République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université de Bordj Bou Arreridj
Faculté des Mathématiques et d'informatique
Département d'informatique

Thèse de Doctorat

Par:

Cherbal Sarra

Pour Obtenir le grade de Docteur en :

Domaine : **mathématiques et informatique**
Filière : **Informatique**
Option : **Informatique distribuée et informatique décisionnelle**

Titre

Optimisation de l'énergie dans les systèmes pair–à–pair

Devant le jury composé de :

M. Mostefai Messaoud	Professeur à l'université de Bordj Bou Arreridj	Président
M. Tari Kamel	Professeur à l'université de Bejaia	Examinateur
M. Semchedine Fouzi	Maitre de conférences A à l'université de Sétif	Examinateur
M. Rahmoune Azedine	Maitre de conférences A à l'université de Bordj Bou Arreridj	Examinateur
M. Boukerram Abdellah	Professeur à l'université de Bejaia	Rapporteur
M. Boubetra Abdelhak	Professeur à l'université de Bordj Bou Arreridj	Co-rapporteur

Année Universitaire 2016–2017

Remerciements

« Après avoir remercié ALLAH le tout puissant »

Au terme de ce travail, je tiens à exprimer ma profonde reconnaissance à mon directeur de thèse, Professeur Abdellah BOUKERRAM qui m'a guidé tout au long de ces années de recherche. Je le remercie pour toute l'attention qu'il a portée à mon travail, pour ses conseils ainsi que pour son soutien moral. Je remercie profondément Professeur Abdelhak BOUBETRA d'avoir encadré mon travail avec beaucoup de soutien et de conseils scientifiques et pédagogiques. J'ai pu à plusieurs reprises apprécier et profiter de leur haute compétence scientifique.

Je souhaite aussi présenter mes remerciements aux membres de jury qui ont accepté d'évaluer ce travail.

Je suis très reconnaissante au Professeur Messaoud MOSTEFAI de me faire l'honneur de présider le jury de cette thèse. J'adresse mes sincères remerciements au Professeur Abdelkamel TARI, Docteur Fouzi SEMCHEDINE et Docteur Azedine RAHMOUNE de m'avoir fait l'honneur de participer au jury de ma thèse.

J'adresse mes remerciements au Docteur Mahfoud BENCHAIBA pour l'aide qu'il m'a aimablement apporté.

Bien sûr, je remercie profondément mes chers parents pour leurs sacrifices, leur disponibilité et leur soutien tout au long de mes années de travail. Je remercie ma grand-mère, ma tante, mon frère et ma sœur pour leurs prières et souhaits de réussite et pour tout.

Un merci à toute ma famille, à tous mes proches, et à tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail.

Résumé

Les systèmes pair-à-pair à base de table de hachage distribuée (DHT) ont connu une grande popularité en termes de performance. Ils ont connu de multiples améliorations pour augmenter l'efficacité, comme le mécanisme de réplication qui sert à améliorer la disponibilité des données et l'adaptation à la topologie dynamique. Outre ces avantages, ces méthodes souffrent de trafic excessif généré lors du processus de maintenance des données répliquées. Le déploiement de l'overlay DHT au réseau mobile ad-hoc, bénéficie certes de l'architecture sans infrastructure, mais présente certaines lacunes en raison de la bande passante et l'énergie limitée, ce qui nécessite une réduction d'overhead. C'est dans ce contexte que s'inscrit l'objectif de ce travail, avec en prime l'amélioration de l'efficacité de recherche de Chord à base de DHT sur un underlay de réseau ad-hoc mobile. Pour cela, nous proposons un mécanisme de réplication basé sur une nouvelle formule qui détermine l'emplacement des nœuds de répliques ; un processus de recherche qui vise à réduire le chemin de recherche et un mécanisme de maintenance qui s'adapte à la topologie dynamique tout en évitant l'overhead excessif. En conclusion, ce travail permet d'améliorer les performances de la recherche dans CHORD sans générer de trafic excessif, pour un Chord mobile plus efficace en énergie. La méthode proposée est évaluée suite à une étude de simulation étendue.

Mots clés : Système pair à pair; Chord; DHT « table de hachage distribuée »; overhead; réseau ad hoc mobile.

Table des matières

Tables des figures

Liste des tableaux

Introduction générale

Contexte

Dès les débuts d'internet, le paradigme client-serveur a envahi le domaine de partage de fichiers : principe de centralisation de données dans une unité appelée « serveur », accessibles à partir d'unités appelées « clients » pour accéder et récupérer des données à la demande. Dès lors, l'augmentation d'utilisateurs a exigé plus de ressources à partager et donc plus de serveurs, afin d'assurer la disponibilité des services aux nombreuses demandes traitées séquentiellement ou encore mieux, de façon simultanées; ce qu'on appelle le passage à l'échelle. Dans ce contexte, le modèle client-serveur a rapidement montré ses limites, d'où la nécessité de systèmes qui possèdent de fortes ressources cumulées et une multitude de services. Le paradigme pair-à-pair (de l'anglais : peer to peer) est appelé à lever toutes ces contraintes et apporter des réponses temps réel.

Le principe du pair-à-pair (P2P) consiste à éliminer la notion de centralisation de serveur et à distribuer les services à travers tous les pairs participants. Autrement dit, le P2P est un système distribué contenant des pairs consommateurs et fournisseurs à la fois, construisant un réseau logique appelé ***overlay*** au-dessus d'un réseau physique appelé ***underlay***. Bien que, les premières applications de P2P s'intéressent exclusivement au partage des fichiers, ils ont été développés après, afin d'être utilisés dans d'autres domaines tels que la communication en temps réel, le calcul distribué, les moteurs de recherche ou encore le domaine des jeux.

Le P2P se divise en deux architectures, celle apparue en premier lieu : c'est l'architecture non structurée avec une distribution aléatoire des nœuds à travers l'overlay

et un routage à base d'inondation. Napster [Nap99] a par exemple considérablement popularisé le concept de P2P avec un pourcentage de 49% à 83% du trafic internet. D'autres logiciels P2P se sont succédés, tels que : Gnutella [Rip01], eMule [eMu02], KaZaA [KaZ01], Bittorent [Bit04], pour ne citer que ceux-là. La seconde architecture apparue est bien l'architecture structurée avec des améliorations proéminentes, à base d'une topologie bien définie (anneau, arbre, …) et un protocole de routage explicite.

La majorité des protocoles P2P structurés sont basés sur un système appelé « table de hachage distribuée » de l'anglais « *Distributed hash table* » (DHT), qui utilise une fonction de hachage pour fournir à chaque nœud et à chaque ressource (donnée) un identifiant unique sous un format spécifique, selon lequel ils seront placés sur le système et communiquer ainsi entre eux. En conséquence, la DHT offre plus d'organisation au système, par éviter l'emplacement aléatoire des membres et la propagation aléatoire des requêtes. Parmi les protocoles P2P connus et les plus populaires à base de DHT, nous pouvons citer : Chord [Sto01], Pastry [Row01a], CAN [Rat01], Viceroy [Mal02] et Bamboo [Rhe04].

De l'autre côté, on a l'évolution illimitée des dispositifs mobiles, comme les PDA, les smartphones, les tablettes ou autres dispositifs similaires, en terme de mémoire de stockage, de capacité de calcul ou d'autres fonctionnalités. Par conséquent, ils peuvent être utilisés non seulement pour l'échange des messages et des appels téléphoniques, mais également pour profiter de l'accès à internet via le Wifi, la 3G ou la 4G. Cependant, ces réseaux avec infrastructures ne sont pas toujours disponibles à cause du coût onéreux et de la longue durée de l'installation du système. Ce qui conforte l'importance des réseaux sans-infrastructure, comme les réseaux mobiles ad-hoc (MANET), dans le but de communiquer via le Bluetooth ou le Wifi dans des environnements comme les campus universitaires, les concerts de musique, les cafétérias, ou les zones militaires.

Avec ces progrès, les chercheurs se sont intéressés au déploiement du P2P sur les réseaux mobiles sans-fil en général et sur les MANets en particulier, afin de bénéficier de l'efficacité de partage (localisation et stockage) de données et de la mobilité sans-infrastructure. Les systèmes P2P structurés et les MANet partagent certaines caractéristiques telles que :

- La distribution, où chaque nœud est un serveur et un client au même temps.
- L'auto-organisation face aux arrivées et aux départs imprévisibles des nœuds (topologie dynamique).
- L'auto-guérison contre les défaillances des nœuds.
- Les exigences de base sont la montée à l'échelle et l'efficacité de routage.

En outre, comme le P2P a été destiné en premier lieu aux réseaux fixes filaires, il présente quelques défis sur le MANET qui doivent être pris impérativement en considération, comme l'énergie, la bande passante limitée et la topologie hautement dynamique.

Dans les protocoles traditionnels à base de DHT, chaque donnée (appelé également ressource, item ou objet) est destinée à être stockée par un nœud selon les valeurs d'identifiants les plus proches. Ainsi, parmi les améliorations appliquées sur ces protocoles, on trouve la notion de réplication. Répliquer les données dans plus d'un nœud (nœud de réplique), permet d'augmenter la disponibilité de données, de réaliser l'équilibrage de charge et de réduire la latence de recherche. En plus de ces avantages, la réplication de données génère plus de trafic réseau dans les processus de mise à jour et de maintenance, quand les nœuds de répliques mettent à jour leurs répliques (les données répliquées) à chaque fois qu'un groupe de nœuds est changé par l'arrivée ou le départ d'un nœud. Le nombre de messages de maintenance dépend de la méthode de réplication utilisée, mais le trafic excessif n'est pas adaptable aux dispositifs mobiles avec une bande passante limitée et une énergie restreinte.

Contributions

Dans cette thèse, nous nous sommes intéressés à l'adaptation du protocole P2P à base de DHT qui est le protocole Chord au réseau mobile sans-fil MANET. On voudrait améliorer les performances de la recherche dans Chord sans générer un trafic excessif et ainsi optimiser la consommation d'énergie des nœuds. On s'est principalement focalisé sur les deux importants aspects des DHT qui sont : l'emplacement des données dans le réseau logique et la recherche des données dans ce dernier.

Dans ce travail, nous proposons une approche de réplication dans le protocole structuré CHORD au-dessus de MANET, appelée RepMChord, afin d'augmenter la disponibilité de données et de réduire la latence de recherche. Ceci doit être fait en évitant le problème le plus connu des méthodes de réplications qui est l'augmentation de trafic généré, dans le but de considérer et optimiser la consommation d'énergie des dispositifs mobiles et donc augmenter la durée de vie de réseau. Le travail est basé sur une nouvelle formule pour déterminer l'emplacement de ressources. On étudie et on évalue cinq processus pour assurer l'efficacité de l'approche, qui sont : le stockage d'une ressource, l'arrivée d'un nouveau nœud, le départ d'un nœud, le processus de recherche et la mobilité du nœud.

Déterminer un processus de recherche efficace est l'un des principaux objectifs de tout travail à base de DHT. La recherche présente la fonction de base des protocoles DHT. Dans ce présent travail, nous ne suivons pas le processus de recherche traditionnel de CHORD, mais nous essayons d'apporter quelques modifications, dans le but de mieux bénéficier de la stratégie de réplication proposée.

Ce travail diffère des autres approches de réplication existantes dans :

- La prise en charge et la mise à jour du trafic de maintenance
- La réduction de la consommation de l'énergie,
- L'application d'une méthode de réplication sur un CHORD mobile.

A notre humble connaissance, la tentative d'utilisation de la réplication pour améliorer la consommation de l'énergie dans un protocole P2P mobile est une des toutes premières tentatives dans cette thématique de recherche.

Organisation du document

Ce manuscrit s'articule autour de quatre chapitres, précédés d'une introduction générale et se termine avec une conclusion générale.

Le premier chapitre est consacré à la présentation des systèmes P2P. Des concepts du pair à pair et leurs applications constituent ce chapitre. Les différentes architectures des P2P avec leurs caractéristiques sont également bien élucidées. Les systèmes P2P à base de table de hachage distribuée (DHT) aux quelles, nous nous intéressons sont étudiés. Le principe de DHT et les différentes topologies DHT existantes suivis de quelques protocoles connus dans la littérature, sont également décrits.

Le deuxième chapitre introduit les réseaux sans-fil mobiles en général et les MANET en particulier. Il mentionne les protocoles de routage et explique les contraintes de l'énergie. Les techniques de conservation de l'énergie dans MANet, au niveau du routage, des couches MAC et physique sont expliquées. La partie DHT-mobile, est introduite et énumère un ensemble de problèmes intéressants à aborder. Les travaux d'optimisation de DHT fixe et mobile, existants, sont étudiés et catégorisés selon notre propre classification. Un tableau comparatif d'un ensemble de travaux effectué selon les catégories à base d'un certain nombre de critères est largement commenté.

Le troisième chapitre est consacré à la présentation de nos contributions avec la signalisation de l'emplacement des ressources sur les nœuds de l'overlay et la description de l'heuristique de réplication de donnée. La démarche au niveau des processus est bien explicitée: arrivée du nouveau nœud, départ d'un nœud et gestion de la mobilité des nœuds. Les modifications proposées au niveau du processus de recherche pour tirer profit de la nouvelle méthode de stockage de ressources et d'assurer l'efficacité de la méthode constitue l'essentiel de ce chapitre.

Le quatrième chapitre, dresse les performances de notre stratégie. La validation et la mise en œuvre, s'appuient sur le simulateur réseau Omnet++ [Var08] et les deux plateformes Oversim [Bau09] et Inet [Ine12]. Les métriques de performances prouvant l'efficacité de nos résultats comparés aux résultats obtenus à l'aide de Chord et à ceux d'un autre travail récent, connu pour ses performances au niveau de processus de recherche, sont élaborées. Les résultats obtenus sont présentés sous forme des graphes d'évaluation et ont fait objet d'une judicieuse comparaison avec les deux travaux mentionnés ci-dessus.

Ce mémoire s'achève par une conclusion générale, reprenant les points forts de cette thèse suivis des perspectives futures suggérées.

Chapitre 1

Les systèmes pair-à-pair (P2P)

1. Introduction

L'évolution de l'internet exige le développement et l'accessibilité à un maximum de ressources, à travers des systèmes ouverts et à grande échelle. Dans ce contexte, le modèle client-serveur classique a rapidement atteint ses limites. Avec une seule machine contenant les ressources et plusieurs programmes y accédant, on a forcément une sous-exploitation de ressources. Dans l'optique d'éviter ces limites, le modèle pair-à-pair est né et c'est dans ce contexte que s'inscrit notre travail.

