la fonction $Psim$ (4.11) nous calculons la similarité entre la requête et chaque voisin comme suit :

$$\begin{aligned}
Psim_{n_1}(n_2, q) &= Cosine(q_1, q) + Cosine(q_2, q) + Cosine(q_3, q) \\
&= 1/9 + (1/9 + 1/9) + (1/9 + 1/9) = 5/9 \\
Psim_{n_1}(n_3, q) &= Cosine(q_3, q) + Cosine(q_4, q) + Cosine(q_5, q) + Cosine(q_6, q) \\
&= (1/9 + 1/9) + 0 + 0 + 0 + 0 = 2/9 \\
Psim_{n_1}(n_4, q) &= Cosine(q_1, q) + Cosine(q_5, q) + Cosine(q_4, q) \\
&= 1/9 + 0 + 0 = 1/9
\end{aligned}$$

Calcul du score de pertinence de chaque voisin

En utilisant la formule $Pertinence$ (4.12), nous obtenons les valeurs de pertinence suivantes :

$$\begin{aligned}
Pertinence(n_2, q, t_{10}) &= Link_stability(12)(t_{10}) \times [Peer_Load(n_2)(t_{10}) + Psim(n_2, q)] \\
&= 1.6 \times [3 + 5/9] = 6.58 \\
Pertinence(n_3, q, t_{10}) &= Link_stability(13)(t_{10}) \times [Peer_Load(n_3)(t_{10}) + Psim(n_3, q)] \\
&= 1.23 \times [1 + 2/9] = 1.5 \\
Pertinence(n_4, q, t_{10}) &= Link_stability(14)(t_{10}) \times [Peer_Load(n_4)(t_{10}) + Psim(n_4, q)] \\
&= 3.6 \times [0.3 + 1/9] = 1.48
\end{aligned}$$

D'après les scores obtenus, la requête sera routée au voisin n_2.

4.4 Apports de notre approche

Comme nous avons mentionné précédemment, l'efficacité et la performance de la recherche d'une méthode de routage dépendent de l'efficacité de la fonction de sélection des meilleurs voisins et des informations contextuelles considérées. Le tableau 4.4

	Mobilité	Charge de pair mobile	Énergie de batterie	Contenu de la requête
ORION	non	non	non	non
EAODV	non	non	non	non
[Shah 2009]	non	non	non	non
Gossiping-LB	non	oui	non	non
P2PSI	non	non	non	oui
WI-Share	non	oui	oui	non
Notre méthode	oui	oui	oui	oui

TABLE 4.4 – Comparaison de notre méthode de routage pour les systèmes P2P mobiles avec les méthodes existantes

compare notre méthode avec les méthodes existantes dans la literature selon différentes informations contextuelles, que nous avons défini dans le deuxième chapitre.

4.5 Conclusion

Dans ce chapitre, nous avons présenté notre méthode de routage de requête intégrée pour les systèmes P2P mobiles de partage de fichiers. La méthode proposée prend en considération des informations contextuelles liées à l'utilisateur et au dispositif de communication utilisé, comme le profil de l'utilisateur qui est déduit à partir de ses requêtes passées, la mobilité, l'énergie de batterie et la charge de traitement de dispositif mobile. Le chapitre suivant est consacré à l'étude expérimentale que nous avons menée pour évaluer les performances de nos solutions de routage pour les systèmes P2P.

Chapitre 5

Validation et expérimentation

5.1 Introduction

Ce chapitre a pour objectif la mise en œuvre et l'évaluation de nos méthodes de routage des requêtes dans les systèmes P2P sur Internet et mobiles. Pour valider nos solutions de routage de requêtes dans les systèmes P2P sur Internet, nous avons implémenté les différentes méthodes proposées sur le simulateur de systèmes P2P à large échelle PeerSim [Jelasity 2010]. En outre, nous avons utilisé le simulateur des systèmes P2P mobiles NS2 [Issariyakul 2008] afin d'évaluer notre méthode de routage de requêtes dans les systèmes P2P mobiles. Ainsi, une démarche de validation par simulation a été réalisée sur la base de quatre collections de tests standards, les résultats d'expérimentation sont comparés à ceux des approches existantes, sur le plan de l'efficacité et de la performance.

Dans ce chapitre, nous présentons en détail les collections de test et les mesures d'évaluation que nous avons utilisées pour évaluer nos méthodes. Ensuite, nous décrivons l'environnement de simulation des systèmes P2P sur Internet et mobiles. Une étude expérimentale pour chaque type de systèmes est détaillée. Dans chaque étude expérimentale différents scénarios de tests seront discutés. Nous présentons pour chaque étude les résultats expérimentaux associés à chaque scénario de test.

5.2 Métriques d'évaluation

Le principal défi pour une méthode de routage dans un système de partage de fichier en P2P, est de sélectionner pour une requête donnée les "meilleurs" pairs pouvant la satisfaire pour obtenir des résultats similaires à une recherche centralisée. L'évaluation d'une telle méthode peut porter sur son efficacité et sa performance. Le premier critère ou efficacité est généralement estimé en fonction des résultats pertinents retournés par rapport aux besoins de l'utilisateur, tandis que la performance se rapporte à un accès rapide incluant le temps de calcul et surtout le temps de réponse, les coûts des mises à jour, le nombre de messages à transférer, etc. Pour cela, plusieurs mesures d'efficacité et de performance ont été proposées dans la littérature.

5.2.1 Mesures d'efficacité

Les mesures les plus importantes en Recherche d'Information (RI) sont la précision et le rappel. Rappelons que, pour une requête donnée :

- La *précision*, notée P, reflète la capacité du système à ne retourner que les documents pertinents, i.e ;

$$P = \frac{|Pert \cap Ret|}{|Ret|} \quad avec \quad |Ret| \neq 0$$

- Le *rappel*, noté R, reflète la capacité du système à retourner tous les documents pertinents, i.e ;

$$R = \frac{|Pert \cap Ret|}{|Pert|} \quad avec \quad |Pert| \neq 0$$

Où :

- $Pert$ dénote l'ensemble des documents pertinents contenus dans le corpus du système pour une requête donnée.

- Ret est l'ensemble des documents que le système a retourné.

Le rappel dépend de la fonction d'appariement "document-requête" utilisée au niveau de chaque pair pour sélectionner les documents pertinents pour une requête donnée. En plus, il dépend de la méthode de routage utilisée. En effet, si une méthode de routage route une requête vers tous les pairs pertinents mais la fonction d'appariement sélectionne des documents non pertinents pour cette requête, cela influe négativement le résultat du rappel malgré l'efficacité de la méthode de routage. Pour éviter ce problème, dans les tests que nous avons effectués, nous avons supposé que lorsqu'un pair reçoit une requête donnée, ce dernier retourne tous les documents pertinents pour cette requête. En effet, de cette façon nous évitons l'influence de la fonction d'appariement sur le résultat du rappel.

En outre, avec cette démarche la précision sera toujours égale à 1 car le système ne retourne que les documents pertinents. De ce fait, nous avons eu recours à l'utilisation des mesures relatives, qui évaluent une "efficacité relative" du système. Plusieurs variantes de ces mesures existent, nous citons à titre d'exemple la précision à une position k, la précision moyenne (non interpolée), etc. Nous donnons un intérêt spécial à la mesure de précision à une position k, qui permet d'évaluer d'une manière quantitative nos méthodes. La précision pour les k premières positions $P@k$ est formulée comme suit :

$$P@k = \frac{|Pert \cap Ret|@k}{|Ret|} \quad avec \quad |Ret| \neq 0$$

k est un rang donné de la liste retournée. Dans les tests que nous avons effectué K est fixé à 3.

Dans les systèmes P2P mobiles les valeurs de rappel et de précision sont très faibles à cause des contraintes du MANET. En effet, le plus important dans ces systèmes est

de retrouver au moins une réponse à chaque requête. De ce fait, pour évaluer notre méthode de routage dans les systèmes P2P mobiles, nous avons utilisé en plus du rappel, le taux de succès moyen. Ce taux est le ratio entre les requêtes résolues (Il y a au moins une réponse à cette requête reçue par le pair initiateur) et le nombre total de requêtes envoyées.

En général, lorsqu'on effectue des tests, plusieurs requêtes sont disponibles, et dans ce cas, c'est la moyenne des valeurs de précision, de taux de succès et du rappel pour l'ensemble des requêtes qui est calculée.

5.2.2 Mesure des performance

Les mesures de performance sont généralement liées aux caractéristiques physiques du système, telles que, la largeur de la bande passante, le volume de données transférées, la charge du traitement, etc. Pour tester les performances de nos méthodes de routage dans les systèmes P2P sur Internet, nous avons utilisé les métriques suivantes :

- Le nombre de messages (NM) : est le nombre de messages requis pour localiser les pairs pertinents d'une requête donnée. Un nombre élevé de messages échangés implique une utilisation importante de la bande passante ;
- Le nombre de pairs visités (VP) : est le nombre de pairs visités lors de la propagation d'une requête donnée, se sont les pairs impliqués dans le processus de routage. Un nombre élevé de pairs visités implique probablement la présence de pairs non pertinents donc une charge inutile est associée à ces pairs.

Généralement, une méthode de routage améliore la performance, si elle arrive à réduire le nombre de messages échangés et le nombre de pairs visités.

En outre, pour évaluer notre méthode de routage de requêtes dans les systèmes P2P mobiles, nous avons défini la mesure de performance "*Temps moyen de réponse* TMR". En effet, pour une requête donnée le temps moyen de réponse TMR est calculé comme suit :

$$TMR = \frac{(TRP - TE) + (TRD - TE)}{2}$$

TRP dénote le temps de reception de la première réponse, TRD, le temps de reception de la dernière réponse et TE indique le temps d'envoie de la requête.

5.3 Collections de test

En général, un système de recherche d'information fournit une liste ordonnée de documents selon l'estimation du degré de pertinence de ces documents par rapport à la requête. Le but de tout système est de retourner des résultats qui satisfont l'utilisateur. Dans le but d'évaluer différents systèmes, des collections de tests ont été créées. Ces collections, doivent contenir un ensemble de documents, un ensemble de requêtes et une liste de jugements de pertinence associée à chaque requête. Il existe plusieurs

collections de tests pour les systèmes de recherche d'information centralisés, les plus répandues sont TREC [TREC 2010], NTCIR [NTCIR 2010] mais malheureusement, peu de collections de tests sont dédiées aux systèmes de recherche d'information en P2P. C'est la raison pour laquelle, plusieurs travaux se sont orientés vers l'étude de la distribution d'une collection centralisée pour construire plusieurs collections de test distribuées [Zammali 2010]. Ceci consiste, à placer dans les différents pairs, des sous-collections, disjointes ou non, à partir de la collection centralisée. En outre, les requêtes doivent être aussi distribuées sur l'ensemble des pairs constituant le réseau P2P.