Dans ce chapitre, nous allons fournir quelques définitions de P2P avec quelques caractéristiques principales. Puis, nous allons présenter l'application de base pour laquelle le P2P a été conçu, et quelques applications de recherche qui s'apparentent à ces technologies. Une présentation des différentes architectures P2P, suivies de leurs avantages et de leurs inconvénients font l'objet d'une étude par la suite. La partie des systèmes P2P à base de table de hachage distribuées (DHT) est élucidée et ce en présentant le principe de DHT et ses propriétés. Nous décrivons d'une façon succincte les différentes topologies DHT existantes en mentionnant quelques-uns des protocoles les plus connus dans la littérature.

2. Définitions des systèmes pair-à-pair

Un réseau pair-à-pair (de l'anglais peer-to-peer : P2P) est un réseau virtuel logique appelé ***Overlay*** construit sur un réseau réel physique appelé ***Underlay***, à travers lequel les pairs (les nœuds) participants sont organisés d'une manière distribuée sans hiérarchie et sans contrôle centralisé.

Le système P2P ne respecte pas le paradigme Client/Serveur traditionnel, à cause des pairs, dont chacun d'eux joue le rôle d'un client et d'un serveur en même temps. Ils communiquent entre eux pour établir des structures d'overlay, auto-organisées au sommet des réseaux physiques sous-jacents dans le but de manipuler et de soutenir la variété du niveau d'application (services et applications).

L'absence du contrôle central apporte au système des avantages ainsi que des contraintes. L'avantage est d'augmenter l'évolutivité du système dont les charges sont réparties sur tous les nœuds de même priorité et ne résidant pas sur une seule entité, ce qui permet l'adaptation à la grande échelle et la réalisation d'un système robuste par éviter le seul point d'échec. Il accélère

ainsi l'accomplissement des taches en réduisant le temps de traitement à travers les liens directs entre les pairs. Comme contrainte, le système dont la structure est distribuée est difficile à administrer et la connaissance globale de l'état des ressources et du réseau est impossible, ce qui nécessite la mise en place d'un mécanisme de stabilisation (auto-organisation), c'est-à-dire de continuer à fonctionner et assurer la disponibilité des ressources même avec la déconnexion de quelques nœuds.

3. Les applications P2P

Les systèmes P2P ont été utilisés dans plusieurs domaines depuis leur apparition [And04] et on peut les classifier dans les cinq catégories :

3.1. Partage de fichiers

C'est l'application de base pour laquelle le système P2P a été conçu. Elle consiste à permettre aux utilisateurs de fournir des fichiers et des ressources au réseau, ainsi de pouvoir localiser et accéder aux contenus dans d'autres pairs. Son mécanisme est de permettre la livraison de contenu d'un nœud fournisseur (***requested***) à un nœud demandeur (***requester***) sans passer par un serveur central. Par exemple, l'application très connue BitTorent, ainsi que le streaming qui est considéré comme une application de diffusion de contenu (video/audio) en temps réel.

3.2. Communication en temps réel

Cette communication a pour but de permettre l'échange de messages, de vidéo ou d'audio en temps réel. Autrement dit, permettre aux utilisateurs de collaborer en ligne pour se parler et/ou pour se voir. Par exemple, Skype fournit un service de téléphonie internet à base de P2P, et Fring pour la messagerie instantanée et la voix IP (VoIP).

3.3. Calcul distribué

Il permet de répartir un gros calcul sur plusieurs pairs, dont chacun d'eux travaille sur une partie (résultat partiel) et à la fin ils collaborent pour donner le résultat final. Ce mécanisme permet aux pairs de tirer profit des ressources non disponibles chez eux et qui se trouvent sur d'autres pairs, ce qui permet ainsi d'accélérer la tache de calcul.

3.4. Moteur de recherche

La recherche à base de P2P est distribuée, tel que le pair demandeur (requester) n'envoie pas la requête de recherche au serveur central, mais il transmet la requête à ses voisins qui à leur tour les transmettent à leurs voisins. Le pair qui reçoit la requête cherche dans son propre système et répond au nœud source.

3.5. Gaming

Gaming permet de jouer et d'entrer en compétition avec d'autres pairs sur le réseau P2P dans un environnement virtuel. Le pair lui-même doit contrôler l'état de son jeu et de faire intervenir les autres joueurs pour des changements ou d'autres actions importantes.

4. Les architectures P2P

Les architectures P2P se distinguent selon le niveau de centralisation de l'index de données. Elles peuvent être classifiées en 3 grandes familles : centralisée, décentralisée et partiellement centralisée.

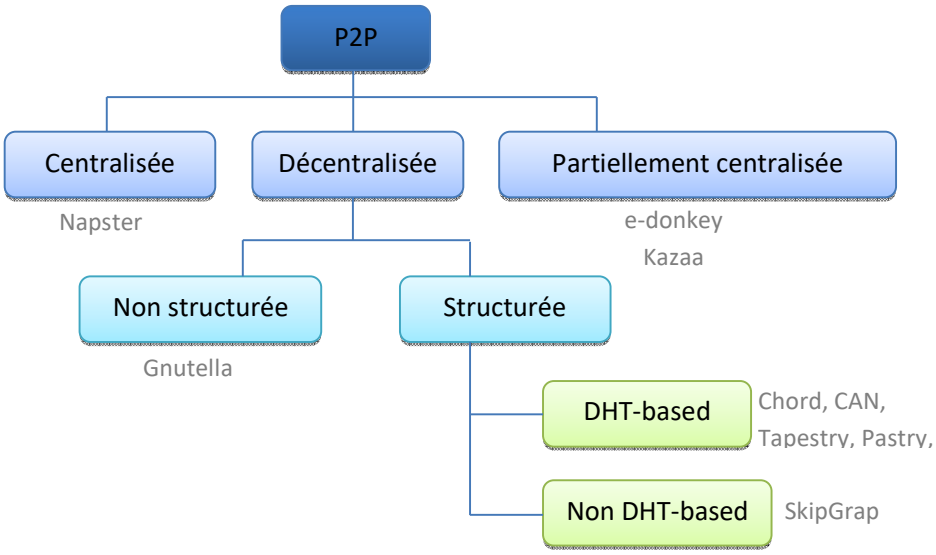

Figure1.1 Classification des systèmes P2P

4.1. L'architecture centralisée

C'est la première architecture conçue avec l'apparition du premier protocole P2P Napster [Nap99]. Elle utilise un serveur d'index central servant d'annuaire, dans lequel les pairs du

réseau mettent des métadonnées sur les ressources qu'ils partagent (ex : nom de fichier) ainsi que des informations sur eux même (ex : adresses IP, numéro de port,…). Ensuite, si un pair est intéressé par l'une de ces ressources, il contacte le fournisseur pour un échange direct, et c'est ce point-là qui distingue cette architecture de celle de client-serveur.

Cette architecture a comme avantage, la facilite de la recherche en accédant directement au bon serveur d'index (l'annuaire), et elle réduit le trafic réseau puisque les nœuds ne se communiquent seulement s'ils ont des informations à échanger.

L'inconvénient de cette architecture est toujours le problème de la centralisation avec le seul point de défaillance, ainsi que l'absence de l'anonymat avec l'identification de chaque fournisseur sur le serveur.

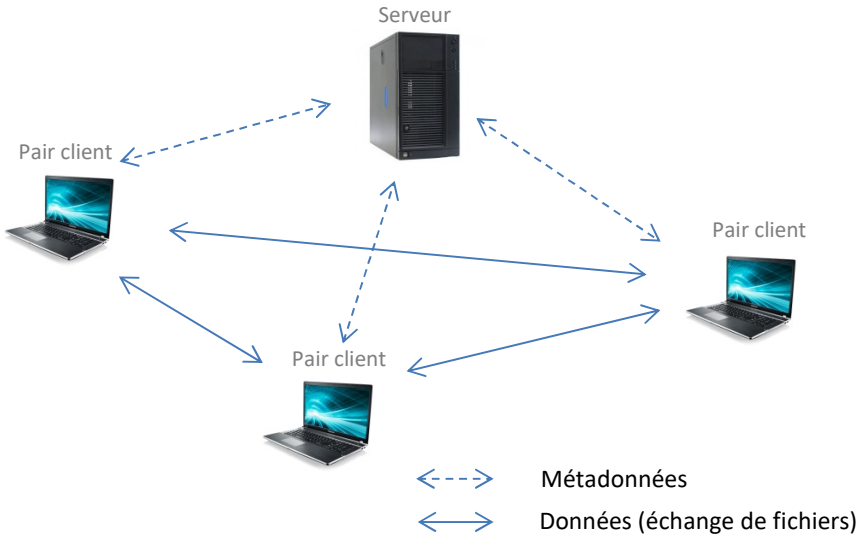

Figure1.2 Architecture P2P centralisée

4.2. L'architecture décentralisée

Elle est appelée ainsi l'architecture P2P pure, où on élimine tout point de centralisation et l'index devient local sur chaque pair. Elle a l'avantage d'éviter le seul point de défaillance mais cela rajoute des contraintes supplémentaires lors de la recherche de l'information. Cette architecture elle-même est divisée en deux catégories :

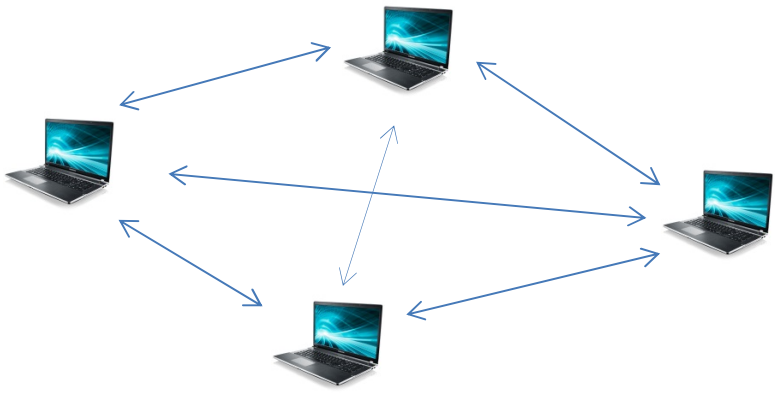

Figure1.3 Architecture P2P décentralisée

4.2.1. P2P non-structurée

Elle est non-structurée à cause de l'absence d'une contrainte topologique définissant la répartition des pairs et des ressources sur le système, où chaque pair a des informations uniquement sur ses propres ressources et aucune sur celles des autres pairs. Il choisit ses voisins d'une façon aléatoire ; un exemple de protocole basé sur cette architecture est Gnutella, dans sa première version [Rip01]. Pour effectuer une recherche, le pair utilise la méthode d'inondation pour envoyer une requête à ses voisins et eux-mêmes la transmettent à leurs voisins, en limitant le nombre de sauts par le TTL (Time to live) qui est typiquement 7 pour Gnutella. Ensuite, le possesseur de la ressource répond au *requester* et l'échange se fait d'une manière directe.

L'inconvénient de cette architecture est le grand trafic généré par l'inondation ainsi que la faible efficacité de la recherche causée par le TTL où la recherche peut s'arrêter avant d'arriver au nœud possesseur.

4.2.2. P2P structuré

Dans le but de résoudre les problèmes posés par les architectures non-structurées, Cette architecture suit une contrainte topologique qui définit le placement des pairs et des ressources sur l'overlay P2P. Afin d'offrir une organisation au système, par éviter la répartition aléatoire des membres, et utiliser un algorithme de routage bien défini au lieu de celui de l'inondation.

La majorité des protocoles P2P utilisant cette architecture, sont basés sur un système appelé DHT. Ce type d'architecture suivi de ses différentes topologies sera présenté dans la suite.

4.3. L'architecture partiellement centralisée

On l'appelle également architecture hybride avec de nombreux serveurs. Elle fait une combinaison entre les deux architectures précédentes de manière à bénéficier des avantages des deux architectures. Ici, les pairs ne sont plus égaux comme dans l'architecture décentralisée, mais à l'addition des pairs ordinaires il existe aussi, ce qu'on appelle les supers-pairs, dont chacun d'eux joue le rôle d'un serveur d'indexe pour une partie de l'overlay (cluster) et non, pour la totalité du réseau comme dans l'architecture centralisée. Ces super-pairs sont choisis généralement selon le critère de la bande passante, la puissance de calcul ou la capacité de stockage.

Un exemple des protocoles qui suivent cette architecture : e-Donkey [Hec04] avec une définition des super-pairs dès le début de la construction de système (pré-définition), et le protocole kazaa avec la possibilité de définir des super-pairs à tout moment selon certaines conditions.

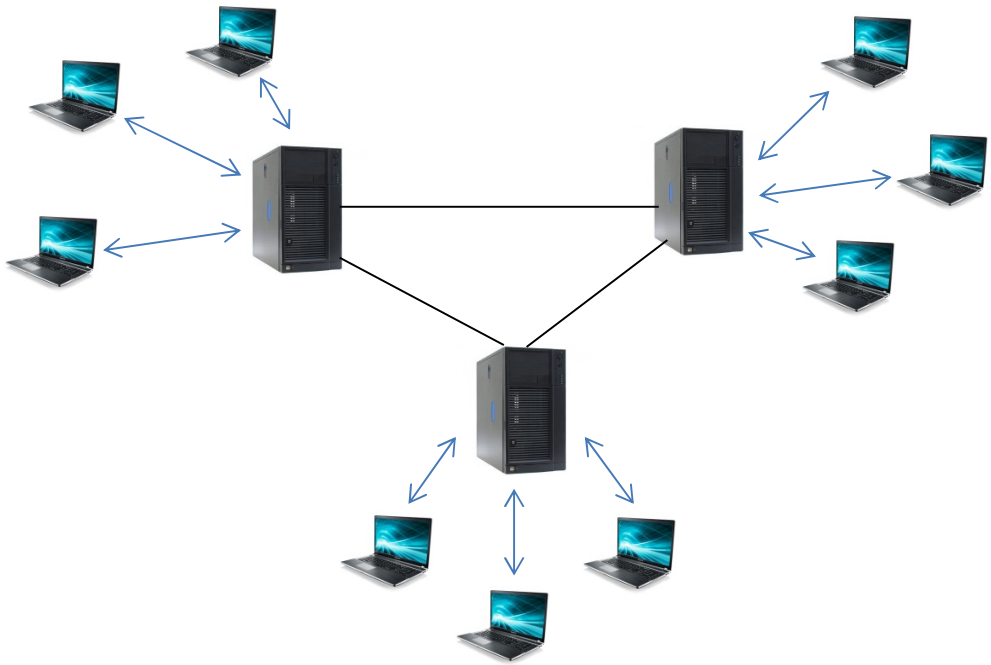

Figure1.4 Architecture P2P partiellement centralisée (Hybride)

5. Table de hachage distribuée

Afin d'éviter les problèmes posés par l'inondation, utilisée par les systèmes P2P non-structurés, la propagation aléatoire des requêtes entre pairs a été remplacée dans les P2P structurés par un système appelé DHT « Table de Hachage Distribuée » pour fournir un routage efficace entre pairs et assurer l'efficacité de la recherche.