Pour tester nos méthodes de routage, nous avons utilisé les collections de test centralisées *BigDataSet* et *SmallDataSet* développées dans le cadre du projet RARE [RARE 2010]. Ces jeux de données ont été obtenus à partir d'une analyse statistique sur des données collectées d'un système pair-à-pair Gnutella [Goh 2005] et des données de la collection TREC [TREC 2010], ce qui nous permet de réaliser des simulations en conditions réelles. La collection *BigDataSet* est composée de 25000 documents et de 5000 requêtes. Cependant, la collection *SmallDataSet* est composée de 1700 documents et de 700 requêtes. Pour distribuer ces collections centralisées, nous avons utilisé le Framework de distribution proposé par [Zammali 2010]. Ce Framework prend en entrée deux fichiers XML décrivant la collection centralisée :

- Le fichier *document_ definition.xml* décrit l'ensemble des documents. Chaque document est défini par un identifiant et une liste de mots clés ;
- Le fichier *query_ definition.xml* décrit l'ensemble des requêtes. Chaque requête est définie par un identifiant, une liste de mots clés et une liste de documents pertinents avec pour chacun une mesure de similarité.

Il fournit deux fichiers XML, qui décrivent le placement des requêtes et des documents sur le réseau des pairs :

- Le fichier *document_ distribution.xml* décrit la distribution des documents sur le réseau des pairs. En effet, à chaque pair est associé un ensemble de documents ;
- Le fichier *query_ distribution.xml* décrit la distribution des requêtes sur le réseau des pairs. En effet, à chaque pair est associé un ensemble de requêtes.

En outre, ce framework est un outil configurable qui offre deux modèles de distribution de documents et des requêtes : un modèle statistique et un autre sémantique. Plusieurs méthodes de réplication de requêtes et de documents peuvent être appliquées pour répliquer un certain nombre de documents ou de requêtes afin de se situer dans des conditions réelles de test. Nous présentons, dans ce qui suit, les modèles que nous avons appliqués sur nos collections centralisées, ainsi que, les méthodes de réplication utilisées afin de simuler les scénarios réels et possibles.

5.3.1 Modèle statistique

Le modèle statistique consiste à appliquer une loi de distribution pour distribuer la collection centralisée sur l'ensemble des pairs. Il existe plusieurs distributions statistiques possibles. Nous avons choisi une distribution aléatoire et une distribution uniforme.

- Distribution uniforme : La distribution uniforme, comme son nom l'indique, distribue de façon équitable les documents et les requêtes de la collection centralisée sur le nombre de pairs constituant le système P2P. La collection distribuée obtenu est appelée *"Uniform Benchmark" (UB)* ;
- Distribution aléatoire : Une distribution aléatoire quant à elle, exploite le pur hasard pour la distribution de documents. Aucune corrélation n'existe entre les documents d'un pair. En effet, le nombre de documents et de requêtes affecté à chaque pair se fait de façon purement aléatoire. La collection distribuée obtenu est appelée *"Random Benchmark" (RB).*

5.3.2 Modèle sémantique

Les modèles statistiques permettent d'avoir des sous-collections qui regroupent des thèmes hétérogènes. De même, la nature des documents dans une collection n'est pas homogène, ce qui contribue à créer des collections non spécialisées relativement à un thème donné. Dans la réalité, les utilisateurs partagent des documents homogènes (i.e. qui portent sur un thème spécifique)et lancent des requêtes similaires ou proches. Pour prendre en compte cette hypothèse nous avons utilisé une distribution sémantique regroupant les documents similaires (i.e., en se basant sur les termes communs) dans une sous-collection affectée à un pair. De même, les requêtes similaires sont regroupées dans un même cluster puis elles sont affectées à un pair donnée. La collection obtenue est appelée *"Clustering Benchmark" (CB)*

5.3.3 Réplication des données

La réplication des documents dans une collection de test distribuée consiste à dupliquer un certain nombre de documents ou de requêtes. De ce fait, un document ou une requête peuvent être affectés à plusieurs pairs, ce qui contribue à la formation de collections non disjointes. Les réplications des documents et des requêtes peuvent être faites de différentes manières. Il existe plusieurs techniques de réplication, nous devons en choisir une ainsi le nombre de duplications de ressources (documents ou requêtes) [Zammali 2010]. Une méthode de réplication peut être aléatoire, elle choisit aléatoirement les ressources à répliquer, ou uniforme, dans ce cas elle duplique toutes les ressources. Une autre méthode de réplication plus proche de la réalité, duplique uniquement les requêtes les plus populaires, ou les ressources les plus demandées. Dans notre cas, nous avons choisi cette méthode de réplication pour effectuer des tests proches du cas réel.

5.3.4 Caractéristiques des collections de test distribuées utilisées

Nous avons préparé quatre collections de tests distribuées pour évaluer nos méthodes de routage. Les trois premières collections UB, RB et CB sont utilisées pour comparer les méthodes de routage dans les P2P sur Internet. Ces collections sont issues de l'application de trois modèles de distribution et réplication sur la collection centralisée $BigDataSet$. La quatrième collection $RBSmall$ est obtenue en appliquant

une distribution aléatoire sur la collection centralisée *SmallDataSet*. Nous avons choisi cette collection pour simuler les systèmes P2P mobiles. En effet, dans cette collection, le nombre de requêtes et de documents est moins élevé par rapport à la collection *BigDataset*, ce qui permet de simuler de petits réseaux comme MANET. Dans notre cas, nous avons varié le nombre de pairs de 25 à 100. Le tableau 5.1 résume les caractéristiques des différentes collections de test utilisées.

TABLE 5.1 – Caractéristiques de différentes collections de test utilisées

	UB	*RB*	*CB*	*RBSmall*
Taux de réplication de requêtes	8	8	8	1
Taux de réplication de documents	1	1	1	1
Nombre de pairs	810	810	810	de 25 à 100

5.4 Environnement de simulation

Les caractéristiques des systèmes P2P (nombre important de pairs, dynamicité du reseau, etc.) empêchent les concepteurs d'application P2P d'évaluer leurs applications avant de les déployer. Une solution à ce problème consiste à utiliser un simulateur afin de simuler et de tester ces applications avant de les déployer. Dans ce contexte, plusieurs simulateurs ont été développés avec plus ou moins de succès et selon les besoins des chercheurs. Dans notre cas, nous avons utilisé le simulateur de systèmes P2P à large échelle PeerSim [Jelasity 2010] pour tester les méthodes de routage des requêtes dans les systèmes P2P sur Internet. En outre, nous utilisons le simulateur des systèmes P2P mobiles NS2 [Issariyakul 2008] afin d'évaluer notre de méthode de routage intégrée pour les systèmes P2P mobiles.

5.4.1 Le simulateur PeerSim

Le simulateur PeerSim [Jelasity 2010] est un outil Open source écrit en Java, qui présente l'avantage d'être déjà spécialisé pour l'étude des systèmes P2P et dispose d'une architecture ouverte et modulaire dont l'adaptation et l'intégration de nouvelles couches est simple.

5.4.2 Intégration de nos solutions

Comme nous l'avons déjà mentionné, le simulateur PeerSim est un environnement de simulation pour les réseaux P2P qui n'est pas adapté à un système de recherche d'information P2P. Pour cela et afin de tester nos méthodes de routage, nous avons utilisé le simulateur générique pour les systèmes de recherche d'information en pair-à-pair [RARE 2010]. Ce simulateur a été réalisé dans le cadre du projet de recherche RARE (Routage optimisé par Apprentissage de REquêtes) [RARE 2010] au sein de TélécomSudParis. Il a pour objectif de permettre à des concepteurs de systèmes de recherche d'information P2P de pouvoir facilement simuler leurs algorithmes avant

de les implanter grâce à son architecture et à ces fonctionnalités dédiées à la recherche d'information. Ce simulateur est construit au dessus de PeerSim et peut être vu comme une spécialisation de PeerSim à ce domaine. L'utilisation de ce simulateur nous a permis de développer plus facilement les différentes méthodes testés.

5.4.3 Le simulateur NS2

Pour tester notre méthode de routage intégrée dans les systèmes P2P mobiles, nous avons utilisé le simulateur NS2. NS2 est un simulateur à événements discrets destiné à la recherche développé en C++. Aujourd'hui, il est le simulateur de réseau le plus utilisé et est devenu avec le temps une référence dans ce domaine. Son avantage réside aussi dans le fait qu'il soit multi-plateforme (UNIX et Windows, avec l'émulateur Cygwin14) et que son utilisation soit gratuite. NS2 intègre un grand nombre de fonctionnalités pour l'étude des algorithmes de routage multipoint ou unipoint, des protocoles de transport, de session, d'application (HTTP par exemple). Il gère aussi très bien la couche physique (couche 1) du modèle OSI avec différents systèmes de transmission, filaires ou non. En outre, il offre la possibilité de configurer plusieurs paramètres de simulation comme la vistesse de déplacement des nœuds, le modèle de déplacement, le niveau de batterie de chaque nœud, etc. Il reste cependant uniquement adapté aux petits réseaux.

Pour tester notre méthode de routage, nous avons implémenté une application P2P de partage de fichier, qui utilise notre méthode de routage $CARM$, sur NS2.

5.5 Etude expérimentale des solutions proposées aux systèmes P2P sur Internet

5.5.1 Scénarii de simulation

Dans les tests que nous effectuons, nous cherchons à valider les défis annoncés dans cette thèse, à savoir :

1. Exploitation des corrélations entre les requêtes passées et les pairs positifs pour représenter le profil utilisateur, au lieu d'une représentation simple du profil basée sur des statistiques concernant les requêtes passées. Pour valider cette proposition, nous avons comparé l'efficacité et la performance de la méthode LRM par rapport à la méthode "Intelligent Search (IS)". La méthode IS représente le profil utilisateur par les termes des requêtes passées sans prendre en compte les relations entre eux ;
2. Démarrage à chaud de la méthode d'apprentissage pour obtenir de meilleurs résultats durant la phase d'apprentissage au lieu de démarrer à froid avec une base de connaissances vide. Pour valider cette proposition, nous avons comparé l'efficacité et la performance de notre méthode LRM en utilisant une base de connaissances initiale lors du démarrage et en démarrant à froid avec une base de connaissances vide. Dans ce dernier cas, LRM est dénotée par $LRMFroid$;

3. Éviter le problème d'échec de sélection pour maintenir la qualité du routage même en cas d'échec de sélection des pairs pertinents à partir de la base de connaissances locale. Pour valider cette proposition, nous avons testé l'efficacité et la performance de la méthode $LRMCN$ par rapport à LRM. Cette dernière ne traite pas le problème d'échec de sélection.

Plusieurs paramètres peuvent être pris en considération pour simuler les différentes méthodes de routage. Dans notre cas, nous avons simulé la distribution des données et des requêtes. Plusieurs modèles doivent être sélectionnés et appliqués sur notre collection de test centralisée. De plus, la simulation doit prévoir l'arrivée de nouvelles requêtes et le changement du besoin de l'utilisateur, d'où les mises à jours des bases des connaissances de nos méthodes de routage. En outre, nous devons simuler le passage à l'échelle des méthodes proposées. Ainsi, plusieurs scénarii de test peuvent être réalisés afin de détecter l'impact de ces différents paramètres. Nous citons respectivement :

S1. l'impact du démarrage à chaud sur l'efficacité et la performance de la méthode de routage proposée ;

S2. l'impact des mises à jour des connaissances sur l'efficacité et la performance des méthodes de routage par apprentissage ;

S3. l'impact de l'exploitation des corrélations entre les requêtes passées et les pairs positifs dans la représentation du profil utilisateur sur l'efficacité et la performance d'une méthode de routage par apprentissage ;

S4. l'impact de l'échec de sélection sur l'efficacité et la performance de la méthode de routage proposée ;

S5. l'impact du modèle de distribution de jeu de données sur l'efficacité et la performance des méthodes de routage proposées ;

S6. l'impact de changement du nombre de pairs dans le réseau sur l'efficacité et la performance des méthodes de routage proposées.

5.5.2 Configuration et simulation

La simulation des cinq premiers scénarii s'est basée sur les paramètres suivants :

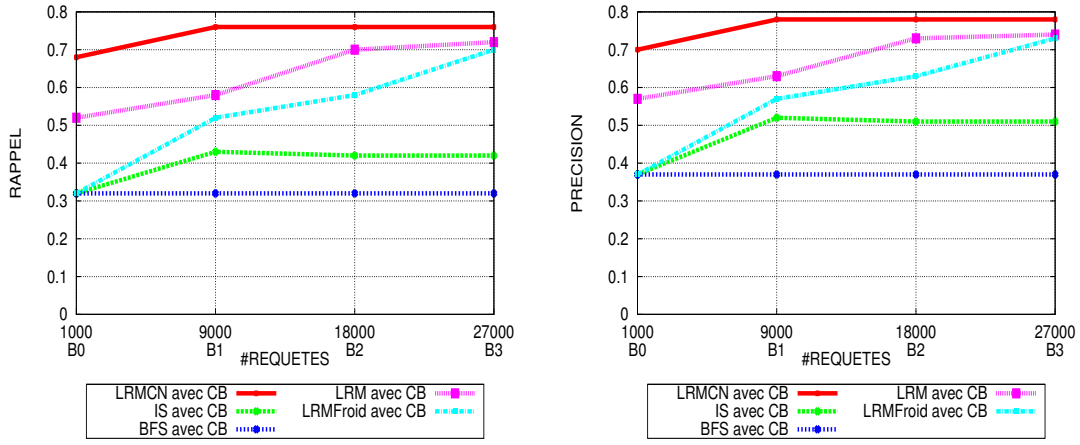
- TTL : Profondeur maximale de recherche, fixée à 4 ;
- $Pmax$: Nombre de pairs auxquels la requête doit être propagée, fixé à 4 ;
- *Taille du réseau* : Nombre de pairs dans le réseau, fixé à 810 (nombre de pairs dans le jeu de données).
- *Nombre d'amis* : Pour simuler la méthode $LRMCN$, nous avons fixé le nombre d'amis à 7.

Les méthodes LRM et $LRMCN$ démarrent avec une base de connaissances initiale B_0 générée lors du démarrage du système en lançant des requêtes extraites à partir de la collection locale de chaque pair. Dans les tests effectués, le nombre de requêtes lancées par chaque pair pour construire cette base initiale est égal à 3. Cependant, $LRMFroid$ démarre avec une base de connaissances vide.

Pour simuler le sixième scénario nous avons gardé les mêmes valeurs de TTL et $Pmax$ et nous avons varié la taille de réseau de 500 à 2000.

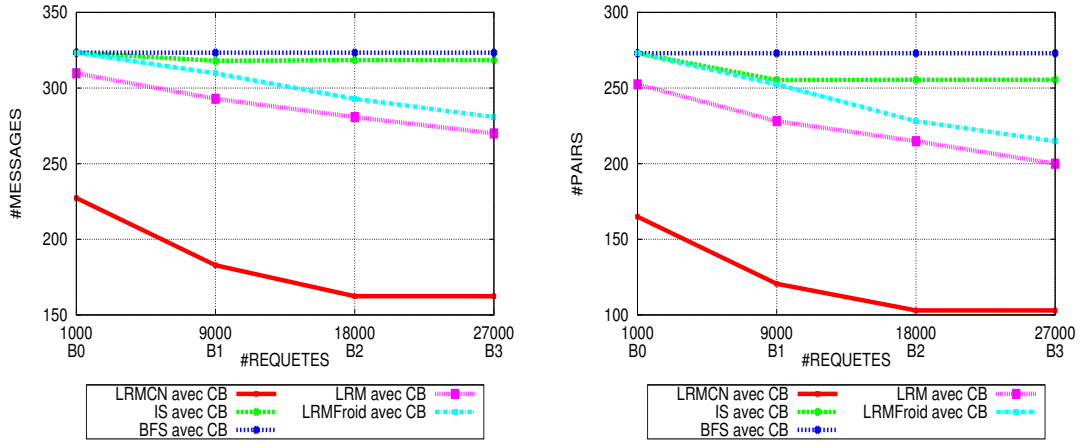
5.5.3 Résultats d'expérimentation

Nous présentons dans ce qui suit l'ensemble des scénarii de tests réalisés et les interprétations associées.



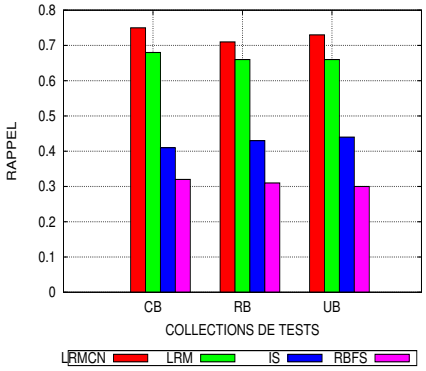
Rappel par requête(a) Précision par requête(b)

FIGURE 5.1 – Rappel et précision en fonction du nombre de requêtes des méthodes $RBFS$, IS, LRM et $LRMCN$

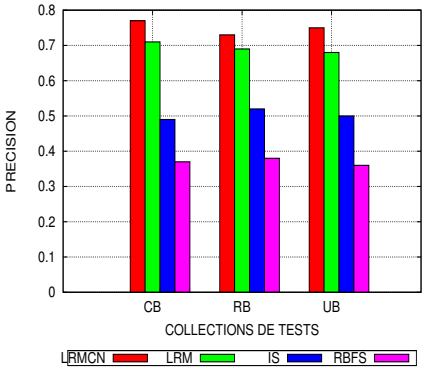


Nombre de messages par requêtes (a) Nombre de pairs par requête(b)

FIGURE 5.2 – Nombre de messages et nombre de pairs visités en fonction du nombre de requêtes des méthodes $RBFS$, IS, LRM et $LRMCN$

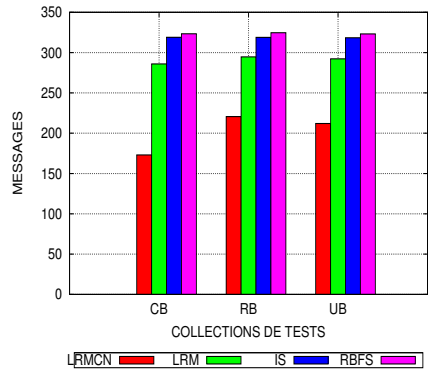


Rappel moyenne (a)

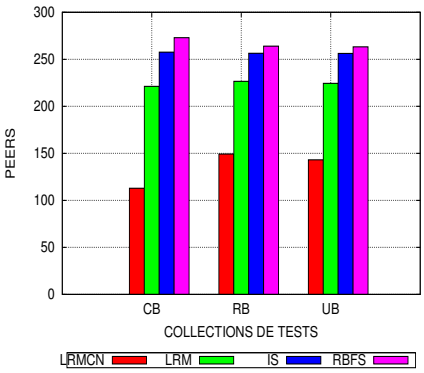


Précision moyenne (b)

FIGURE 5.3 – Moyenne du rappel et de la précision selon les collections de test CB, RB et UB



Moyenne du nombre de messages (a)



Moyenne du nombre de pairs(b)

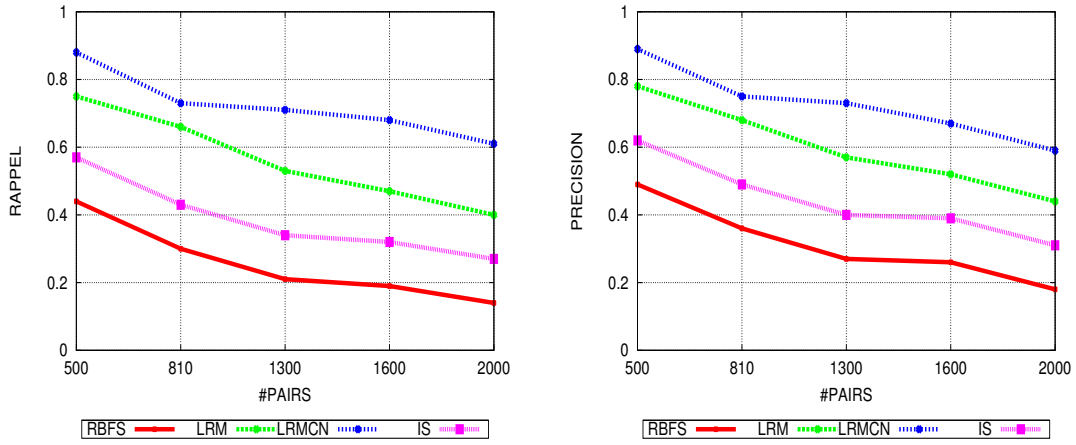
FIGURE 5.4 – Moyenne du nombre de messages et du nombre de pairs visités selon les collections de test CB, RB et UB

S1 : Impact du démarrage à chaud

Pour ce scénario, nous avons comparé l'efficacité et la performance de LRM par rapport à ceux de $LRMFroid$. En effet, pour comparer l'efficacité des deux méthodes, nous avons calculé la moyenne du rappel et de la précision par intervalle de 9000 requêtes envoyées par les différents pairs du système (i.e., 11 requêtes par pair). De même, pour comparer les performances, nous avons calculé la moyenne du nombre de messages et du nombre de pairs visités. Il est à noter que dans les différents tests nous avons utilisé la collection de test CB.