5.1. Principe

La DHT utilise une fonction de hachage possédant de bonnes propriétés. Au début, c'était la SHA-1 (*Secure Hash Algorithm*) qui permet de convertir une suite d'octets d'un maximum de 2^{64} bits à un nombre de 160 bits, et donc de distinguer 2^{160} données différentes, avec un niveau de sécurité élevé pour rendre très difficile et presque impossible l'obtention des collisions (en 2^{63} opérations).

L'utilisation de SHA-1 permet aux DHTs de définir un identifiant unique sous un format spécifique, à chaque nœud et à chaque ressource pour être placés dans le système P2P. On identifie le nœud par le hachage de son adresse IP et on appelle le résultat un ***ID*** et on identifie la ressource par le hachage de son nom qu'on appelle une clé (***key***).

L'idée est de répartir l'espace de clés en zones, et assigner un nœud responsable à chaque zone selon la proximité des identifiants et des clés, c'est-à-dire de choisir le nœud avec l'identifiant le plus proche des clés de la zone. La DHT stocke ces informations sous format de couples (clé, valeur) répertoriés par tous les nœuds du système, de telle façon que la déconnexion d'un utilisateur n'entraîne pas de pertes importantes pour l'ensemble du réseau. De plus, la DHT permet de gérer un important nombre d'utilisateurs, ainsi que l'arrivée de nouveaux nœuds, le départ d'anciens ou encore les erreurs de certains utilisateurs.

La DHT est utilisée dans le système de stockage de données distribuées et dans le système de localisation de données. Elle assigne un ID unique à chaque nœud et une *clé* unique à chaque ressource, les deux sont de longueur de *m* bits.

Les opérations de base réalisables sur les DHT sont les opérations :

- **Lookup (clé)** : permet de récupérer la valeur associée à *clé*, dont la valeur ici peut présenter l'ID du nœud (ou des nœuds) qui possède ou stocke réellement le fichier ayant l'identifiant ***clé***.
- **Store (clé, valeur)** : permet de stocker une clé et sa valeur associée dans la table de hachage distribuée.

5.2. Propriétés

Les DHTs ont montré leurs performances sur les systèmes P2P, en se basant sur des systèmes purement décentralisés dont le nœud possède des informations locales sur quelques nœuds proches et pas de connaissance globale. Les DHTs présentent les propriétés suivantes :

- **Passage à l'échelle et performance :** la structure de la table de DHT permet de placer un très grand nombre de pairs et de ressources dans le système tout en gardant une taille raisonnable de la table, et de permettre ainsi aux utilisateurs de trouver le responsable d'une clé donnée dans une période constante ou logarithmique quel que soit la taille du réseau. Ce qui leur permet de s'adapter à la grande échelle tout en assurant la performance de système.
- **Petit diamètre logique:** en utilisant la DHT, le pair choisit le nœud le plus proche logiquement de la destination pour lui transmettre la requête. Dans la plupart des cas, le nombre de sauts logiques parcourus s'exprime par $O(\log_2 N)$, tel que N est la taille du réseau P2P.
- **Tolérance aux fautes:** l'absence de centralisation permet d'éviter le problème de point de défaillance, de telle façon que le système peut continuer à fonctionner même avec le départ de quelques nœuds.
- **Fiabilité:** ici on parle de la fiabilité de la recherche assurée par les systèmes DHT**.** A chaque envoi d'une requête de recherche, le nœud *requester* reçoit une réponse indiquant le nœud *requested* proche. Si ce n'est pas le cas, c'est seulement lorsque la ressource n'existe pas dans le système. C.-à-d. si la ressource existe dans le système on va surement la trouver.

6. Les topologies DHT existantes

Dans la littérature ils existent plusieurs algorithmes et protocoles P2P qui se basent sur la DHT, mais chacun a sa propre topologie logique, et donc on peut les classifier selon le graphe logique utilisé.

Les DHT s'inspirent des graphes, et on peut classifier ses protocoles selon l'architecture de graphe utilisée, comme CAN [Rat01] qui utilise une topologie hypercube, Tapestry [Zha01], Pastry [Row01a] et Bamboo [Rhe04] sous une topologie en arbre basée sur l'algorithme de plaxton [Pla99], et on a le protocole Chord [Sto01] à base d'une topologie en anneau. C'est à ce dernier protocole que l'on s'intéresse particulièrement, que nous allons décrire ci-dessous.

6.1. Topologie en anneau

C'est une topologie facile à gérer grâce à sa géométrie circulaire, elle est utilisée par plusieurs protocoles P2P d'une façon directe ou indirecte, comme le protocole Pastry qui n'est pas principalement basé sur cette topologie, mais il l'utilise pour finaliser le routage. Le protocole Chord, lui, par contre, suit une topologie annulaire d'une façon directe et principale, où l'espace d'identification est circulaire et la proximité est numérique.

6.1.1. Chord

6.1.1.1. Vision globale

Chord [Sto01] fournit un protocole évolutif et efficace pour la recherche dynamique, dans les systèmes P2P avec de fréquents arrivées et départs de nœuds. Il spécifie comment localiser une ressource, comment le nouveau arrivé joint le système et comment gérer les défaillances.

Chord dispose d'une topologie simple, composée d'un seul anneau de 2^m nœuds, placés sur l'anneau dans l'ordre croissant de leurs IDs, avec des liens directionnels dans le sens horaire (***Clockwise direction***), dont m est l'espace d'adressage. Chaque ressource et chaque nœud possède un ID logique de longueur m bits, défini par une fonction de hachage (ex. SHA-1).

- L'identifiant du nœud (ID) = hacher (l'adresse IP)
- L'identifiant de ressource (*key*) = hacher (le nom de fichier)

Pour améliorer l'évolutivité, le nœud Chord n'a pas besoin d'avoir des informations de routage sur tous les autres nœuds de système mais sur seulement un nombre de $O(\log_2 N)$ d'autres nœuds ainsi la recherche nécessite un max de O(log N) messages.

6.1.1.2. Espace de stockage

La clé ***k*** est assignée au nœud avec un ID supérieur ou égal à *k* dans l'espace d'identifiants. Ce nœud est appelé le successeur de *k*, noté *successeur(k)*. Comme dans la figure1.5, le nœud successeur de la clé 2 est le nœud avec l'ID 3, sur lequel elle sera stockée. Par conséquent, chaque nœud est responsable de quelques clés formant sa *zone de responsabilité*.

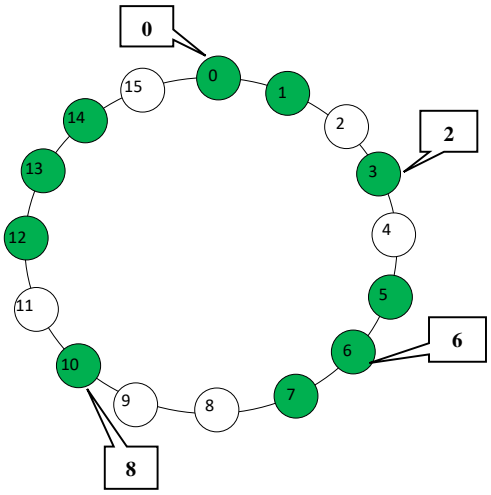

Figure1.5 Anneau Chord à 10 nœuds et 4 clés

Pour maintenir l'espace de stockage, lorsqu'un nœud *n* rejoint le réseau, certaines clés assignées au successeur de *n* deviennent assignées à *n*. Lorsqu'un nœud *n* quitte le réseau toutes ses clés seront assignées à son nœud successeur.

6.1.1.3. Recherche naïve

C'est une recherche simple mais lente, dont chaque nœud nécessite seulement de savoir comment contacter son nœud successeur dans l'espace d'identifiant. La requête de recherche passe d'un successeur à un autre jusqu'à arriver au pair qui contient la ressource, et ça résulte une recherche linéaire en nombre de nœuds.

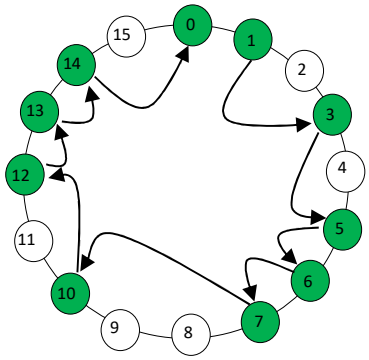

Figure1.6 Recherche naïve : le nœud N1 cherche la clé = 0 qui se trouve chez N0

6.1.1.4. Recherche améliorée

Pour accélérer le protocole de recherche, Chord maintient pour chaque nœud une table de routage appelée une table de ***fingers*** de *m* entrées, dont $m=\log_2 N$. La i^{eme} entrée dans la table du nœud *n* contient l'ID du premier nœud *s* qui suit *n* par au moins 2^{i-1} nœuds dans l'anneau d'identifiants, c'est-à-dire s=successeur $((n+2^{i-1}) \bmod m)$, tel que *i* est entre 0 et *m,* et *m* égal à l'espace d'adressage de la DHT utilisée et égal au logarithme ($\log_2 N$) de la taille réseau N.

Lorsqu'un nœud souhaite trouver une clé, il utilise l'algorithme suivant :

Si: la clé est entre le nœud de départ et son successeur alors la clé se trouve chez ce successeur.
Sinon: il cherche parmi ses fingers le nœud dont l'identifiant est le plus grand tout en étant inférieur à la clé, et lui transmet le message. Le nœud qui reçoit le message exécute à son tour cet algorithme.

Dans la figure1.7, le nœud avec ID=7 cherche le successeur de la clé 1. La table de fingers du nœud 7 montre que le nœud avec le plus grand ID et qui précède la clé 1 est le nœud d'ID=0, donc le nœud 7 lui transmet la requête. A son tour, le nœud 0 trouve que son successeur est le nœud cible qui contient la clé 1, il lui renvoie la requête.

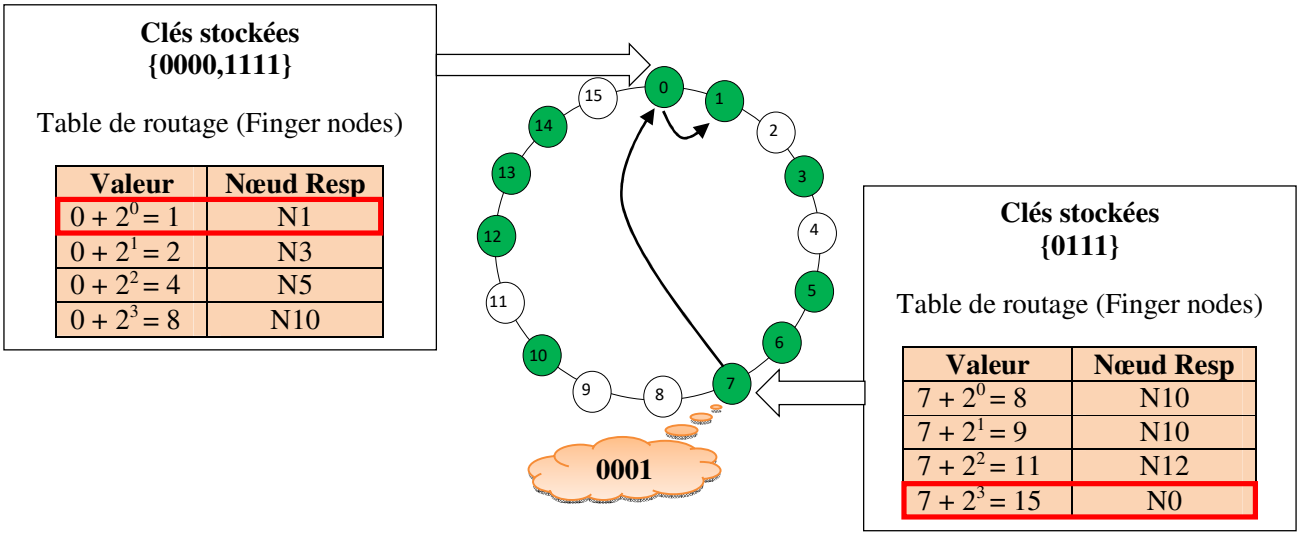

Figure1.7 Recherche améliorée : le nœud avec ID=7 cherche la clé=1 qui se trouve chez le nœud avec ID=1.

6.1.1.5. Arrivée d'un nœud

Pour l'insertion d'un nouveau nœud dans l'anneau :

- le nœud *n* est inséré entre le nœud d'identifiant immédiatement inférieur et celui immédiatement supérieur (son prédécesseur p(n) et son successeur s(n) sur l'anneau). Le nœud *n* gère alors les clés de valeur comprises entre p(n)+1 et *n*. Lors de son insertion, le nœud *n* récupère alors la gestion d'une partie de l'espace des clés associées à son successeur.
- L'insertion d'un nœud au réseau nécessite la mise à jour des tables de fingers.

6.2. Topologie en arbre basée sur Plaxton

Cette famille se base sur l'algorithme de Plaxton [Pla99], qui se charge à la fois de la localisation des objets et le routage dans des environnements logiques décentralisés, dont l'identification est définie d'une manière aléatoire en utilisant une fonction de hachage. Les tables de routage réparties sur les pairs sont de taille fixe. Le routage est effectué selon la méthode hyper-cube à base de préfixes, où on se rapproche successivement d'un seul digit à la fois dans l'ID numérique, jusqu'à atteindre la destination. Parmi les protocoles P2P qui se basent sur cette topologie, nous mentionnons Tapestry, Bamboo et Pastry qui est explicité ci-dessous.

6.2.1. Pastry [Row01a]

C'est un protocole de localisation et de routage, il associe la clé au pair avec l'identifiant le plus proche numériquement. Pour le routage, il utilise dans une première partie l'algorithme de Plaxton ensuite il utilise les voisins virtuels répartis selon un cercle non-orienté contrairement à Chord. Les identifiants des pairs et de ressources sont de 128 bits, en base 2^b et la valeur de b est souvent 2 ou 4.