A. Test de l'efficacité de LRM et $LRMFroid$

Les figures 5.1 (a) et 5.1 (b) montrent qu'au démarrage le rappel et la précision de LRM sont supérieurs à ceux de $LRMFroid$. En effet, le rappel et la précision de LRM sont respectivement autour de 0.51 et 0.57, alors qu'ils sont autour de 0.3 et 0.37 pour $LRMFroid$. Ceci montre l'effet de la base de connaissances initiale. En démarrant



Rappel par requête(a) Précision par requête(b)

FIGURE 5.5 – Rappel et précision en fonction du nombre de pairs des méthodes $RBFS$, IS, LRM et $LRMCN$

avec cette base nous augmentons, durant la phase d'apprentissage, de 72% le rappel et la précision de $LRMFroid$. Ces résultats sont très encourageants, car ils donnent aux utilisateurs une bonne appréciation sur le système dès le lancement de la première requête.

B. Test de performance de LRM et $LRMFroid$

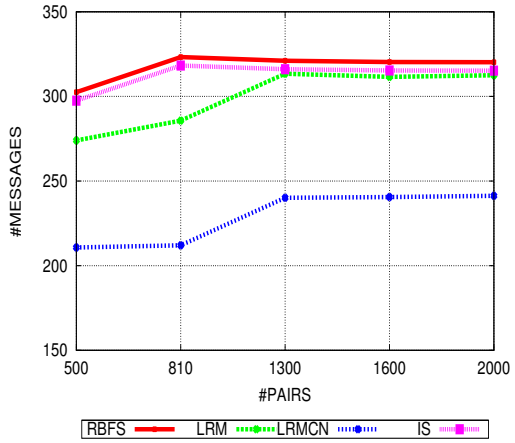
Les figures 5.2 (a) et 5.2 (b) montrent qu'au démarrage le nombre de messages et le nombre de pairs visités de LRM sont inférieurs à ceux de $LRMFroid$. En effet, le nombre de messages et le nombre de pairs visités de LRM sont respectivement autour de 309 et 252, alors qu'ils sont respectivement autour de 324 et 273 pour $LRMFroid$. En effet, avec une base de connaissances initiale, nous diminuons d'une manière significative le nombre de messages (i.e., environ 5%) et le nombre de pairs visités (i.e., environ 8%).

S2 : Impact de l'évolution des connaissances

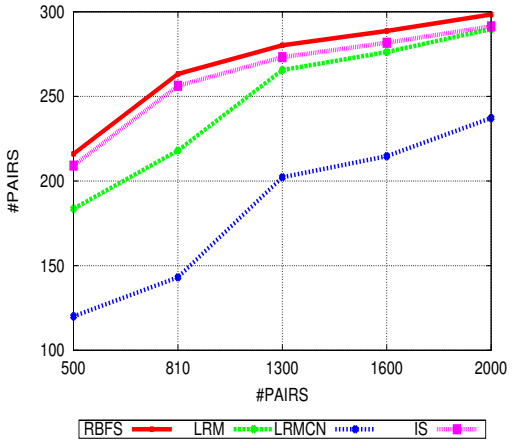
Pour étudier l'impact des mises à jour des connaissances sur l'efficacité de LRM et IS, nous avons calculé la moyenne du rappel et de la précision par intervalle de 9000 requêtes envoyées par les différents pairs du système. De même, pour étudier les performances de chaque méthode, nous avons calculé la moyenne du nombre de messages et du nombre de pairs visités. Les tests sont réalisées avec la collection CB.

A. Test de l'efficacité de LRM et IS

Les figures 5.1 (a) et 5.1 (b) montrent que le rappel et la précision de IS s'améliorent avec le temps, lorsque les profils des pairs voisins sont appris. En effet, le rappel et la précision de IS augmentent respectivement de 0.30 à 0.46 et de 0.36 à 0.53 après l'émission des 9000 premières requêtes. Cette augmentation est observée au début, cependant les valeurs du rappel et de la précision restent stables par la suite, car les tables de profils ont atteint la taille maximale, fixée à 500 entrées.



Nombre de messages par requêtes (a)



Nombre de pairs par requête(b)

FIGURE 5.6 – Nombre de messages et nombre de pairs visités en fonction du nombre de pairs des méthodes $RBFS$, IS, LRM et $LRMCN$

En outre, nous remarquons que le rappel et la précision relatives à LRM augmentent à chaque opération de mise à jour de la base des connaissances. En effet, la figure 5.1 (a) montre que le rappel de LRM augmente de 0.50 à 0.59 en utilisant la base B_1, de 0.59 à 0.68 en utilisant la base B_2 et augmente légèrement pour atteindre 0.70 en utilisant la base B_3. De même, la figure 5.1 (b) montre que le rappel de LRM augmente de 0.57 à 0.64 en utilisant la base B_1 et de 0.64 à 0.70 en utilisant la base B_2 et B_3.

B. Test de performance de LRM et IS

Les figures 5.2 (a) et 5.2 (b) montrent que le nombre de messages et le nombre de pairs visités de IS ont subit une baisse de 1% pour atteindre 318 et 255. Nous observons, ainsi, que le nombre de messages et le nombre de pairs visités de LRM diminuent après chaque opération de mise à jour de la base de connaissances. En effet, le nombre de messages et le nombre de pairs visités ont subit une baisse de 11% pour atteindre respectivement 280 et 218 après la troisième opération de mise à jour.

S3 : Impact de l'exploitation des corrélations entre les requêtes passées et les pairs positifs

Pour évaluer notre idée, qui consiste à représenter le profil utilisateur par des corrélations entre les requêtes passées et les pairs positifs, nous avons comparé l'efficacité et la performance de notre méthode LRM par rapport à celles de IS. Rappelons que, la méthode IS représente le profil utilisateur par les termes des requêtes passées sans prendre en compte les relations entre eux. En premier lieu, nous avons étudié le comportement de deux méthodes durant la simulation en utilisant la collection de test CB. En effet, pour comparer l'efficacité nous avons calculé la moyenne du rappel et de la précision par intervalles de 9000 requêtes envoyées par les différents pairs du système. De même, pour comparer la performance, nous avons calculé la moyenne du nombre

de messages et du nombre de pairs visités. En second lieu, nous avons comparé les résultats de ces méthodes selon les différentes collections de test distribuées CB, UB et RB.

A. Test de l'efficacité de LRM et IS

Les figures 5.1 (a) et 5.1 (b) montrent que le rappel et la précision de LRM sont plus élevés que ceux de IS durant toute la durée de simulation. En outre, les figures 5.3 (a) et 5.3 (b) montrent que l'efficacité de LRM excède celle de IS et $RBFS$ selon les trois collections. Nous observons dans la figure 5.3 (a) que LRM augmente de 65%, 53% et 50% le rappel donné par IS, respectivement selon les collections de test CB, RB et UB. Egalement, la figure 5.3 (b) précise que LRM augmente de 43%, 32% et 36% la précision donnée par IS, respectivement selon les collections de test CB, RB et UB. Par conséquent, nous pouvons dire que l'exploitation des corrélations entre les requêtes passées et les pairs positifs dans la représentation du profil utilisateur (LRM), induit une meilleure efficacité qu'une représentation plate de profil basée uniquement sur des statistiques (IS).

B. Test de performance de LRM, $RBFS$ et IS

Les figures 5.2 (a) et 5.2 (b) montrent que le nombre de messages et le nombre de pairs visités de LRM sont inférieurs à ceux de IS durant toute la durée de simulation. De plus, les figures 5.4 (a) et 5.4 (b) montrent que LRM est plus performante que IS selon les trois collections de test. Nous observons dans la figure 5.4 (a) que LRM diminue de 10%, 7% et 8% le nombre de messages générés par IS, respectivement selon les collections de test CB, RB et UB. Également, nous constatons (voir figure 5.4 (b)) que LRM diminue de 14%, 11% et 12% le nombre de pairs visités par IS, respectivement selon les collections de test CB, RB et UB. Ces résultats confirment que l'exploitation des corrélations entre les requêtes passées et les pairs positifs pour représenter le profil utilisateur (LRM), donne une meilleure performance qu'une représentation simple de profil (IS).

S4 : Impact de l'échec de sélection

Pour évaluer notre solution au problème d'échec de sélection dans les méthodes basées sur l'historique de requêtes, nous avons comparé l'efficacité et la performance de notre méthode de routage hybride $LRMCN$ par rapport à la méthode LRM. Rappelons que, LRM ne traite pas le problème d'échec de sélection. En premier lieu, nous avons étudié le comportement des deux méthodes en fonction du nombre de requêtes émises selon la collection de test CB. En effet, pour comparer l'efficacité, nous avons calculé la moyenne du rappel et de la précision par intervalles de 9000 requêtes envoyées par les différents pairs du système. De même, pour comparer la performance, nous avons calculé la moyenne du nombre de messages et du nombre de pairs visités. En second lieu, nous avons comparé les résultats de ces méthodes selon les différentes collections de test distribuées CB, UB et RB.

A. Test de l'efficacité de $LRMCN$ et LRM

Les figures 5.1 (a) et 5.1 (b) montrent que $LRMCN$ et LRM ont le même comportement, le rappel et la précision des deux méthodes augmentent à chaque opération de

mise jour. De plus, nous remarquons que le rappel et la précision de $LRMCN$ sont supérieurs à ceux de LRM durant toute la durée de simulation. En outre, les figures 5.3 (a) et 5.3 (b) montrent que l'efficacité de $LRMCN$ excède celle de LRM pour les trois collections de tests utilisées. Nous observons dans la figure 5.3 (a) que $LRMCN$ augmente le rappel donné par LRM respectivement de 10%, 8% et 10% selon les collections de test CB, RB et UB. Également, nous observons dans la figure 5.3 (b) que la précision de $LRMCN$ dépasse celle de LRM respectivement de 9%, 6% et 9% selon les collections de test CB, RB et UB.

Les différents tests réalisés prouvent que $LRMCN$ est plus efficace que LRM. De ce fait, nous pouvons déduire que le traitement du problème d'échec de sélection améliore d'une manière significative l'efficacité des méthodes de routage par apprentissage.