Chaque pair possède une table de routage de taille fixe de $\log_{2^b} N$ lignes et 2^{b-1} colonnes (Figure1.8). Elle est constituée de trois parties, la première partie qui est « *leaf set* » concerne les voisins logiques avec les identifiants les plus proches numériquement de l'ID du pair actuel, une moitié de cette partie est pour les prédécesseurs et l'autre moitié pour les successeurs. La deuxième partie de la table, consiste en une table de i lignes, dont chaque ligne i contient 2^{b-1} entrées, et la $i^{ème}$ ligne contient un ID qui partage les i premiers digits en commun avec ceux de l'ID du nœud actuel. La troisième partie appelée « neighborhood set », contient les voisins réels du nœud actuel, qui sont les plus proches physiquement selon le nombre de sauts dans le réseau sous-jacent.

Pair **10233102**			
Voisins virtuels			
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232
Table de routage			
-0-2212102	**1**	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	**2**	10-3-23302
102-0-0230	102-1-1302	102-2-2302	**3**
1023-0-322	1023-1-000	1023-2-121	**3**
10233-0-01	**1**	10233-2-32	
0		102331-2-0	
		2	
Voisins réels			
13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Figure1.8 Extrait de [Row01a] : Table de routage Pastry d'un pair d'identifiant 10233102 avec b=2.

Lors d'une recherche d'une clé *key*, le nœud commence par chercher dans son *leafset* le nœud responsable de *key*, s'il le trouve il lui transmet la requête. Sinon, il vérifie le nombre *l* de digits de son ID qui sont en commun avec les digits de *key*, puis il cherche dans sa table de routage l'ID qui partage un nombre *c* de digits en commun avec *key* tel qu'il est plus grand que *l*, et effectue un routage de Plaxton. Ensuite, s'il ne trouve pas ce nœud dans la table, il cherche dans l'ensemble de voisinage (neighborhood-set) le nœud avec l'ID le plus proche à *key,* qui partage un nombre de digit plus grand que *l* en commun avec *key*. Cet algorithme est répété jusqu'à ce qu'il n'existe plus de nœuds dans la table, plus proche à *key*. Pour le meilleur de notre connaissance, le nombre de sauts parcourus par une requête Pastry s'exprime par O ($\log_{2^b} N$).

6.3. Topologie en hypercube

Parmi les protocoles connus de cette catégorie est CAN [Rat01] car il peut être vu comme un cube de *d* dimensions à coordonnées cartésiennes. Il a une architecture torique à un espace multidimensionnel différemment à Chord qui a un espace circulaire à une seule dimension.

L'espace de CAN est réparti en plusieurs zones, et chaque pair est responsable d'une zone. Pour placer les ressources sur cet espace, on applique une fonction de hachage sur l'identifiant de la ressource pour obtenir ses coordonnées, selon lesquelles elle sera placée dans une zone de l'espace et associée au pair responsable de cette dernière.

Chaque pair possède une table de routage dans laquelle se trouvent des informations sur ses voisins, qui sont les responsables des zones voisines dans l'espace logique. Comme dans l'exemple de la figure1.9, le pair A a comme voisins B et C. Le nombre de voisins dans l'espace à *d* dimensions est *2d*.

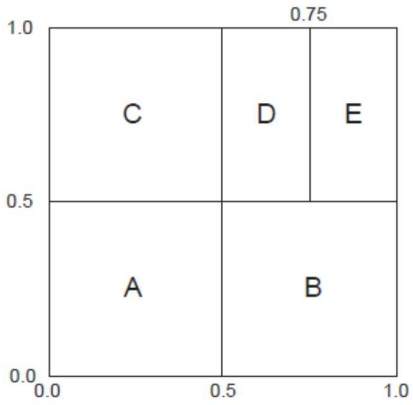

Figure1.9 Extrait de [Rat01]. Exemple d'espace CAN à deux dimensions avec 5 pairs

Pour la recherche d'une ressource, le *requester* hache le nom de cette ressource pour avoir ses coordonnées et connaitre son emplacement et son nœud responsable. Il effectue ensuite, un acheminement de proche en proche passant par les voisins qui présentent des coordonnées les plus proches du nœud cible. L'augmentation de la taille « N » du réseau n'affecte pas le nombre de voisins par nœud, mais elle affecte la longueur moyenne du chemin emprunté par une requête, qui s'exprime par O ($N^{1/d}$).

6.4. Topologie en papillon

6.4.1. Viceroy

Viceroy [Mal02] est un protocole de routage P2P, dite en papillon, car les liens reliant les nœuds dans l'overlay présentent une forme en papillon, comme dans la figure1.10. Viceroy s'inspire de Chord, avec une architecture multi-anneau (multi-niveau), dont les indices des niveaux se trouvent dans l'ordre croissant. Viceroy contient log_2N anneaux, dont chacun comporte $\frac{N}{log_2 N}$ nœuds.

Chaque pair est connecté à un nombre constant de voisins (typiquement 7). Il a deux liens généraux avec le successeur et le prédécesseur dans le même anneau, deux liens de niveau avec le prédécesseur et le successeur sur l'anneau de niveau 1, et trois liens ***butterfly*** de longues distances avec des nœuds d'autres niveaux.

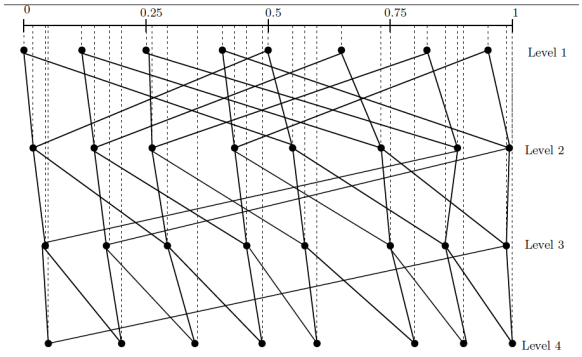

Figure1.10 Extrait de [Mal02]. Un réseau de Viceroy idéal.

Le routage de la requête de recherche, commence par le niveau actuel, puis la requête remonte vers le niveau le plus supérieur, ensuite elle descend vers le niveau le plus bas en s'approchant de la *clé*, jusqu'à satisfaction de la requête.

Ulysses [Kum04] est un protocole de routage P2P d'une architecture en papillon, qui propose une amélioration de Viceroy, avec l'ajout des liens supplémentaires vers des nœuds d'autres niveaux lorsque les autres liens sont congestionnés (à cause de la dynamicité de réseau), afin de mieux résister au *churn*. Chaque nœud maintient O(log N) voisins et le routage est réalisé en $O(\frac{\log N}{\log \log N})$.

6.5. Graphe de Bruijn

Ce graphe [DeB46] est introduit par le mathématicien Bruijn. C'est un graphe B(k,n) orienté (Fig1.11), dont chaque nœud présente une suite de longueur *n*, et maintien des liens avec *k* autres nœuds, où *k* est définit selon l'ordre de robustesse désirée.

Le routage dans le graphe de bruijn est basé sur le principe de décalage, en passant d'un nœud à un autre dans la direction des arcs. Pour router vers la clé $x_1x_2\ldots x_n$, on cherche les nœuds avec les suffixes $x_1x_2\ldots x_n$, autrement dit chercher les nœuds qui partagent les mêmes suffixes que les préfixes de la clé. Ce routage se fait en O(log N) pour un graphe de degré constant.

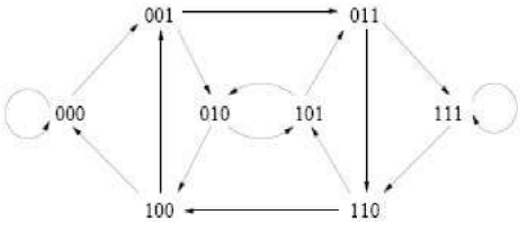

Figure1.11 Exemple de graphe de bruijin, B(2,3).

Parmi les protocoles P2P qui se basent sur le graphe de bruijn, nous citons: Koorde [Kaa03], broose [Gai04] et D2B [Fra06].

7. Conclusion

Ce chapitre dresse un état de l'art sur les systèmes P2P, agencé en deux parties :

- Les définitions du P2P et les différentes architectures existantes en mentionnant les avantages et inconvénients de chacune d'elles.
- Les tables de hachage distribuées et ses différentes caractéristiques.

La seconde partie présente le modèle P2P à base de DHT, dont le concept a été défini et bien explicité. Les différentes topologies DHT existantes, sont étudiées suivies d'une présentation des plus populaires protocoles P2P, correspondants.

Chapitre 2

La DHT mobile et l'énergie

1. Introduction

Le monde informatique a connu une rapide invasion des réseaux mobiles, grâce à l'évolution non limitée, des dispositifs mobiles (ex. PDAs, Smartphones, Tablettes, …). Cette évolution est conséquente en termes de mémoire de stockage, de capacité de calcul et d'autres fonctionnalités. Ils peuvent être utilisés non seulement pour l'échange des messages et des appels téléphoniques mais aussi pour tirer profit des accès à l'Internet via WIFI, la 3G ou la 4G pour les réseaux avec infrastructure, et de communiquer via Bluetooth ou WIFI pour les réseaux sans-infrastructure, comme les réseaux mobiles ad hoc (MANETs).

Suite à ces progrès, les chercheurs se sont intéressés au déploiement de P2P sur les réseaux sans-fil mobiles en général, et sur les MANETs en particulier, afin de bénéficier de l'efficacité de localisation et stockage de données et de la mobilité sans-infrastructure, que nous expliquons dans ce qui suit.

Dans ce chapitre, nous allons commencer par définir les réseaux sans-fil mobiles avec et sans infrastructure. Nous allons décrire les réseaux MANETs d'une façon un peu plus détaillée, en mentionnant ses protocoles de routage proactifs et réactifs, expliquant la contrainte de l'énergie liée à la durée limités des batteries. Les techniques de conservation de l'énergie dans MANETs, au niveau du routage, de la couche MAC et physique, sont également abordées.

Dans la suite, nous allons entamer la partie de déploiement de DHT sur les réseaux mobiles ou sur, ce qu'on appelle (DHT mobile), en donnant une vision globale. Les motivations et les défis de ce concept sont bien élucidés. Puis, nous allons présenter les défis et les problèmes qui restent à aborder par les travaux de DHT fixe et mobile. Ensuite, nous allons présenter les travaux d'optimisation de DHT fixe et mobile qui existent, en commençant par décrire les catégories définies selon notre propre classification. Les travaux définis selon les catégories sont étudiés et largement discutés. A la fin de ce chapitre, des tableaux comparatifs de ces travaux catégorisés selon certains critères jugés opportuns sont présentés.

2. Les réseaux mobiles sans-fil

Un réseau mobile est un réseau informatique composé d'unités mobiles (PDA, tablette, PC-portable, Smartphone, …), possédant toutes des caractéristiques différentes (indépendance,

capacité de stockage, vitesse de calcul, …). Elles se connectent à travers une interface de communications sans-fil. Cette technologie se base sur des ondes radio pour transmettre les messages via le réseau au lieu d'utiliser des câbles physiques. Elle offre une grande flexibilité d'emploi et de mise en réseau des unités dont le câblage serait pénible et coûteux, voir même impossible. Les réseaux mobiles ou sans-fil, peuvent être classés en deux catégories, avec infrastructure et sans infrastructure. La première classe concerne les réseaux où l'acheminement des messages entre les unités mobiles passent par une (ou plusieurs) unités fixes, appelée station de base (SB) ou station de support mobile (Mobile Support Station), qui possède une interface sans fil pour communiquer avec les unités mobiles situées dans une zone géographique limitée, appelée cellule. Par ailleurs, ces unités fixes se connectent entre elles, via un réseau de communication filaire, généralement fiable et avec un débit élevé, contrairement aux communications sans fil avec une bande passante limitée qui réduit le nombre d'informations échangées. Un bon exemple de réseau correspondant à cette classe, est bien le réseau cellulaire, dont chaque nœud se trouve à un moment donné connecté à une seule SB, soit, celle qui se retrouve dans sa zone géographique. Cette station est utilisée pour communiquer avec d'autres unités mobiles correspondant à la même SB ou à une autre station. Si le nœud se déplace et rejoint une nouvelle zone, il sera connecté à une nouvelle SB.

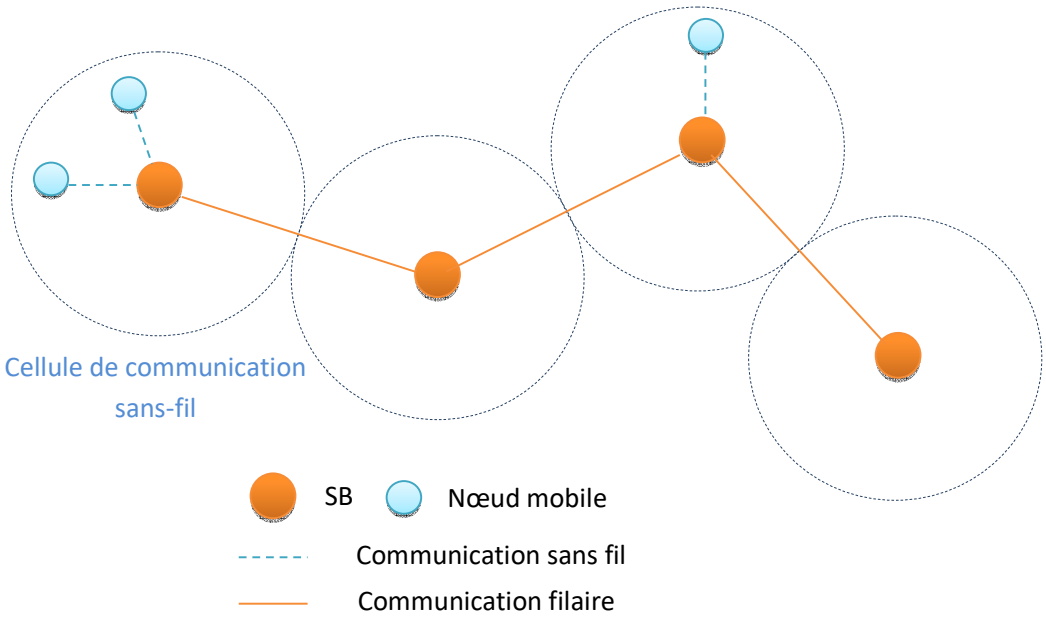

Figure2.1 Modèle de réseau mobile avec infrastructure.

La deuxième classe concerne les réseaux où il y a que des unités mobiles qui se connectent entre elles d'une façon directe, via des communications sans fil, sans existence d'une unité fixe. Cependant, l'absence d'une infrastructure exige aux unités mobiles d'agir comme des routeurs lors du processus de recherche et de maintenance d'une route entre différents nœuds. Le bon exemple correspondant à cette classe est le réseau mobile ad-hoc appelé MANet (Mobile Ad-hoc Network).

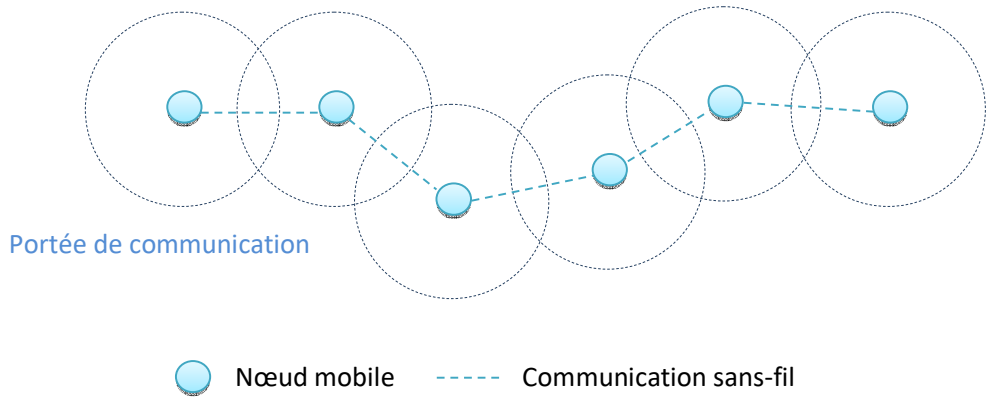

Figure2.2 Modèle de réseau mobile sans infrastructure

3. Les Réseaux ad-hoc mobiles (MANET)

Les MANETs sont des réseaux sans-fil qui sont capables de s'organiser, de se configurer et de fonctionner de manière autonome, sans aucune infrastructure fixe préexistante ou une administration centralisée. Les nœuds dans MANet interagissent et coopèrent pour échanger des services. Ils peuvent s'interconnecter directement avec les nœuds qui se trouvent dans leur portée radio, sinon ils vont passer par des autres nœuds qui se comportent aussi comme des routeurs, pour construire et maintenir une route de recherche jusqu'au nœud destinataire. L'intérêt pour MANET a grandi avec l'augmentation de la popularité des dispositifs sans-fil avec des fonctionnalités très riches. Ces réseaux peuvent être construits d'une façon temporaire pour réaliser un certain objectif ou pour répondre à un certain service, dans des environnements où l'infrastructure n'est pas disponible ou elle n'est digne de confiance, comme des zones militaires, des campus universitaires, des concerts de musique ou d'autres lieux de rencontre.

3.1. Les protocoles de routage dans MANet

Les protocoles de routage dans MANet peuvent être classifiés en fonction de la stratégie de routage et la structure du réseau. Selon la stratégie de routage, les protocoles peuvent être catégorisés comme proactifs ou réactifs. Selon la structure réseau, ils peuvent être catégorisés comme routage plat (*flat routing*), comme routage hiérarchique (*hierarchical routing*) ou comme routage de position géographique assistée (*geographic position assisted routing*). Les deux catégories de réactifs et proactifs relèvent dans le cadre du routage plat.

3.1.1. Protocoles de routage proactifs

Ces protocoles sont appelés proactifs comme ils maintiennent les informations de routage même avant qu'il soit nécessaire. Les informations de routage sont généralement entretenues dans les tables de routage et celles-ci sont mises à jour d'une façon périodique ou avec le changement de la topologie réseau. Elles doivent être diffusées par les différents nœuds du réseau. Autrement dit, les protocoles proactifs essaient de maintenir les meilleurs chemins existants pour toute destination, qui peuvent présenter l'ensemble de tous les nœuds de réseau, au niveau de chaque nœud, et ces chemins sont sauvegardés même si ils ne sont pas utilisés. Les protocoles proactifs ne sont pas convenables pour les réseaux à grande échelle, comme ils nécessitent de maintenir les entrées de tables de routage pour chaque nœud et pour chaque période. Cela cause plus de trafic réseau (***Overhead***) pour les tables de routage et donc plus de bande passante consommée. Par ailleurs, cette technique accélère le processus de transmission de paquets en utilisant immédiatement des routes déjà connues.

Ils existent plusieurs protocoles de routage proactifs pour MANet, comme par exemple : DSDV [Per94], WRP [Mur96], OLSR [Jac01], CGSR [Chi97], HSR [Pat09], MMRP, etc… Nous introduisons le protocole OLSR dans la partie ci-dessous.

- ***Optimized Link State Routing (OLSR)***

Le protocole OLSR [Jac01] est un protocole de routage IP à état de lien optimisé pour être adapté aux réseaux MANet. C'est un protocole proactif, qui utilise des messages de Hello et de contrôle de topologie (Topology Control : TC) pour découvrir et diffuser les informations sur l'état de lien, partout dans le réseau. Chaque nœud utilise cette information de topologie

pour trouver le saut suivant de la destination pour tous les nœuds du réseau, en utilisant les plus courts chemins de transmission.

Dans OLSR, certains nœuds sont sélectionnés comme des MPR (relais multipoint, en anglais : Multi point relays), qui sont responsables de la transmission des messages durant la recherche et la diffusion des informations sur l'état de lien. Chaque nœud sélectionne un sous ensemble de ses voisins à 1-saut comme MPR, d'une façon où ces MPR couvriront tous les voisins à deux-sauts (Figure2.4). Par conséquent, les messages sont distribués seulement à travers ces MPR. Cette technique utilisée dans OLSR réduit le trafic réseau, et minimise le nombre de transmissions redondantes dans le réseau.

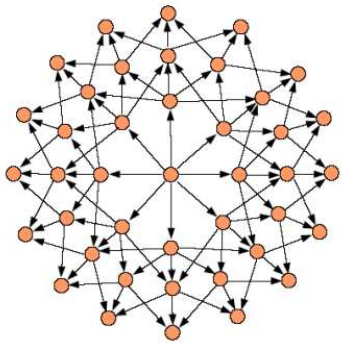

Figure2.3 Diffusion par inondation

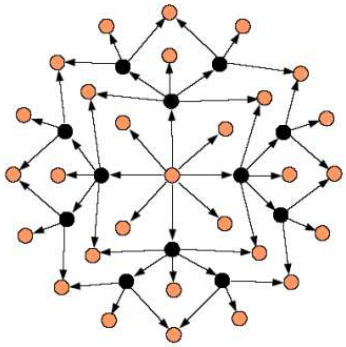

Figure2.4 Diffusion par MPR

Les nœuds maintiennent les informations de voisinage et de MPR, par échange des messages de HELLO avec leurs voisins (envoi-réception). OLSR n'épuise pas le réseau MANET par un mécanisme d'inondation, à travers tous les nœuds du réseau et pour chaque petite période lors de la mise à jour des informations de routage. Il distribue ces informations à travers les MPR assez souvent afin d'assurer qu'elles ne restent pas désynchronisées pendant une longue période de temps.

3.1.2. Protocoles de routage réactifs

Ces protocoles sont appelés réactifs (dits aussi : à la demande) comme ils ne maintiennent pas les informations de routage ou les activités de routage chez les nœuds de réseaux quand il n'y a pas une communication à effectuer. Si un nœud veut transmettre un paquet à un autre nœud, le protocole réactif commence à chercher le chemin et il établit la connexion pour l'envoi et la

réception du paquet. La découverte du chemin se présente généralement par une inondation des requêtes de recherche de route à travers le réseau, pour obtenir une information spécifiée, inconnue au préalable. Cependant, le trafic de maintenance des tables de routage généré par les protocoles réactifs est minimisé par rapport à celui des protocoles proactifs, mais ils causent une augmentation dans la latence de découverte de route ainsi que la grande utilisation d'inondation peut conduire à une obstruction du réseau. Ils existent plusieurs protocoles de routage réactifs pour MANet, comme par exemple : AODV [Per99], DSR [Dav96] et DYMO. Nous allons introduire le protocole AODV dans la partie ci-dessous.

- ***Ad-hoc On Demand Distance Vector (AODV)***

AODV [Per99] offre un routage unicast et multicast. C'est un protocole de routage réactif qui établit une route à une destination seulement à la demande. Il se base sur l'algorithme classique de vecteur de distance, en évitant le problème majeur qui est le comptage à l'infini (*counting to infinity*) par l'utilisation de nombres de séquences dans la mise à jour des routes.

Dans OADV, lorsqu'une demande de connexion s'établit, le nœud demandeur diffuse une requête de recherche de route, les autres nœuds font suivre cette requête en sauvegardant le nœud source pour créer une chaine temporaire des nœuds formant la route de retour (Figure2.5). Quand un nœud reçoit cette requête et possède déjà une route pour cette destination. Il renvoie une requête au nœud demandeur à travers la route de retour. Ensuite, ce dernier utilise la route qui a le moins de nombre de sauts (Figure2.6). Les entrées des tables de routage non utilisées seront recyclées après un certain temps. Dans le cas d'un lien échoué, une erreur de routage est retransmise au nœud source, et le processus de connexion se répète.

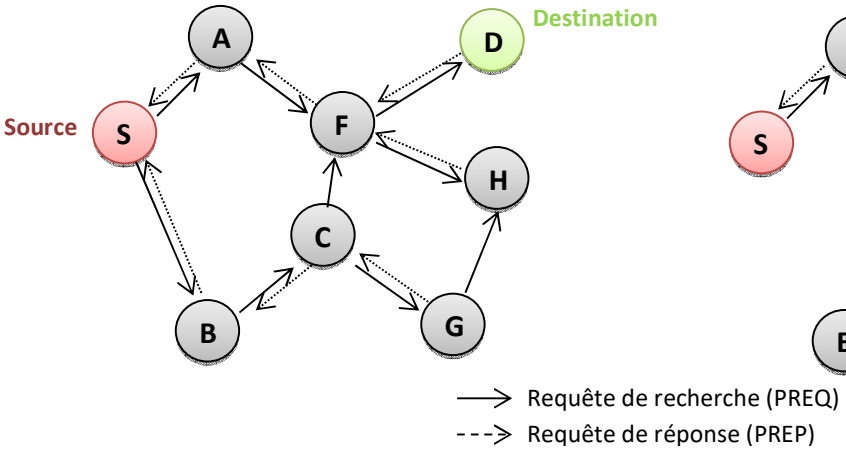

Figure2.5 Propagation de la requête de recherche d'une route

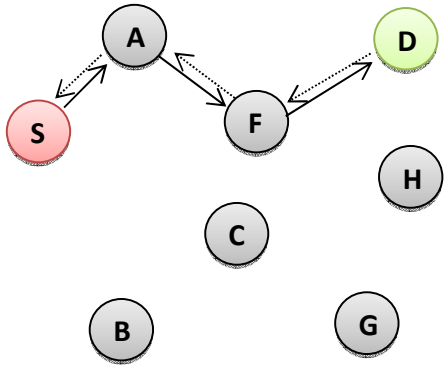

Figure2.6 Détermination de route de la source à la destination

Une grande partie de la complexité de ce protocole consiste à minimiser le nombre de messages pour conserver la capacité du réseau. Par exemple, chaque requête de route possède un nombre de séquences. Les nœuds utilisent ce nombre pour éviter l'utilisation des routes déjà disparues. Un autre concept consiste à assigner un temps de vie (*time to live : TTL*) aux requêtes, qui sert à limiter le nombre de retransmissions de cette requête.

3.2. La contrainte de l'énergie dans les réseaux MANET

La majorité ou la totalité des nœuds de MANet reposent sur des batteries ou des moyens épuisables pour leur énergie, ce qui fait de la conservation de l'énergie, un critère de performance très important dans ces réseaux. Il est nécessaire de noter que l'énergie consommée durant l'envoi de paquet est la source la plus signifiante de la consommation de l'énergie par rapport à tous les modes existants (envoi, réception, inactif et en veille) [Sin12]. Ceci est suivi par la consommation de l'énergie durant le mode de réception. Bien que, pendant le mode inactif, le nœud ne manipule aucune opération de communication, il a été constaté que l'interface sans-fil consomme une quantité considérable d'énergie, proche de celle consommée dans le mode de réception [Pla06]. L'énergie du mode en veille, est une perte d'énergie qui doit être éliminée ou réduite par les schèmes d'efficacité énergétique. A travers des études de mesures de consommation de l'énergie, les expériences ont été menées pour déterminer la consommation d'énergie instantanée pour chaque mode.

3.3. Les techniques de conservation de l'énergie dans les réseaux mobiles sans-fil

3.3.1. Conservation de l'énergie au niveau routage

La conservation de l'énergie au niveau de la couche routage, consiste à choisir la bonne route qui consomme le moins d'énergie depuis la source jusqu'au destinataire. Dans ce cas, le choix du prochain saut se base sur la métrique de l'énergie, qui repose sur la distance du chemin à parcourir, le temps de transmission ou le niveau d'énergie du prochain nœud récepteur. Ils existent plusieurs approches dans la littérature, qui s'intéressent par améliorer ou proposer des protocoles de routage ad-hoc à base de la métrique de l'énergie. Certains travaux [Toh01] ont proposé de transmettre les paquets à travers les nœuds ayant un niveau élevé d'énergie et éviter ceux avec du niveau faible, dans le but d'utiliser équitablement le niveau de puissance entre les nœuds, d'augmenter la durée de vie du réseau et d'éviter les ruptures de routes. D'autres [Ama11][Ama10], ont basé leurs études sur la réduction du nombre de messages de maintenance envoyés, comme les messages de TC, où l'échange d'un grand nombre de TC provoque une consommation d'énergie très considérable.