A. Test de performance de $LRMCN$ et LRM

Les figures 5.2 (a) et 5.2 (b) montrent que le nombre de messages et le nombre de pairs visités de $LRMCN$ sont inférieurs à ceux de LRM durant toute la durée de simulation. De plus, les figures 5.4 (a) et 5.4 (b) montrent que la méthode $LRMCN$ est plus performante que LRM selon les trois collections de test utilisées. Nous observons dans la figure 5.4 (a) que $LRMCN$ diminue respectivement de 39%, 25% et 27% le nombre de messages générés par LRM, selon les collections de test CB, RB et UB. Également, la figure 5.4 (b) montre que $LRMCN$ diminue le nombre de pairs visités par LRM respectivement de 48%, 34% et 36%, selon les collections de test CB, RB et UB. Ces résultats confirment que le traitement du problème d'échec de sélection améliore la performance des méthodes de routage par apprentissage.

S5 : Impact du modèle des distribution des jeux de données

A. Impact du modèle de distribution sur l'efficacité de $LRMCN$, LRM, IS et $RBFS$

Les figures 5.3 (a) et 5.3 (b) montrent que la précision et le rappel de IS sont très proches pour les trois collections de test. En plus, nous observons une légère augmentation (environ 5%) du rappel et de la précision des méthodes LRM et $LRMCN$ en utilisant la collection CB. Ceci est expliqué par le fait que dans la collection CB les pairs partagent des sous collections homogènes. Par conséquent, il y a une forte probabilité de trouver plus qu'un document pertinent à une requête donnée dans la sous collection d'un pair. En effet, lorsque $LRMCN$ ou LRM route la requête vers le meilleur pair ami, ce dernier peut fournir plusieurs documents pertinents, ce qui contribue dans l'augmentation du rappel et de la précision.

A. Impact du modèle de distribution sur la performance de $LRMCN$, LRM, IS et $RBFS$

Les figures 5.4 (a) et 5.4 (b) montrent que le nombre de messages et le nombre de pairs visités des méthodes $RBFS$, IS et LRM avec une distribution sémantique (Clustering Benchmark : CB) sont respectivement très proches de ceux calculés avec une distribution statistique (UB ou RB). La seule exception que nous avons observé est la diminution (environ 18%) du nombre de messages et du nombre de pairs visités de

LRMCN avec une distribution sémantique par rapport à une distribution statistique. Ceci est expliqué par le fait que l'algorithme de propagation des messages de la méthode *LRMCN* détermine rapidement les pairs pertinents à une requête donnée. Ces pairs ne traitent qu'une seule fois la même requête et stoppent la propagation de celle-ci lorsqu'ils la reçoivent une deuxième fois, ce qui contribue à la diminution du nombre de messages et du nombre de pairs visités.

S6 : Impact de changement du nombre de pairs dans le réseau

Pour évaluer le passage à l'échelle des méthodes proposées, nous avons étudié le comportement de *RBFS*, *IS*, *LRM* et *LRMCN* en fonction de la taille du réseau (nombre de pairs) selon la collection de test *UB*. Il est à noter que la collection de test est regénérée pour chaque taille de réseau spécifiée. En effet, pour comparer l'efficacité des différentes méthodes, nous avons calculé la moyenne du rappel et de la précision des requêtes envoyées par les différents pairs du système. De même, pour comparer la performance, nous avons calculé la moyenne du nombre de messages et du nombre de pairs visités.

A. Test de l'efficacité de *RBFS*, *IS*, *LRM* et *LRMCN*

Les figures 5.5 (a) et 5.5 (b) montrent que les quatre méthodes ont le même comportement, le rappel et la précision diminuent à chaque augmentation du nombre de pairs dans le réseau. Cela est très logique car en augmentant le nombre de pairs dans le réseau la probabilité de trouver des pairs pertinents diminue, ce qui influe négativement sur le rappel et la précision. En outre, la figure 5.5 montre que le rappel, (respectivement la précision), de *RBFS* se dégradent d'une manière significative (environ 68%) de 0.44 à 0.14 (respectivement de 0.49 à 0.19) ce qui prouve que la propagation aléatoire de requêtes ne permet pas le passage à l'échelle. Egalement, nous remarquons une baisse (environ 52%) du rappel, (respectivement de la précision), de la méthode *IS* de 0.57 à 0.27 (respectivement de 0.62 à 0.31). En effet, avec 2000 pairs la méthode *IS* devient inefficace.

De plus, nous remarquons une baisse du rappel, (respectivement de la précision), de *LRM* de 0.75 à 0.40 (respectivement de 0.78 à 0.44). Cette baisse est importante (alentours de 45%), cependant, le rappel et la précision de *LRM* restent acceptables. Nous observons ainsi, que le rappel, (respectivement la précision), de *LRMCN* diminuent de 0.88 à 0.61 (respectivement de 0.89 à 0.59). Bien que cette diminution est alentours de 30%, le rappel et la précision de *LRMCN* restent élevés (environ 0.61) vu que dans les simulations effectuées les documents ne sont pas répliqués. En effet, nous avons choisi le pire des cas. Les résultats de *LRMCN* sont encourageantes et montrent que notre méthode passe bien à l'échelle.

A. Test de performance de *RBFS*, *IS*, *LRM* et *LRMCN*

La figure 5.5 montre que le nombre de messages, (respectivement le nombre de pairs visités), de *RBFS* augmente de 300 avec 500 pairs à 320 avec 2000 pairs, (repectivement de 216 à 298). En effet, nous remarquons que le nombre de messages et le nombre de pairs visités avec un réseau de petite taille sont inférieurs à ceux avec un réseau de taille

plus importante. Ceci, est expliqué par fait que dans un petit réseau il y a une forte probabilité que les pairs reçoivent plusieurs fois les mêmes requêtes. Dans ce cas, ces pairs stoppent la propagation de celles-ci, ce qui contribue à la diminution du nombre de messages et de nombre de pairs visités. Ainsi, pour les même raisons IS a le même comportement de $RBFS$. Ce qui montre bien que ces méthodes ne permettent pas le passage à l'échelle.
En outre, nous observons que le nombre de messages repectivement le nombre de pairs visités de LRM sont inférieurs à ceux de IS et $RBFS$ avec un réseau de petite ou moyenne taille (500 ou 810 pairs). Cependant, ils sont proches lorsque le nombre des pairs dans le réseau devient important. Ceci, est expliqué par le fait que dans les tests éffectués, nous avons lancé le même nombre de requêtes en le redistribuant à chaque fois d'une manière uniforme sur les pairs du réseau. Par conséquent, le nombre de requêtes attribuées à chaque pair diminue en augmentant le nombre de pairs dans le réseau. De ce fait, la base de connsaissances de chaque pair est riche (i.e., contient plus des requêtes et des réponses) lorsque le nombre de pairs est moins élevé, ce qui permet de sélectionner plus de pairs pertients pour une requête donnée, de dimunier l'effet du problème d'échec de sélection et de contribuer à la dimunition de nombre de messages et de pairs visités. Contrairement, lorsque le nombre de pairs est élevé, la base de connsaissances de chaque pair contient peu de requêtes, ce qui augmente la probabilté d'échec de sélection et contribue à l'augmentation de nombre de messages et de pairs visités. Enfin, nous remarquons qu'en traitant le problème d'échec de sélection, la méthode $LRMCN$ passe bien à l'échelle. Le nombre de messages et le nombre de pairs visités de $LRMCN$ sont acceptables, alentours de 240 avec un réseau de grande taille.

5.6 Etude expérimentale de la solution proposée aux systèmes P2P mobiles

5.6.1 Scénarii de simulation

Dans les tests que nous effectuons, nous cherchons à valider les défis annoncés dans cette thèse, à savoir : *(i)* la sélection des meilleurs voisins pour une requête donnée et *(ii)* garantir que les pairs pertinents soient atteins. Pour valider notre proposition, nous avons comparé l'efficacité et la performance de notre méthode $CARM$ par rapport à la méthode $Gossiping - LB$. En effet, nous voulons comparer $CARM$ à une méthode qui ne considère pas le contenu de la requête pour voir l'impact de ce paramètre. En plus, notre vision de la stabilité du lien est différente à celle de $Gossiping - LB$. En effet, nous considérons plus qu'un paramètre pour garantir la stabilité (i.e., la vitesse, la charge, l'énergie de batterie, etc.), tandis que $Gossiping - LB$ ne considère que la charge des pairs voisins.

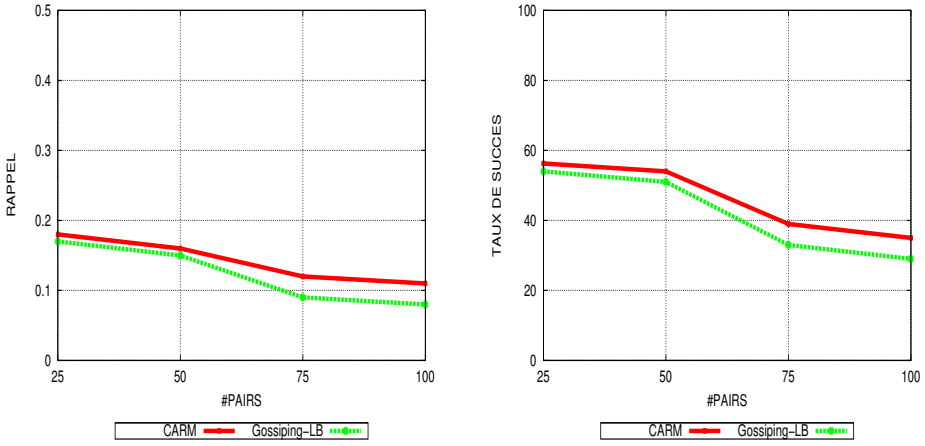
Plusieurs paramètres peuvent être pris en considération pour simuler les différentes méthodes de routage dans les systèmes P2P mobiles. Dans notre cas, nous avons simulé

le changement du nombre de pairs dans le réseau P2P et le changement de la vitesse de déplacements des nœuds mobiles dans le réseau. En effet, deux scénarii de test peuvent être réalisés afin de détecter l'impact de ces paramètres sur l'efficacité et la performance de notre méthode. Nous étudions les deux scénarii suivants :

S1. Impact de la mobilité des nœuds sur l'efficacité et la performance de notre méthode de routage ;

S2. Impact de la taille du réseau sur l'efficacité et la performance de notre méthode de routage.

5.6.2 Résultats d'expérimentation



Rappel (a) Taux de succès moyen (b)

FIGURE 5.7 – Rappel et taux de succès moyens de $CARM$ et $Gossiping - LB$ en fonction de la taille du réseau

Nous présentons dans ce qui suit l'ensemble des scénarii de tests réalisés et les interprétations associées.

S1 : Impact de la taille du réseau

Pour étudier l'impact de la taille du réseau sur l'efficacité et la performance de notre méthode $CARM$ et ceux de $Gossiping - LB$, nous avons varié le nombre de pairs de 25 à 100 en fixant la vitesse de déplacement des nœuds. Le tableau 5.2 résume les différents paramètres utilisés pour réaliser ce scénario de test.

A. Impact de la taille du réseau sur l'efficacité de $Gossiping-LB$ **et** $CARM$
Pour étudier l'impact de la taille du réseau sur l'efficacité de $Gossiping-LB$ et $CARM$, nous avons calculé la moyenne du rappel et du taux de succès pour toutes les requêtes soumises en variant le nombre des nœuds de 25, 50, 75 jusqu'à 100.

La figure 5.7 (a) montre que le rappel de $CARM$ est supérieur à celui de $Gossing-LB$ pour toutes les variations du nombre de pairs dans le réseau. Le rappel des deux

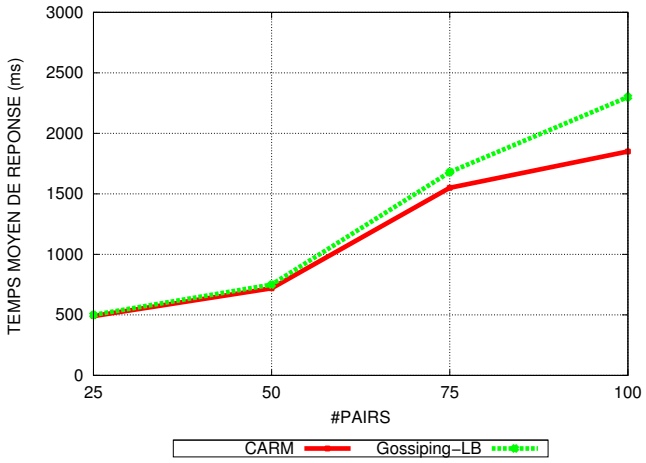
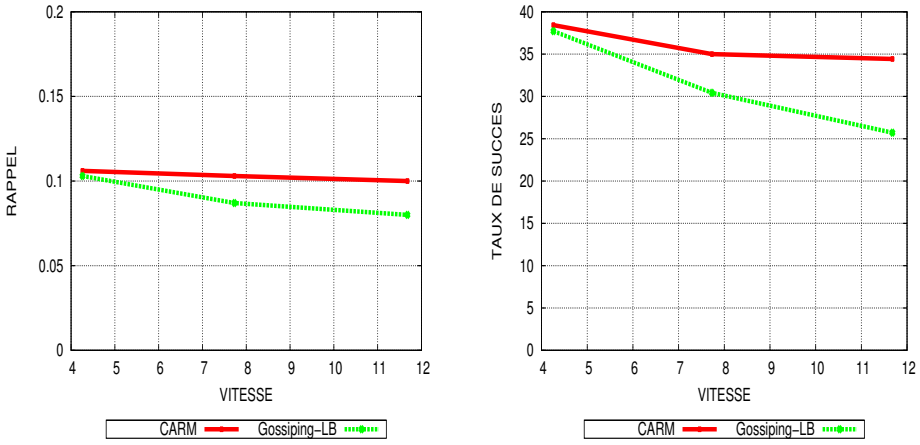


FIGURE 5.8 – Moyenne du temps de réponse de $CARM$ et $Gossiping - LB$ en fonction de la taille du réseau

TTL	Profondeur maximale de recherche, initialisé à 3.
K	Nombre de pairs auxquels la requête doit être propagée, initialisé à 3
Nombre de pairs	Nous avons varié ce nombre de 25 à 100
protocole MAC	IEEE 802.11b
Taille de la zone	$500 \times 500\ m^2$
Placement des nœuds	Distribution uniforme
Modèle de mobilité	Way-point aléatoire
Vitesse de déplacement des nœuds	$8 \leq V_2 \leq 9\ m/s$
La portée de transmission	100 m
Temps de pause	5 s

TABLE 5.2 – Paramètres de la simulation du scénario $S1$

méthodes est faible grâce aux facteurs de mobilité (i.e., nous avons choisis une configuration où la topologie du réseau est très dynamique) et de la non réplication des documents (i.e., dans la collection de test, le taux de réplication est égal à 1). De même, la figure 5.7 (b) montre que le taux de succès de $CARM$ est meilleurs que celui de $Gossiping - LB$ pour toutes les variations du nombre de pairs dans le réseau. Nous observons, ainsi que notre méthode améliore de 2% à 6% le taux de succès de $Gossiping - LB$ respectivement avec 25 et 100 pairs. En effet, $CARM$ considère en plus de la charge d'un nœud, le contenu de la requête et la stabilité des liens, ces deux derniers paramètres contribuent bien respectivement à la sélection des pairs pertinents et à la réception des réponses de ceux-ci. Dans un environnement mobile, il ne suffit pas de retrouver les pairs pertinents, il faut aussi garantir un chemin stable entre ces derniers et l'initiateur de la requête pour recevoir leurs réponses.



Rappel (a) Taux de succès moyen (b)

FIGURE 5.9 – Rappel et taux de succès moyens de $CARM$ et $Gossiping - LB$ en fonction de la vitesse des nœuds

De plus, nous remarquons que le taux de succès et le rappel des deux méthodes diminuent en augmentant la taille de réseau. Ceci peut être expliqué par le fait que la probabilité de retrouver un pair pertinent dans un réseau de petite taille est plus forte que de le retrouver dans un réseau de grande taille.

B. Impact de la taille du réseau sur la performance de $Gossiping - LB$ et $CARM$

Pour étudier l'impact de la taille du réseau sur la performance de $Gossiping - LB$ et $CARM$, nous avons calculé la moyenne du temps de réponse de toutes les requêtes soumises en variant le nombre de nœuds de 25, 50, 75 jusqu'à 100.

La figure 5.8 montre que le temps de réponse moyen de $CARM$ est inférieur à celui de $Gossiping - LB$ pour toutes les variations du nombre de pairs dans le réseau. En effet, $CARM$ considère le contenu de la requête, ce qui permet de retrouver des pairs pertinents avec un minimum de nombre de sauts. En minimisant le nombre de sauts entre l'initiateur de la requête et le pair destination, le nombre de relais diminue ce qui permet de réduire le temps de réponse.

De plus, nous remarquons qu'en augmentant la taille du réseau, le temps de réponse moyen des deux méthodes augmente proportionnellement. Ceci est expliqué par le fait que l'augmentation de la taille du réseau engendre une augmentation de trafic (flux), ce qui provoque un ralentissement global de celui-ci d'où l'augmentation des délais d'attente de réponses.

S2 : Impact de la mobilité

Pour étudier l'impact de la mobilité des nœuds sur l'efficacité et la performance de notre méthode $CARM$ et ceux de $Gossiping - LB$, nous avons varié la vitesse de

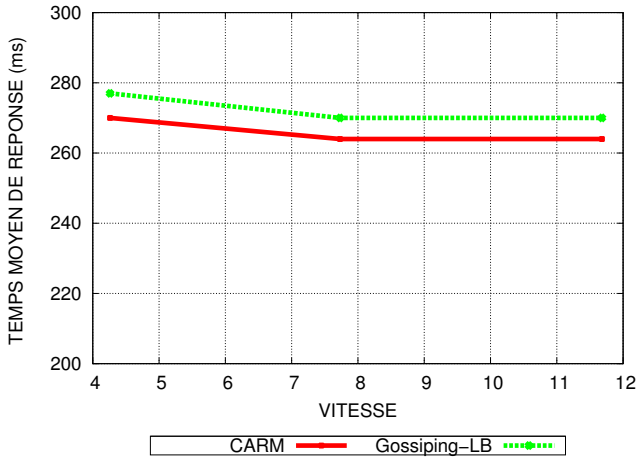


FIGURE 5.10 – Temps de réponse moyen de $CARM$ et $Gossiping - LB$ en fonction de la vitesse des nœuds

TTL	Profondeur maximale de recherche, initialisé à 3.
K	Nombre de pairs auxquels la requête doit être propagée, initialisé à 3
Nombre de pairs	100
protocole MAC	IEEE 802.11b
Taille de zone	$500 \times 500\ m^2$
Placement des nœuds	Distribution uniforme
Modèle de mobilité	Way-point aléatoire
Vitesse de déplacement des nœuds	nous avons varié la vitesse moyenne de 4.26, 7.73 à 11.68 m/s
La portée de transmission	75 m
Temps de pause	3 s

TABLE 5.3 – Paramètres de la simulation du scénario $S2$

déplacement des nœuds en fixant le nombre de nœuds à 100. Le tableau 5.3 résume les différents paramètres utilisés pour réaliser ce scénario de test.

A. Impact de la mobilité sur l'efficacité de $Gossiping - LB$ et $CARM$

Pour étudier l'impact de la mobilité sur l'efficacité de $Gossiping - LB$ et $CARM$, nous avons calculé la moyenne du rappel et du taux de succès pour toutes les requêtes soumises en variant la vitesse moyenne de 4.26, 7.73 à 11.68 m/s.

La figure 5.9 (a) montre que le rappel et le taux de succès de $CARM$ et $Gossiping - LB$ se dégradent en augmentant la vitesse. Ceci est simplement expliqué par le fait qu'en augmentant la vitesse, les routes deviennent instables. En effet, plusieurs pairs pertinents sont localisés mais leurs réponses n'atteignent pas l'initiateur de la requête, ce qui contribue à la baisse à la fois du rappel et du taux de succès moyen. De plus, nous observons que le rappel et le taux de succès moyen sont supérieurs à ceux de $Gossiping -$

LB pour toutes les variations de la vitesse. Ainsi, nous remarquons que $CARM$ résiste mieux aux changements de vitesse que $Gossiping-LB$. En effet, l'efficacité de $CARM$ se dégrade (environ 3%) une seule fois en variant la vitesse de 4.26 à 7.73, tandis que l'efficacité de $Gossiping-LB$ se dégrade pour chaque augmentation de vitesse.

B. Impact de la mobilité sur la performance de $Gossiping-LB$ et $CARM$ Pour étudier l'impact de la mobilité sur la performance de $Gossiping-LB$ et $CARM$, nous avons calculé le temps de réponse moyen pour toutes les requêtes soumises en variant la vitesse moyenne de 4.26, 7.73 à 11.68 m/s.

La figure 5.10 montre que le temps de réponse moyen des deux méthodes a subit une baisse en augmentant la vitesse de déplacement. Nous remarquons que la vitesse n'a pas un impact sur le temps de réponse des deux méthodes. En effet, le comportement des deux méthodes est le même, ce qui fait que le taux de réponse moyen de $CARM$ reste inférieur à celui de $Gossiping-LB$ pour toutes les variations de la vitesse.