Parmi les métriques de conservation d'énergie les plus connues, nous mentionnons celle basée sur le plus court chemin, qui consiste à choisir la route avec le moindre nombre de sauts (le nombre de nœuds à traverser depuis la source jusqu'au nœud destinataire). Dans le travail de [Ban02], les auteurs ont proposés de prendre en considération le taux d'erreur du lien radio, afin de minimiser le nombre de transmissions perdues ou ce que l'on appelle les requêtes perdues durant le routage, ce qui par conséquent minimise l'énergie perdue, consommée durant les transmissions et les retransmissions de ces requêtes. Les auteurs de [She02] ont intégré le facteur de puissance de transmission (portée de transmission) afin de calculer l'efficacité énergétique de la route correspondante. Ces auteurs ont montré que la conservation d'énergie dépend de deux facteurs essentiels : la longueur du chemin entre la source et la destination et la puissance de transmission, plus la longueur augmente plus l'énergie consommée augmente. Dans [Bem12], on trouve un protocole de routage hiérarchique à base de clusters, adapté pour les réseaux maillés. Dans le cas d'une variation fréquente de la topologie dynamique dans les réseaux mobiles sans-fil, les processus de construction et de maintenance génèrent un nombre excessif de messages, ce qui est réduit dans la solution de [Bem12] où les cluster-heads sont généralement des nœuds

fixes. Des travaux sur l'efficacité énergétique sont proposés dernièrement [Cap12][Amo12] à base des techniques d'optimisation théoriques, ex. problème d'optimisation formulé en programmation linéaire.

3.3.2. Conservation de l'énergie au niveau MAC

La plupart des approches de conservation d'énergie au niveau de la couche MAC, reposent sur l'idée d'endormissement du nœud pendant les durées du temps où le nœud est en mode inactif. Comme les études ont montré que l'énergie consommée par le nœud quand il est inactif, est égal approximativement à celle consommée durant le mode de réception [Pla06] [XuH01]. C'est la raison pour laquelle, le protocole TDMA est considéré comme le plus économique en énergie par rapport aux autres protocoles de gestion d'accès au canal radio. Il utilise le mode endormissement pour les nœuds qui sont en mode inactif. Cette solution est largement abordée dans la littérature. Elle a été appliquée sur les réseau ad-hoc [Bel07][Cha05], sur les réseaux de capteurs [Van03] et récemment sur les réseaux cellulaires [Bou12][Khi12].

3.3.3. Conservation de l'énergie au niveau physique

Le paramètre qui consomme le plus d'énergie au niveau de la couche physique est la puissance de transmission, qui est aussi un point considérable au niveau de la couche routage, comme il a une influence sur la variation du nombre de sauts entre la source et la destination. Ainsi, au niveau de la couche MAC, comme il affecte la quantité d'interférences [Gup00] qui est un facteur essentiel dans l'ordonnancement des liens radio. Le contrôle de puissance est un mécanisme qui contrôle la puissance de transmission et donc les interférences et les collisions. Par conséquent il peut assurer un niveau acceptable de conservation d'énergie [Bur04]. Ce mécanisme a été utilisé dans certains travaux, pour optimiser l'énergie. La plupart de ces travaux fournissent une diversité de niveau de puissance de transmission pour chaque nœud. Ce dernier, lorsqu'il veut transmettre des paquets, il choisit le niveau qui consomme le moins d'énergie. Cependant, il est nécessaire d'adapter le schéma de codage (MCS) pour s'adapter à la variation de la qualité du canal radio, et d'assurer la coordination entre les nœuds voisins, afin d'assurer des connexions fiables et éviter les retransmissions des requêtes perdues qui consomment plus d'énergie. Ce dernier point, est abordé dans le travail de [Lop11], appliqué sur les réseaux

cellulaires, et leurs résultats montrent que l'allocation du taux de transmission faible aux nœuds, présente un moindre débit et énergie par rapport à l'allocation du taux de transmission élevée.

4. Le déploiement de DHT sur le mobile

D'un côté, nous avons des systèmes à base de DHT, destinés dans un premier lieu aux réseaux fixes et filaires, et d'un autre côté, on a l'évolution énorme des technologies mobiles et sans-fils avec l'arrivée du WiFi qui permet de construire un réseau informatique pour connecter des dispositifs fixes ou mobiles (PDA, tablette, smartphone,…) avec des ondes radio, pour permettre la transmission de données. De plus, il y a le progrès non limité des dispositifs mobiles en terme de mémoires, de capacité des processeurs, de la puissance de calcul et d'autres fonctionnalités qui leurs permettent de stocker des Giga octets de contenus numériques (texte, image, audio, vidéo,…).

Cette évolution conjuguée à la grande utilisation des dispositifs mobiles dans le temps courant, avec l'importance connue par les systèmes de DHT, font du déploiement de DHT sur les réseaux mobiles sans-fils un domaine de recherche très significatif. Par ailleurs, cela présente un grand défi de la nécessité de développer un système qui s'adapte à la mobilité, au taux élevé de perte de paquets, à la fluctuation de bande passante, et à l'énergie limitée des batteries.

Le réseau P2P, quand il est appliqué sur les réseaux mobiles sans-fil, il offre des services multiples tout en se déplaçant avec ou sans infrastructure, tels que le stockage distribué, le partage de fichiers distribués, les communications en temps réel, et la diffusion en directe des chaines de radio/TV locales (LeD14) (Ven10).

Comme les DHTs ont été prévus au premier lieu pour les réseaux fixes et Internet, ils posent des défis pour les réseaux avec des dispositifs mobiles, dans lesquels on doit prendre en considération les ressources limitées (exemple : bande passante et énergie) aussi bien que la topologie de haute dynamicité qui nécessite d'accélérer le processus de recherche.

Ils existent récemment un nombre signifiant de solutions proposées dans ce domaine de déployer les DHT sur réseaux mobiles (ex. les réseaux ad-hoc mobiles, cellulaires ou maillés) dont certains sont mentionnés dans la section suivante avec notre propre classification.

5. Les défis des DHTs fixes et mobiles

Les problèmes qui restent à aborder par les travaux à base de DHT, peuvent être mentionnés dans les points suivants :

a. Le trafic réseau (en anglais *Overhead*): avec toutes les solutions d'amélioration de DHT connues avec le temps, le trafic réseau (ou le nombre de messages circulant dans le réseau) reste un problème important à être aborder. Il peut être généré durant :
   - Le processus de maintenance : le nombre de messages transmis et reçus afin de stabiliser le système (ex. pour mettre à jour les tables de routage, l'emplacement de données ou de nœuds, etc.) est augmenté avec l'augmentation de churn (le taux d'arrivée et de départ des nœuds) ou la dynamicité du système, qui est plus élevée dans les réseaux mobiles.
   - Le processus de recherche : nombre de messages transmis durant le chemin depuis le nœud demandeur jusqu'au nœud cible.

b. La correspondance entre le réseau physique et logique : placer les voisins physiques aussi proches que possible dans l'overlay tout en évitant le trafic élevé, reste une issue importante à développer dans les systèmes à base de DHT.

c. La latence de la recherche : comme le processus principal des protocoles à base de DHT est la recherche, par conséquent l'amélioration de la latence reste toujours une issue signifiante.

d. Le non équilibrage des zones de responsabilités (les données dont le nœud est responsable): les systèmes DHT réalisent un équilibrage de charge en termes de tables de routage, avec les mêmes tailles, mais ils présentent le non équilibrage des zones de responsabilités. Où chaque pair a un nombre très diffèrent de ressources dont il est responsable par rapport à d'autres pairs (ex. dans CHORD, la clé de données est assignée au pair avec un ID égal ou supérieur à la clé).

e. Le taux de succès de la recherche, qui représente le nombre de requêtes de recherche réussite par rapport au nombre total de requêtes envoyées. Cependant, comme l'objectif principal des DHT est la recherche, soit, assurer un taux acceptable de succès et viser à l'améliorer reste un point très important.

f. L'énergie limitée dans les réseaux mobiles : les améliorations de DHT doivent prendre en considération le niveau d'énergie du nœud afin d'éviter l'épuisement rapide du système.

6. Les optimisations existantes de DHT dans les réseaux fixes et mobiles

6.1. Classification [Che16]

Les travaux qui s'intéressent par, appliquer des améliorations sur les DHT existants, visent à atteindre un ou certains des objectifs suivants :

- améliorer le temps de réponse et le taux de succès de la recherche
- réduire le trafic réseau (nombre de messages).

Dans cette partie, nous présentons une classification des approches qui ont été proposées dans la littérature pour renforcer les performances des systèmes DHT (Figure2.7), dans laquelle nous nous appuyons sur la méthode utilisée et par conséquent nous définissons trois catégories : tables de routages étendues, sensibilisation de la localité, et la réplication. Pour chaque catégorie, nous présentons des travaux connexes ainsi qu'une discussion détaillée afin de montrer leurs avantages et leurs inconvénients.

6.1.1. Les catégories

a. Catégorie 1 : « tables de routage étendues »

Les approches de cette catégorie consomment plus d'espace de stockage pour améliorer le temps de recherche, et ce, par rajout de plus d'informations de routage dans les tables, autrement dit, on observe une expansion de la table de routage. Nous divisons cette catégorie en trois sous-catégories :

- Ajout de nouveaux champs dans les TF (Table de Finger).
- Ajout de nouvelles entrées dans les TF.
- Ajout de nouvelles TF.

b. Catégorie 2 : « sensibilisation de la localité »

L'idée des travaux de cette catégorie repose sur la considération de la localité physique du pair, lors de la construction d'overlay. Dans les protocoles les plus populaires (Chord, Pastry,…), un nœud rejoint sa position dans l'overlay (Ring, Tree, …) d'une manière aléatoire sans prendre en considération sa localité physique. Dans ce cas, on va avoir des voisins dans l'overlay qui peuvent être physiquement très loin les uns des autres. Par conséquent, au lieu d'effectuer un processus de recherche avec un voisin proche, on

traverse une longue distance pour atteindre un nœud qui peut quitter le réseau au cours de cette période.

La sensibilisation de la localité permet d'avoir des nœuds voisins sur l'overlay qui sont aussi proches sur l'underlay, ce qui offre une recherche plus rapide et efficace.

c. Categorie 3 : « Réplication »

Le principe est de créer des copies de données, nommées répliques (en anglais : replicas), et les distribuer sur les *r* nœuds appelés nœuds de réplique (en anglais : replica nodes), où *r* est désigné comme le degré de réplication et défini selon la méthode utilisée et les objectifs visés du travail correspondant, tels que: l'augmentation de la disponibilité des ressources, la réduction de la latence ou la réalisation d'un équilibrage de charge.

En se basant sur la méthode de réplication utilisée, nous divisons cette catégorie en trois sous-catégories :

- Réplication de liste des voisins.
- Réplication de groupes associés.
- Réplication de multi fonctions de hachage.

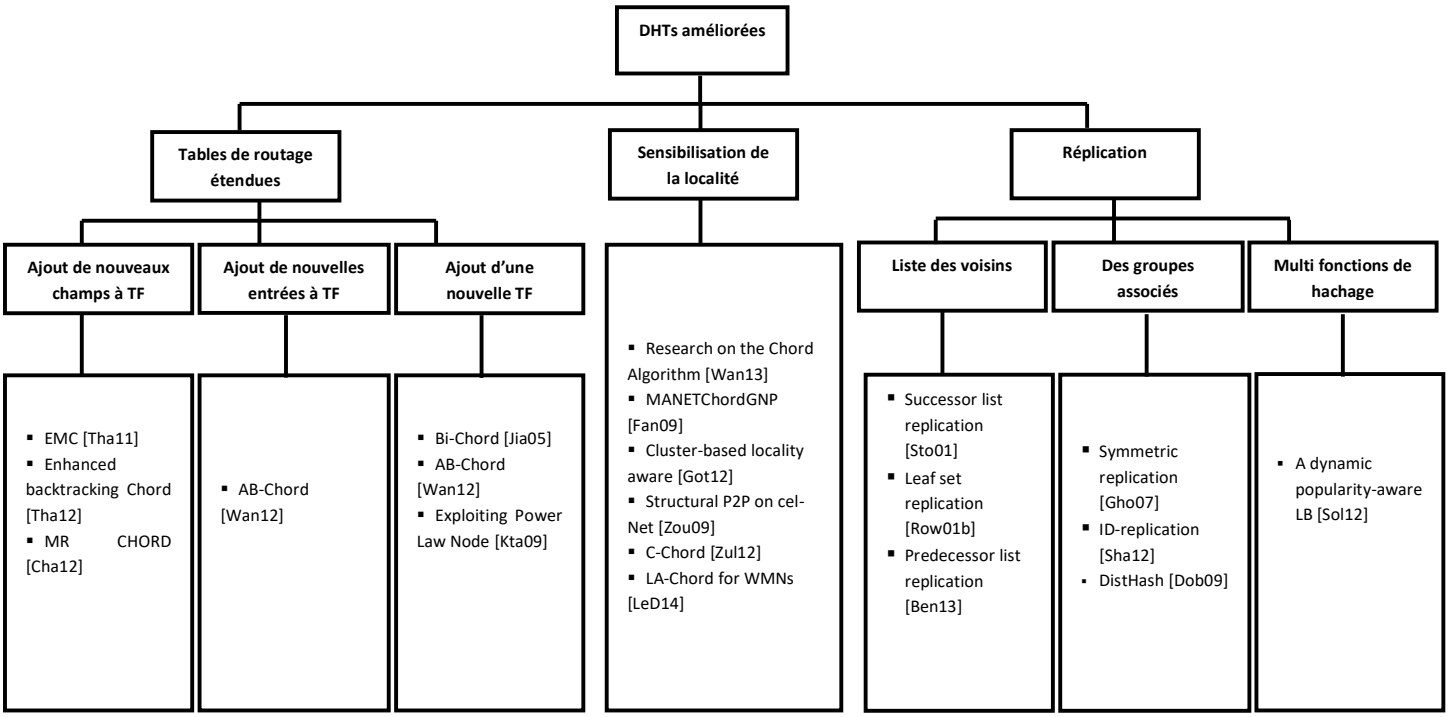

Figure2.7 Classification des solutions proposées pour les systèmes à base de DHT. [Che16]

6.2. Travaux d'optimisation selon les catégories

Cette partie est consacrée à la présentation des solutions d'optimisation des DHT les plus connues dans la littérature.