5.7 Conclusion

Dans ce chapitre, nous avons présenté en détail les collections de test et les mesures d'évaluation que nous avons utilisées pour évaluer nos méthodes. Ensuite, nous avons décrit l'environnement de simulations des systèmes P2P sur Internet et mobiles. Une étude expérimentale pour chaque type de systèmes est détaillée. Dans chaque étude expérimentale différents scénarios de tests sont discutés. Dans les systèmes P2P sur Internet, nous avons réalisé des scénarii de test qui permettent d'étudier respectivement les impacts du démarrage à chaud, des mises à jour des bases de connaissances, de l'exploitation des corrélations entre les requêtes passées et les pairs positifs dans la représentation du profil d'utilisateur, de l'échec de sélection, du modèle de distribution de jeu de données modèles de distribution (i.e., statistique ou sémantique) et la taille du réseau sur l'efficacité et la performance de nos méthodes de routage LRM et $LRMCN$. De même, dans les systèmes P2P mobiles, nous avons réalisé des scénarii de test qui permettent d'étudier respectivement l'impact de la taille du réseau et la mobilité sur l'efficacité et la performance de notre méthode de routage $CARM$. Les différentes expérimentations ont montré l'amélioration apportée par nos méthodes par rapport aux méthodes de l'état de l'art.

Conclusion générale et perspectives

Les travaux présentés dans cette thèse s'inscrivent dans le contexte général de la Recherche d'Information Distribuée. Plus particulièrement, nous avons abordé le problème de routage de requêtes dans les systèmes à large échelle à savoir les systèmes P2P non structurés.

L'objectif principal recherché dans cette thèse était de définir de nouvelles méthodes de routage de requêtes dans les systèmes P2P non structurés sur Internet ou sur des réseaux mobiles. La première partie de la thèse, s'est focalisée sur le routage de requêtes dans les systèmes P2P sur Internet. C'est ainsi, que nous avons proposé *(i)* un modèle de routage sémantique basé sur l'historique des requêtes. Dans ce modèle, le profil de l'utilisateur est représenté par un ensemble d'intérêts. Chaque intérêt représente des relations sémantiques entre les requêtes passées, leurs termes et les pairs positifs. Ensuite, ce modèle est instancié pour définir une nouvelle méthode de routage par apprentissage (Learning Routing Method : LRM). Pour pallier le problème de démarrage à froid, *(ii)* nous avons proposé une méthode prédictive de l'intention de l'utilisateur qui construit une base de connaissances à priori pour chaque pair. Enfin, *(iii)* nous avons proposé une méthode de routage hybride (Learning Routing Method based on Clustred Network : $LRMCN$) pour traiter le problème d'échec de sélection. Cette méthode est basée sur l'historique des requêtes et le regroupement de pairs dans des groupes sémantiques.

La deuxième partie de la thèse, s'est focalisée sur le routage de requêtes dans les systèmes P2P mobiles. L'apparition des MANETs, a soulevé de nouveaux challenges de routage. Ces réseaux souffrent de plusieurs contraintes liées aux supports de transmission ou bien liées aux dispositifs mobiles. Les problèmes majeurs sont la mobilité des nœuds et la faible puissance d'énergie de batterie des nœuds mobiles. Dans ce cadre, nous avons proposé une méthode de routage pour les systèmes P2P non structurés mobiles basée sur le contexte de l'utilisateur (Context-Aware Routing Method : $CARM$). Notre méthode permet de sélectionner les meilleurs voisins pour une requête donnée en se basant sur le contenu de celle-ci et les profils des voisins. De plus, elle considère les contraintes du réseau MANET (la puissance d'énergie de la batterie, la mobilité et la charge du nœud) afin de garantir que les pairs pertinents soient atteint.

Pour valider nos propositions pour les systèmes P2P sur Internet, nous avons implanté les différentes méthodes proposées sur le simulateur PeerSim. Ainsi, une démarche de validation par simulation a été définie. Sur la base de trois collections de tests standards, les résultats d'expérimentation sont comparés à ceux des approches existantes, sur le plan de l'efficacité et des performances. Les différents scénarios réalisés ont prouvé l'efficacité et la performance de nos méthodes de routage par rapport aux approches existantes selon les différents modèles de distribution de documents et de requêtes.

Pour évaluer notre méthode de routage de requêtes dans les systèmes P2P mobiles ($CARM$), nous l'avons implanté sur le simulateur NS2. Ainsi, nous avons comparé l'efficacité et la performance de $CARM$ par rapport à une méthode existante ($Gossiping - LB$). Une série d'expérimentations selon différents scénarios de test a été réalisée, prouve que notre méthode est plus efficace et plus performante que Gossiping-LB.

Perspectives

Les perspectives pouvant faire suite à ces travaux sont multiples. Tout d'abord, comme nous l'avons vu dans ce document, le rôle joué par l'utilisation des profils dans le routage des requêtes est crucial. De plus, ces profils ou contexte peuvent être présents dans de nombreuses autres applications de recherche d'information à large échelle. Par conséquent, il serait probablement bénéfique d'approfondir leurs études. Un premier projet consiste à étudier de façon générique les différents types d'informations constituant un profil ainsi que les techniques sous-jacentes à leur exploitation. Une modélisation générique des profils utilisateurs supportant différentes dimensions est nécessaire. Le profil d'un utilisateur peut contenir différents types d'informations : des données statiques, le centre d'intérêt qui peut être implicite ou explicite, la localisation géographique, les ressources partagées, etc. Ces informations de profil peuvent être utilisées pour sélectionner les meilleurs pairs pour les requêtes ultérieures de l'utilisateur.

Nous envisageons également de définir un processus d'adaptation du système à l'environnement de l'utilisateur ou à son expertise. C'est-à-dire guider le système dans l'évaluation de ses requêtes en se basant sur les dimensions les plus adéquates.

Outre l'élargissement du domaine d'utilisation des profils utilisateurs dans le contexte du RILE (recommandation, recherche, expansion de requêtes, etc.), il serait intéressant, de proposer diverses optimisations pour améliorer les performances du système.

Les résultats de nos travaux ont de plus permis de mettre en évidence certains manques dans l'évaluation du protocole de routage dans le contexte des réseaux MANETs (nous n'avons pas étudié l'impact de la variation des constantes PL et PS et $MaxTime$ qui sont utilisées pour ajuster la valeur de la fonction de pertinence sur l'efficacité et la performance de notre méthode de routage). Ainsi, une évaluation plus poussée est en cours.

Enfin, nous envisageons une évaluation *in situe* sur une application réelle des différentes méthodes proposées.

Bibliographie

[Agarwal 2000] Sulabh Agarwal, Ashish Ahuja, Jatinder Pal Singh et Rajeev Shorey. *Route-Lifetime Assessment Based Routing (RABR) Protocol for Mobile Ad-Hoc Networks.* In International Conference on Communications (ICC' 2000), pages 1697–1701, New Orleans, Louisiana, USA, June 18-22 2000.

[Anne M. 2005] Devin V. Anne M. Crowley J.L. et Privat G. *Localisation intra-bâtiment multitechnologies : RFID.* In Deuxièmes Journées Francophones : Mobilité et Ubiquité (UbiMob' 2005), pages 29–35, Grenoble, France, Mai 31 - June 3 2005.

[Bender 2007] Matthias Bender et Gerhard Weikum. *Advanced Methods for Query Routing in Peer-to-Peer Information Retrieval.* PhD thesis, Saarlandes University, July 2007.

[Bluetooth 2012] Bluetooth. *Bluetooth.* http://simple.wikipedia.org/wiki/Bluetooth, March, 2012.

[Bouzeghoub 2005] Mokrane Bouzeghoub et Dimitre Kostadinov. *Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils.* In COnférence en Recherche d'Infomations et Applications (CORIA' 2005, pages 201–218, Grenoble, France, Mars 9-11 2005.

[Burrell J. 2001] Kubo K. Burrell J. Gray G.K. et Farina N. *Context-aware computing : a text case.* In 4th International Conference on Ubiquitous Computing (UbiComp' 2001), pages 1–15, Berkeley, California, USA, September 30 - October 2 2001.

[Cheverest K. 2002] Mitchell K. Cheverest K. et Davies N. *The role of adaptive hypermedia in a context-aware tourist guide.* IEEE Wireless Communications, vol. 45, no. 5, pages 47–51, 2002.

[Chlamtac 2003] Imrich Chlamtac, Marco Conti et Jennifer J.-N. Liu. *Mobile ad hoc networking : imperatives and challenges.* Ad Hoc Networks, vol. 1, no. 1, pages 13–64, 2003.

[Chord 2008] Chord. *Chord website.* http://pdos.csail.mit.edu/chord, Octobre 2008.

[Christoph 2004] T. Christoph, S. Steffen et W Adrian. *Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors.* In 13th International World Wide Web Conference (WWW' 2004), pages 55–68, New York City, USA, May 17-22 2004.

[Ciraci 2009] Selim Ciraci, İbrahim Körpeoglu et Özgür Ulusoy. *Reducing query overhead through route learning in unstructured peer-to-peer network.* J. Netw. Comput. Appl., vol. 32, no. 3, pages 550–567, 2009.

[Clarke 2001] Ian Clarke, Oskar Sandberg, Brandon Wiley et Theodore W. Hong. *Freenet : a distributed anonymous information storage and retrieval system.* In International workshop on Designing privacy enhancing technologies : design issues in anonymity and unobservability, pages 46–66, Berkeley, California, USA, July 25-26 2001.

[Crespo 2002] Arturo Crespo et Hector Garcia-Molina. *Routing Indices For Peer-to-Peer Systems.* In Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), pages 23–30, Vienna, Austria, July 2-5 2002.

[daH 2009] *Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks.* Computer Communications, vol. 32, pages 1445 – 1459, 2009.

[Duda 2000] Richard O. Duda, Peter E. Hart et David G. Stork. Pattern classification (2nd edition). Wiley-Interscience, 2000.

[Ganter 1997] B. Ganter et R. Wille. Formal concept analysis : Mathematical foundations. Springer-Verlag New York, 1997.

[Gnutella 2009] Gnutella. *Gnutella Web site.* `http://www.gnutella.com`, March, 2009.

[Godin 1995] R. Godin, R. Missaoui et H. Alaoui. *Incremental concept formation algorithms based on galois (concept) lattices.* Computational Intelligence, vol. 11(2), pages 246–267, 1995.

[Goh 2005] S. Goh, P. Kalnis, S. Bakirs et K. L. Tan. *Real Datasets for File-Sharing Peer-to-Peer Systems.* In 10th International Conference on Database Systems for Advanced Applications (DASFAA' 2005), pages 201–213, Beijing, China, April 17-20 2005.