6.2.1. Tables de routage étendues

a. Bi-Chord [Jia05]

Dans Chord de base, la recherche est effectuée uniquement dans le sens horaire de l'anneau (clockwise direction). Dans [Jia05] les auteurs proposent l'idée de chercher également dans le sens inverse (counter clockwise direction) par une structure de routage bidirectionnelle. Pour celà, ils créent une deuxième table de finger appelée reverse_FT (TF inverse), qui a la même structure que la table de finger originale, dont le champ de « valeur » sera remplit selon la formule suivante :

$(j + 2^m - 2^{k-1}) \bmod 2^m$ tel que : $1 <= k <= m$

Autrement dit, ils sélectionnent les successeurs les plus proches logiquement dans le sens inverse de celui de Chord de base.

b. AB-Chord [Wan12]

Dans ce travail, il y a une sorte de *tradeoff* entre la consommation de l'espace de stockage et la réduction de la latence de recherche. Pour cela, ils utilisent deux tables de finger : la table originale de Chord et la table inverse de Bi-Chord [Jia05] afin de chercher dans les deux sens de l'anneau. En outre, ils ajoutent des entrées supplémentaires pour chacune de ces tables, définies par les formules suivantes :

Pour le sens horaire de l'anneau :

$$n.finger_{[C]}[i].start = \begin{cases} (n + 2^0) mod 2^m & i = 1 \\ (n + 2^{i/2}) mod 2^m & i \text{ is even} \\ (n + 3 * 2^{(i-3)/2}) mod 2^m, & i \text{ is odd} \end{cases}$$

Pour le sens inverse :

$$n.finger_{[CC]}[i].start = \begin{cases} (n+2^m-2^0) mod 2^m & i=1 \\ (n+2^m-2^{i/2}) mod 2^m, i \text{ is even} \\ (n+2^m-3*2^{(i-3)/2}) mod 2^m, i \text{ is odd} \end{cases}$$

Tel que: n.finger[i].start est la valeur correspondante à la i^{eme} entrée de la table de finger.

Le nombre d'entrées dans les deux tables présente un inconvénient, qui est l'augmentation du trafic réseau lors de la mise à jour et la maintenance de ces tables de finger. Comme solution à ce problème, les auteurs proposent d'éliminer les nœuds redondants dans les tables pour ne garder qu'un seul nœud avec la rangée complète des clés dont il est responsable.

Durant le processus de recherche, le nœud demandeur commence par chercher dans sa table de finger originale le nœud avec ID supérieur ou égal à la clé cherchée. Si cet ID est trouvé, le processus de recherche est terminé. Sinon, la recherche va sur la table de finger inverse.

c. Exploiting Power-Law Node [Kta09]

Les auteurs de [Kta09] proposent un modèle de routage bidirectionnel, qui utilise les informations de voisinage de chaque nœud, appelé Power-Chord (P-Chord). Ils ajoutent une « table de sélecteur de finger » (FST) plus de la table de finger originale. La FST de nœud « A » possède comme entrées, les sélecteurs de finger (FS) qui sont les nœuds ayant A comme nœud de finger, et pour chaque FS ils mentionnent son nœud prédécesseur (pred). De là, FST possède deux champs : SF et son prédécesseur.

Le nœud demandeur commence par examiner son FST, il compare la clé cherchée avec l'ID de chaque sélecteur de finger (SF) et son prédécesseur correspondant. Pour chaque entrée de FST, si la valeur de la clé se trouve entre FS et son prédécesseur, la requête de recherche sera directement transmise à ce SF, comme possesseur de la clé cherchée. Par conséquent, le processus de recherche est terminé avec succès. Autrement, le nœud transfert la requête au SF ou au prédécesseur le plus proche.

d. EMC [Tha11]

Au lieu d'avoir uniquement le finger et son successeur comme dans basic Chord, les auteurs de ce travail ajoutent deux champs supplémentaires qui sont la valeur de stabilité du nœud et la

latence du chemin défini par le message Ping. Ce travail défini pour chaque destination dans la table de finger, les trois successeurs avec les chemins les plus courts, triés dans l'ordre croissant de leurs latences. Le nœud demandeur envoie une requête au successeur avec le plus court chemin.

e. Enhanced backtracking CHORD [Tha12]

Ce travail utilise la même structure de TF que celle d'EMC [Tha11], avec deux champs supplémentaires. Lors de la recherche d'une clé, le concept de temps mort (timeout) est utilisé. Le nœud demandeur va envoyer une requête avec un temps-mort au premier successeur avec le meilleur chemin. Si aucun paquet de réponse n'est reçu durant ce temps-mort, la requête sera envoyée au successeur avec le deuxième meilleur chemin, et ainsi de suite jusqu'à arriver au nœud destinataire.

f. MR CHORD [Cha12]

L'idée présentée ici est d'avoir une table de finger fraiche et mise à jour. Pour cela, trois champs sont ajoutés à la TF:

- Succ: il est incrémenté par un, si la requête de recherche envoyée au finger correspondant est une réussite.
- Fail : il est incrémenté par un, si la requête envoyée au finger correspondant a échouée et si ce nœud existe encore, car si il est parti (déconnecté), il sera éliminé de la table et remplacé par son nœud prédécesseur.
- Weaknode : ce champ possède une valeur booléenne, il est « faux » si (F[i].succ – F[i].Fail < 2) et il est « vrai » si (F[i].Fail – F[i].succ >= 2), dont F[i] est le finger de la $i^{ème}$ entrée. Dans le cas de « faux », le finger doit exécuter une vérification dans sa table de finger pour voir si l'un de ces finger n'existe plus (déconnecté).

6.2.1.1.Discussions

L'ajout des informations locales sur les ressources, permet de trouver plus rapidement le responsable de la ressource, que ce soit localement ou dans un nombre limité de sauts. Les résultats de simulation des travaux de cette catégorie montrent l'impact que cette idée apporte : comme la réduction du nombre de sauts logiques nécessaires pour atteindre le nœud destinataire,

autrement dit, réduire la longueur du chemin de recherche. Cette solution optimise ainsi le temps de latence et le taux de succès de la recherche. Par ailleurs, comme désavantage, cette idée augmente la taille de la table de routage et les résultats de simulation de ces travaux présentent l'augmentation du trafic réseau en fonction du nombre de nœuds et le taux de churn. Cette augmentation provient des mises à jour régulières de chaque entrée des tables de routage pour assurer la disponibilité des nœuds responsables.

6.2.2. Sensibilisation de la localité

a. Research on the Chord Algorithm [Wan13]

Le modèle présente une architecture de Chord à deux niveaux, un anneau primaire pour les super pairs et un anneau secondaire pour les pairs ordinaires considérés selon la location physique. La répartition des nœuds en sous-anneaux est fait selon la méthode de division de temps, où certains nœuds sont choisis aléatoirement pour être des nœuds repères, et chaque autre nœud ordinaire envoie des messages à ces nœuds repères et calcule le temps de transmission de chaque message, selon lequel il sélectionne le cluster correspondant. De la même manière, les nœuds les proches physiquement seront placés dans le même cluster.

Durant le processus de recherche, le nœud demandeur commence par chercher dans le cluster auquel il appartient, s'il ne trouve pas la ressource cherchée. Il transmet la requête au super pair de son cluster pour chercher dans l'anneau primaire.

b. *MANETChordGNP* [Fan09]

MANETChordGNP est basé sur la considération de la localité physique durant la construction de l'overlay de CHORD à base de MANET et le protocole de routage AODV.

Les auteurs de cette étude utilisent le système de positionnement GNP (Global Network Positionning) pour déterminer l'ID, en fonction des coordonnées géométriques de chaque nœud. GNP sélectionne certains nœuds pour jouer le rôle des Landmarks (qui connaissent déjà leurs coordonnés), comme points de repères pour d'autres nœuds qui veulent s'intégrer dans l'espace géométrique, et puis les pairs mesurent leur latences sur un ensemble fixe des nœuds Landmark par une fonction de distance afin de calculer leurs coordonnées. En fonction de cette dernière, chaque nœud aura un ID temporaire unique, qui définit sa position logique dans l'anneau overlay. Si le Landmark se déplace ou quitte le réseau, l'un des nœuds les plus proches le remplace. Par

ailleurs, les mécanismes de recherche et d'indexation de ressource sont les mêmes que dans Chord de base.

c. *Cluster-based locality aware* [Got12]

L'approche de ce travail combine l'Overlay de DHT avec l'Underlay du réseau MANET, en utilisant OLSR comme protocole de routage. Ils utilisent un système de positionnement (ex. GPS) pour connaître la localisation physique de chaque noeud. Selon laquelle, l'ID est défini d'une manière précise qui consiste à diviser la zone d'opération en clusters et assigner à chaque cluster un ID unique utilisé comme un préfixe dans le processus de determination d'ID. De même, ID est la concaténation de cluster-ID (préfixe) et un nombre aléatoire défini par le nœud (suffixe). Par exemple, un noued avec un nombre aléatoire X appartient au cluster avec clusterID= 001, et donc l'ID du nœud dans le système sera 001X.

Quand le nœud mobile quitte le cluster, un nouveau ID est exigé avec le préfixe du nouveau cluster.

En outre, le mécanisme d'indexation des ressources dans ce travail, est le même que dans Pastry de base. Pour le mécanisme de recherche, les auteurs utilisent une table de routage principale pour les clusters et une table de leaf-set pour les nœuds ordinaires d'un cluster. Ainsi, ils commencent par chercher le bon cluster puis le nœud cible.

d. *Structural P2P model on mobile cellular networks* [Zou09]

Ce travail intègre l'overlay Chord dans les réseaux cellulaires pour construire un système appelé C-Chord à deux-niveaux, le Master-Chord qui comprend les stations de base et le Slave-Chord des nœuds cellulaires.

La sensibilisation de la localité est appliquée ici, en plaçant les nœuds cellulaires de la même zone physique (i.e. de la même SB) dans le même Slave-Chord. Par conséquent les nœuds dans l'overlay seront placés en fonction de leur localisation physique en utilisant la méthode de landmarks-clustering où les stations de base sont les landmarks.

e. *C-Chord* [Zul12]

L'approche de ce travail s'appelle C-Chord. Elle combine Chord avec les réseaux cellulaires, en utilisant non seulement les pairs cellulaires comme utilisateurs mais ainsi les pairs fixes (pairs d'internet), qui forment l'anneu principal (m-Chord) et les pairs cellulaires forment l'anneau auxiliaire (a-Chord). Cette architecture permet aux pairs cellulaires de choisir entre réccupérer les données à partir des pairs d'internet à un taux rapide ou à partir des pairs cellualires du même a-Chord pour éviter les frais d'internet.

La sensibilisation de la localité consiste ici à placer les pairs cellulaires de la même station de base dans le même a-Chord mais la SB elle même ne participe pas au système, seulement la necessité de son nombre d'identification (cell-id). D'une façon plus détaillée, le hachage de cet cell-id donne une clé spéciale (special-key) à placer dans le système selon le chord de base, dans le bon pair fixe (avec ID égal ou supérieur à la clé). Cette méthode ne contient aucun super pair mais elle définit pour chaque pair cellulaire, une liste appelée IPgw qui contient les IP des pairs fixes, qui sont les successeurs et les predecesseurs de la special-key correspondante. Puis, on utilise cette liste comme une passerelle vers m-Chord, pour communiquer avec les pairs fixes et récupérer les données à partir d'internet. Ainsi, chaque pair cellulaire contient une liste IPkey qui contient les IP des pairs cellulaires du même a-Chord.

6.2.2.1.Discussions [Che15]

Les méthodes de cette catégorie montrent les avantages apportés par la sensibilisation de la localité, qui sont l'accélération de la recherche et la réduction du nombre de sauts physiques traversés par les requêtes, c'est le résultat dû au fait d'avoir à la fois des voisins physiques comme des voisins logiques dans l'overlay P2P. Par ailleurs, les techniques utilisées pour déterminer la localisation physique peuvent présenter des inconvénients, tels qu'une augmentation de la consommation de l'énergie causée par les systèmes de localisation (ex. GPS et GNP), ou une augmentation de trafic réseau, causée par les envois périodiques des messages aux nœuds voisins, pour déterminer la location.

6.2.3. Réplication

a. Réplication de la liste des voisins

-*Successor list replication* [Sto01]: les auteurs de Chord proposent une méthode de réplication dans les membres de la liste des successeurs. Autrement dit, le nœud *n* responsable de la ressource *k* ou ce qu'on appelle nœud de racine, crée des répliques de *k*, puis il stocke une réplique dans chaque membre de la liste des successeurs de *n*.

-*Leaf-set replication* [Row01b]: dans ce travail, les auteurs de Pastry répliquent les ressources du nœud *n* dans ses nœuds les plus proches numériquement (r/2 prédécesseurs et r/2 successeurs) qui forment sa table de leaf-set.

-Predecessor list replication [Ben13]: l'idée présentée ici est que le nœud de racine réplique ses ressources dans ses (m-1) prédécesseurs immédiats dans l'anneau de Chord, afin d'augmenter la possibilité d'arriver à l'un de ces nœuds prédécesseurs avant d'arriver au nœud de racine dans le processus de recherche et donc réduire la longueur du chemin de recherche.

Concernant le processus de maintenance de cette sous-catégorie, quand un nœud joint ou quitte le système, les listes de voisinage changent, ce qui implique que les répliques stockées dans les nœuds de ces listes, changent également. Dans le cas d'un processus de jonction, le nouveau nœud contacte l'un des nœuds existants pour avoir des informations sur son emplacement dans le système. Ses voisins l'informent sur les ressources et les répliques dont il est responsable. Puis, ces voisins mettent à jour leurs répliques en fonction des nouvelles listes de voisinage. Dans le cas d'un processus de départ, l'un des voisins sera le responsable de ses ressources et les autres voisins vont mettre à jour leurs listes et leurs répliques.

Discussions

L'avantage connu de la réplication à base de la liste de voisinage est l'augmentation de la robustesse face à la défaillance des nœuds, où dans les réplications de liste des successeurs et de leaf-set, le nœud défaillant est remplacé par l'un de ses voisins, par conséquent la requête de recherche qui était destinée à ce nœud défaillant sera résolue par son voisin remplaçant.

La méthode de réplication de liste de successeurs ne réduit pas la longueur du chemin de recherche, comme la requête de recherche arrive toujours au nœud racine avant d'arriver à ses successeurs, ce qui provoque aussi un goulot d'étranglement chez ce nœud racine. En outre, dans la méthode de Leaf-set, le nœud de destination n'est pas toujours le même, mais il est choisi aléatoirement parmi les membres de la liste de leaf-set. La méthode de liste de prédécesseurs présente de meilleurs résultats concernant ces deux points (latence de recherche et sécurité du système) puisque il y a une possibilité que la requête arrive à l'un des prédécesseurs responsables et donc de réduire la longueur du chemin de recherche. Cette méthode évite également, le problème de goulot d'étranglement puisque le nœud de destination n'est pas toujours le nœud racine mais, il peut être l'un des nœuds de répliques.