[Gupta 2011] Nishant. Gupta, Kopal. Bafna et Neeraj Gupta. *Efficient Data Replication Algorithm for Mobile Ad-Hoc.* In Seventh International Conference on Mobile Ad-hoc and Sensor Networks (MSN' 2011), pages 1697–1701, Beijing, China, December 16-18 2011.

[Hwa 2009] *Cross-layer design of P2P file sharing over mobile ad hoc networks.* Telecommunication Systems, vol. 42, no. 1, 2009.

[Issariyakul 2008] Teerawat Issariyakul et Ekram Hossain. Introduction to network simulator ns2. Springer Publishing Company, Incorporated, 1 édition, 2008.

[Jacquet 2001] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum et L. Viennot. *Optimized Link State Routing Protocol for Ad Hoc Networks.* In Proceedings of IEEE International on Multi Topic Conference. Technology for the 21st Century (IEEE INMIC' 2001), pages 62–68, Lahore University of Management Sciences, Pakistan, December 28-30 2001.

[Jelasity 2010] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi et Spyros Voulgaris. *The Peersim Simulator.* `http://peersim.sf.net`, March 2010.

[Jochen 2003] Schiller Jochen. Mobile communications. 2003.

[Johnson 2001] David B. Johnson, David A. Maltz et Josh Broch. *DSR : The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks.* In In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5, pages 139–172. Addison-Wesley, 2001.

[Kalogeraki Vana 2002] Gunopulos Dimitrios Kalogeraki Vana et Zeinalipour-Yazti D. *A local search mechanism for peer-to-peer networks.* In Proceedings of the eleventh international conference on Information and knowledge management (CIKM' 2002), pages 300–307, McLean, Virginia, USA, November 4-9 2002.

[Kanungo] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman et Angela Y. Wu. *An Efficient k-Means Clustering Algorithm : Analysis and Implementation.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7.

[Karasabun 2009] E. Karasabun, D. Ertemur, S. Sariyildiz, M. Tekkalmaz et I. Korpeoglu. *A path-quality-aware peer-to-peer file sharing protocol for mobile ad-hoc networks : Wi-share.* In 24th International Symposium onComputer and Information Sciences (ISCIS' 2009), pages 322 –327, Guzelyurt, Northern Cyprus, September 14-16 2009.

[Karp 2000] Brad Karp et H. T. Kung. *GPSR : greedy perimeter stateless routing for wireless networks.* In Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom' 2000), pages 243–254, New York, NY, USA, August 6-11 2000.

[Kazaa 2008] Kazaa. *Kazaa website.* http://www.kazaa.com, septembre 2008.

[Klemm 2003] Alexander Klemm, Er Klemm, Christoph Lindemann et Oliver P. Waldhorst. *A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks.* In IEEE Semiannual Vehicular Technology Conference (VTC' 2003-Fall), Orlando, Florida, USA, October 6-9 2003.

[Kumar 2012] D. Kishore Kumar et Y. S. S. R. Murthy. *Article : Hierarchical Routing Protocol for Free-Space Optical Mobile Ad hoc Networks (FSO/RF MANET).* International Journal of Computer Applications, vol. 60, no. 10, pages 1–7, 2012.

[Leung 2008] Andrew Ka-Ho Leung et Yu-Kwong Kwok. *On Localized Application-Driven Topology Control for Energy-Efficient Wireless Peer-to-Peer File Sharing.* IEEE Transactions on Mobile Computing, vol. 7, no. 1, pages 66–80, 2008.

[Li 2009] Ting Li, Hong Ji, Jingqing Mei, Yi Li et Chao Hu. *Topology Mismatch Avoidable Cross-Layer Protocol for P2P File Discovery in MANETs.* In Wireless Communications and Networking Conference, (WCNC' 2009), pages 1–5, Shanghay, China, April 7-10 2009.

[Lv 2002] Qin Lv, Pei Cao, Edith Cohen, Kai Li et Scott Shenker. *Search and replication in unstructured peer-to-peer networks.* In Proceedings of the 16th international conference on Supercomputing (ICS '2002), pages 84–95, New York, USA, 2002.

[Makhoul 1999] John Makhoul, Francis Kubala, Richard Schwartz et Ralph Weischedel. *Performance Measures For Information Extraction.* In Proceedings of DARPA Broadcast News Workshop (DARPA' 1999), pages 249–252, Herndon, VA, February 1-4 1999.

[Mandreoli 2009] Federica Mandreoli, Riccardo Martoglia, Wilma Penzo et Simona Sassatelli. *Data-Sharing P2P Networks with Semantic Approximation Capabilities.* IEEE Internet Computing, vol. 13, pages 60–70, 2009.

[Mostefaoui K. 2004] Pasquier-Rocha J. Mostefaoui K. et Brézillon P. *Context-aware computing : a guide for the pervasive computing community.* In Proceedings of the IEEE/ACS International Conference on Pervasive Services (IPCS'2004), pages 39–48, American University of Beirut (AUB), Lebanon, July 23-23 2004.

[Napster 2008] Napster. *Napster website.* `http://www.napster.com`, septembre 2008.

[NTCIR 2010] NTCIR. *NII Test Collection for IR Systems.* `http://research.nii.ac.jp/ntcir/index-en.html`, March, 2010.

[Pastry 2008] Pastry. *Pastry website.* `http://research.microsoft.com`, Octobre 2008.

[Perkins 1994] Charles E. Perkins et Pravin Bhagwat. *Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers.* In Proceedings of the conference on Communications architectures, protocols and applications (SIGCOMM' 1994), pages 234–244, London, UK, August 31 - September 2 1994.

[Perkins 2003] C. Perkins, E. Belding-Royer et S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*, 2003.

[Raftopoulou 2009] Paraskevi Raftopoulou, Euripides G. Petrakis et Christos Tryfonopoulos. *Rewiring strategies for semantic overlay networks.* Distrib. Parallel Databases, vol. 26, no. 2-3, pages 181–205, 2009.

[RARE 2010] RARE. *Le projet RARE (Routage optimisé par Apprentissage de REquêtes).* `http://www-inf.it-sudparis.eu`, March, 2010.

[Sacha 2006] Jan Sacha, Jim Dowling, Raymond Cunningham et Rena Meier. *Discovery of Stable Peers in a Self-organising Peer-to-Peer Gradient Topology.* In Distributed Applications and Interoperable Systems (DAIS' 2006), pages 70–83, Bologna, Italy, June 13-16 2006.

[Schilit B.N. 2002] Hilbert D.M. Schilit B.N. et Trevor J. *Context-aware communication.* IEEE Wireless Communications, vol. 9, no. 5, pages 37–45, 2002.

[Schütze 1997] Hinrich Schütze et Craig Silverstein. *Projections for efficient document clustering.* SIGIR Forum, vol. 31, pages 74–81, 1997.

[Shah 2009] N. Shah et Depei Qian. *Context-Aware Routing for Peer-to-Peer Network on MANETs.* In IEEE International Conference on Networking, Architecture, and Storage (NAS' 2009), pages 135 –139, Hunan, China, July 2009.

[SHEN 2011] Wen wu SHEN, Sen SU, Kai SHUANG et Fang chun YANG. *SKIP : an efficient search mechanism in unstructured P2P networks.* The Journal of China Universities of Posts and Telecommunications, vol. 17, no. 5, pages 64–71, 2011.

[Shi 2008] Cong Shi, Dingyi Han, Yuanjie Liu, Shicong Meng et Yong Yu. *A dynamic routing protocol for keyword search in unstructured peer-to-peer networks.* Computer Communications, vol. 31, no. 2, pages 318–331, 2008.

[TREC 2010] TREC. *Text REtrival Conference.* http://trec.nist.gov, March, 2010.

[Tsoumakos 2003] Dimitrios Tsoumakos et Nick Roussopoulos. *Adaptive Probabilistic Search for Peer-to-Peer Networks.* In Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P'2003), pages 102–109, September 1-3 2003.

[Wifi 2012] Wifi. *Wifi.* http://simple.wikipedia.org/wiki/Wifi, March, 2012.

[Wikipedia 2011] Wikipedia. *Pastry website.* http://fr.wikipedia.org/wiki/2011_en_informatique, Janvier 2011.

[Yang 2002] Beverly Yang et Hector Garcia-Molina. *Improving Search in Peer-to-Peer Networks.* In The 22nd International Conference on Distributed Computing Systems (ICDCS' 2002), pages 5–14, Vienna, Austria, 2002.

[Yeferny 2010] Taoufik Yeferny et Khedija Arour. *LearningPeerSelection : A Query Routing Approach for Information Retrieval in P2P Systems.* In International Conference on Internet and Web Applications and Services (ICIW' 2010), pages 235–241, Barcelona, Spain, May 09-15 2010.

[Yeferny 2012a] Taoufik Yeferny et Khedija Arour. *Efficient Routing method in P2P systems based upon Training Knowledge.* In The Eighth International Symposium on Frontiers of Information Systems and Network Applications, in conjunction with AINA (WAINA'2012), pages 300–305, Fukuoka, Japan, 2012.

[Yeferny 2012b] Taoufik Yeferny, Amel Bouzeghoub et Khedija Arour. *Context-Aware Routing Method for P2P File Sharing Systems over MANET.* In Proceedings of the 2nd International Workshop on Information Management for Mobile Applications (IMMoA' 2012), in conjunction with (VLDB' 2012), pages 21–25, Istanbul, Turkey, August 31 2012.

[Yeferny 2013a] Taoufik Yeferny, Khedija Arour et Amel Bouzeghoub. *LRS : A Novel Learning Routing Scheme for Query Routing on Unstructured P2P Systems.* Transactions on Large-Scale Data- and Knowledge-Centered Systems XII, vol. 13, pages 54–82, 2013.

[Yeferny 2013b] Taoufik Yeferny, Amel Bouzeghoub et Khedija Arour. *An Efficient Peer-to-Peer Semantic Overlay Network for Learning Query Routing.* In To appear on The 27th IEEE International Conference on Advanced Information Networking and Applications (AINA' 2013), Barcelone, Spain, March 25-28 2013.

[Zammali 2010] Saloua Zammali et Khedija Arour. *P2PIRB : benchmarking framework for P2PIR.* In Proceedings of the Third international conference on Data management in grid and peer-to-peer systems (Globe'2010), pages 100–111, Bilbao, Spain, 2010.

[Zhu 2005] Yingwu Zhu, Xiaoyu Yang et Yiming Hu. *Making Search Efficient on Gnutella-Like P2P Systems.* In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'2005), pages 1–56, Washington, DC, USA, April 3-8 2005.