Une caractéristique commune des travaux de cette sous-catégorie, est que le nœud demandeur (en dehors de la liste de voisinage) ne possède aucune information sur les nœuds de répliques.

Un inconvénient majeur de la réplication de liste de voisinage est le nombre élevé de messages, exigés lors de maintenance (mise à jour) des répliques dans le cas de churn, où la liste de voisinage correspondante sera changée à chaque fois. Avec un degré de réplication *r*, le nombre de nœuds qui seront impliqués dans le processus de mise à jour est $r \times 2$ nœuds dans le cas d'un nœud joignant, et un nombre de $(r \times 2) - 1$ nœuds dans le cas d'un nœud quittant.

b. Réplication des groupes associés

- *Symmetric Replication* [Gho07]: dans ce travail, chaque identifiant *p* est associé à *r* autres identifiants dans le système, dans lesquels *p* réplique ses ressources, comme ces *r* nœuds répliquent leurs ressources dans *p*. Autrement dit, l'espace d'identification est divisé en *r* classes équivalentes d'identifiants et les nœuds de la même classe stockent les même ressources. Un nœud qui vient de joindre le réseau contacte l'un des nœuds associés pour avoir des informations sur les ressources qui vont être de sa responsabilité.

Cette méthode peut être appliquée sur tout système de DHT, et la jonction ou le départ d'un nœud implique *r* nœuds dans le processus de maintenance qui sont les nœuds de la même classe. En outre, une requête de recherche de la clé *k* peut être transmise à n'importe quel nœud associé à *k*.

- *ID-Replication* [Sha12]:l'idée ici est d'utiliser des groupes à la place des nœuds et assigner chaque rangée de ressources à un groupe avec un identifiant spécifique. Les nœuds de chaque groupe répliquent les ressources dont ce groupe est responsable. Comme ce travail est appliqué sur Chord, la clé *k* est assignée au group avec ID égal ou supérieur à *k*, et puis les nœuds de ce groupe stockent la clé *k*. Les nœuds d'un seul groupe utilisent le *gossiping* pour synchroniser la vue du groupe, et les groupes utilisent un algorithme de stabilisation périodique entre eux, pour rafraîchir les tables de routage.

Pour réaliser un équilibrage de charge de ces groupes, les auteurs utilisent un processus de fusionnement pour les groupes sous-chargés et un processus de répartition pour les groupes surchargés.

- *DistHash* [Dob09]: ce travail utilise une structure hiérarchique des super pairs (appelés RAgents), dont chacun est responsable d'un groupe de pairs (Agents) sélectionnés selon les métriques de distances pour avoir un cluster des agents proches physiquement. Les agents contiennent des répliques, et les RAgents en plus de stocker des répliques, contiennent un catalogue de métadonnées qui comprend des informations sur les répliques disponibles dans son cluster. En outre, RAgent contient une liste qui mentionne le nombre de répliques retenues par chaque nœud dans le cluster.

Quand un nœud veut ajouter un nouveau objet au système, il contacte son RAgent qui va trouver l'agent responsable et choisit deux autres agents pour retenir les répliques, Ce choix s'intéresse aux nœuds qui possèdent le plus petit nombre de répliques afin de réaliser un équilibrage de charge. Puis, le RAgent met à jour le catalogue de métadonnées en fonction de ces changements.

La responsabilité prise par le RAgent lors du stockage, recherche et maintenance aide à accélérer ces processus. Comme le nombre de RAgents dans le système est trop faible par rapport au nombre des agents, pour cela la communication entre eux sera plus rapide, mais d'un autre coté elle augmente la charge du travail de RAgent d'une manière très rapide.

Discussions

Le premier objectif des travaux de cette sous-catégorie est de réduire le nombre de messages nécessaires pour la maintenance des répliques par rapport à la sous-catégorie précédente. Où la maintenance des répliques d'un seul groupe exige une communication entre tous les nœuds de ce groupe, par conséquent elle implique *r* nœuds dans le processus de mise à jour dans le cas d'un nœud joignant et implique *(r-1)* nœuds dans le cas d'un nœud quittant. D'un autre côté, il y a des caractéristiques différentes des travaux de cette sous-catégorie, qui sont mentionnés ci-dessous.

L'inconvénient de la réplication symétrique est qu'elle exige lors de l'obtention des données à répliquer et lors de la récupération des clés, une manipulation d'une structure de données complexe pour chaque nœud.

La solution d'Id-Replication permet d'avoir différents degrés de réplication pour chaque zone de clés définies entre rmin et rmax. Ces derniers déterminent le nombre de nœuds dans un groupe pour décider si c'est nécessaire d'appliquer un processus d'équilibrage de charge, contrairement à la réplication symétrique où l'équilibrage de charge entre classes est garanti, puisque le nombre de nœuds dans chaque classe est égale toujours à *r*, en raison de l'utilisation des identifiants numériques.

De plus, contrairement à la réplication symétrique, la méthode d'Id-Replication n'exige pas une manipulation d'une structure de données complexe pour atteindre le nœud cible. Mais comme inconvénient de cette méthode, dans le processus de fusionnement et lors de la recherche d'un groupe avec la plus petite charge parmi tous les pairs dans le système ou certains d'entre eux, pour partager les répliques avec, ça nécessite un contact avec un grand nombre de pairs et donc encore plus de messages.

c. Réplication de multi fonctions de hachage

-A dynamic popularity-aware load balancing [Sol12]:

Ce travail vise en premier lieu à réaliser un équilibrage de charge dans les systèmes P2P structurés en utilisant la méthode de déplacement des nœuds ou une méthode de réplication. En

utilisant un ensemble de répertoires de charges appelés loadDir qui contient des informations sur les charges des nœuds correspondants en plus d'autres informations, et utilise ainsi un ensemble de répertoires de réplication appelées repDir contenant comme entrée, l'item répliqué et sa destination.

Le nouveau nœud joignant, utilise une fonction de hachage appelée FirstHash pour obtenir un ID qui doit être placé dans le système, puis obtenir un identifiant du répertoire appelé IdDir pour savoir à quel loadDir, doit-il transmettre ses informations.

Si un nœud est surchargé à cause de la popularité d'un item A, les auteurs proposent de répliquer cet item A, dans un autre nœud. Pour cela, ils utilisent une deuxième fonction de hachage secHash et puis il cherche le successeur de secHash (A) pour lui rajouter une entrée dans son repDir. Le nœud de réplique sera choisi par loadDir en fonction de la charge minimale.

L'avantage principal de cette solution est la réalisation d'un équilibrage de charge entre les nœuds, tout en prenant en considération la popularité des items. Les items avec une haute popularité vont avoir plus de répliques dans le système. D'un autre côté, le processus de maintenance exige de maintenir non seulement les nœuds de répliques, mais aussi les repDir et les loadDir d'une façon périodique en raison de la dynamicité du système.

Discussions

L'inconvénient de la réplication à base de multi fonctions de hachage est le trafic perdu qui ne retourne aucun résultat. C'est le cas où on cherche une clé *k* avec FirstHash(k) alors que le nœud cible n'est plus disponible et donc la réponse retourne une erreur. Le demandeur va lancer une autre requête avec secondHash(k) et ainsi de suite jusqu'à atteindre la ressource, ce qui provoque la perte de messages.

6.2.3.1.Discussions

Généralement, une requête de recherche peut retourner soit une réponse positive quand elle atteint le nœud cible ou une réponse négative (erreur) quand la destination est introuvable, ce qui est le cas où la ressource est perdue. Par conséquent, le mécanisme de réplication est utilisé pour augmenter la disponibilité de données dans le système, afin d'optimiser le taux de succès de la

recherche et de réaliser la robustesse et la tolérance aux fautes des systèmes P2P dynamiques. De plus, certains travaux comme celui de la liste des prédécesseurs, utilisent le mécanisme de réplication pour réduire la latence de recherche.

Le problème majeur de la réplication est le trafic généré par le processus de maintenance, quand les nœuds mettent à jour leurs répliques, ce qui exige l'échange de messages entre le nœud racine et les nœuds de répliques ou entre les nœuds de répliques eux-mêmes. Le nombre de ces messages diffère d'une méthode à une autre.

Tableau 2.1-Un tableau comparatif des solutions connexes de DHT (Tables de routage étendues) [Che16]

	Caractéristiques / Solutions	Topologie réseau	Protocole d'overlay	Structure hiérarchique	Point unique de défaillance	Sensibilisation de la localité	Tables de routage	Cout de maintenance	Objectif principal	Processus de recherche
Tables de routage étendues	**Bi-Chord [**Jia05]	Fixe	Chord	Un-niveau	Non	Non	-TF Originale -Reverse_FT (table inverse)	Maintenir TF (2*logN) et liste des succ (*s*) par chaque nœuds	Optimiser l'efficacité de la recherche.	Chercher dans les deux TF, l'ID le plus proche à key.
	AB-Chord [Wan12]	Fixe	Chord	Un-niveau	Non	Non	-TF Originale +entrées supplémentaires -Reverse_FT+ entrées supplémentaires	4*logN + s Ils éliminent les entrées redondantes pour réduire ce cout.	Optimiser l'efficacité de la recherche.	Chercher dans la TF originale puis dans la reverse_FT.
	Exploiting Power-Law Node [Kta09]	Fixe	Chord + tout protocole de DHT	Un-niveau	Non	Non	- TF Originale -Table de sélecteur de finger (FST)	logN (maintient la TF par ajouter l'ID de pred aux messages de maintenance.	Optimiser l'efficacité de la recherche.	Vérifier à partir de FST, si key se situe entre SF et son pred, sinon chercher le pred de key.
	EMC [Tha11]	Mobile (MANet)	Chord	Un-niveau	Non	Oui	-TF avec deux nouveaux champs (stabilité et latence)	Maintient le nœud responsable et les trois nœuds correspondants avec meilleurs chemins.	Réduire l'Overhead de la recherche et les défaillances.	Transmet la requête au succ ayant le chemin minimal. Si aucune réponse: la transmet au succ avec deuxième meilleur chemin.
	Enhanced backtracking CHORD [Tha12]	Mobile (MANet)	Chord	Un-niveau	Non	Oui	-TF avec deux nouveaux champs (stabilité et latence)	Maintient le nœud responsable et les trois nœuds correspondants avec meilleurs chemins.	Réduire l'Overhead de la recherche et les défaillances sans accroitre la latence.	Transmet la requête au succ ayant le chemin minimal. Si aucune réponse: la transmet au succ avec deuxième meilleur chemin.
	MR CHORD [Cha12]	Mobile (Type de net n'est pas mentionné)	Chord	Un-niveau	Non	Non	-FT avec trois nouveaux champs (succ, fail et weaknode)	Maintient le nœud responsable dans chaque entrée.	Maintient les informations de routage (TF) fraiches et mises à jour face à la forte dynamicité.	Chercher dans la TF, l'ID le plus proche à key, comme dans Chord de base.

Tableau 2.2 Tableau comparatif des solutions connexes de DHT (Sensibilisation de la localité) [Che16]

	Caractéristiques / Solutions	Topologie réseau	Protocole d'overlay	Structure hiérarchique	Point unique de défaillance	Sensibilisation de la localité	Tables de routage	Cout de maintenance	Objectif principal	Processus de recherche
Sensibilisation de la localité	**Research on the Chord Algorithm** [Wan13]	Fixe	Chord	Deux-niveaux	Super nœud	Oui	-TF originale pour chaque nœud ordinaire et super.	Log N	Optimiser l'efficacité de la recherche.	Chercher dans le sous-anneau correspondant puis dans l'anneau principal.
	MANETChordGNP [Fan09]	Mobile (MANet)	Chord	Un-niveau	Non	Oui	-TF originale.	Log N	Réduire le chemin physique de recherche.	Chercher dans la TF, l'ID le plus proche à key.
	Cluster-based locality aware [Got12]	Mobile (MANet)	Pastry	Deux-niveaux	Non	Oui	-TR principale (clusters) -Table de Leaf-set (tous les pairs dans le cluster)	-Maintenir la TR principale. -Maintenir la table de leafset en incluant tous les nœuds du cluster.	Réduire l'Overhead.	Chercher le bon cluster dans la TR principale, puis chercher le nœud cible dans la table de leaf-set.
	Structural P2P model on mobile cellular networks [Zou09]	Mobile (les réseaux cellulaires)	Chord	Deux-niveaux	Station de base	Oui	-Table de MasterChord (SB) -Table de SlaveChord (pairs-cellulaires)	Log N	Accélérer la recherche et résout le problème d'un seul point de défaillance dans les méthodes de Landmark.	Commencer à chercher le nœud cible dans slaveChord, si ça n'existe pas : chercher dans le mainChord.
	C-Chord [Zul12]	Mobile (les réseaux cellulaires)	Chord	Deux-niveaux	Non	Oui	TF originale + listes d'IPgw et IPkey	Log N + le nombre d'éléments de IPGw et IPkey.	Réduire le trafic de recherche.	Chercher l'ID le plus proche à key, dans la liste de IPgw ou IPkey.
	LA-Chord for WMNs [LeD14]	Mobile (les réseaux maillés)	Chord	Deux-niveaux	Routeur de réseau maillé (MR)	Oui	Nouvelle TF de Chord de $3 \times \log4\ N$ entrées pour chaque MR.	$3 \times \log4\ N$ (pour la maintenance d'une seule TF).	Diminuer le nombre de sauts physiques.	Chercher l'ID le plus proche à key, parmi toutes les entrées de la nouvelle TF.

Tableau 2.3-Un tableau comparatif des solutions connexes de DHT (Réplication) [Che16]

	Caractéristiques / Solutions	Topologie réseau	Protocole d'overlay	Structure hiérarchique	Point unique de défaillance	Sensibilisation de la localité	Tables de routage	Cout de maintenance	Objectif principal	Processus de recherche
Réplication	**Successor list replication** [Sto01]	Fixe	Chord	Un-niveau	Non	Non	-TF originale	LogN + maintenir les répliques (2*r)	Fournir une tolérance aux fautes en face de haut taux de churn.	Chercher dans la TF, l'ID le plus proche à key.
	Leaf set replication [Row01b]	Fixe	Pastry	Un-niveau	Non	Oui	-Table de leaf-set	2^b-1*($\log_{2^b} N$) + maintenir les répliques (2*r).	Fournir une tolérance aux fautes en face de haut taux de churn.	Chercher le nœud qui partage plus de digits avec key.
	Predecessor list replication [Ben13]	Fixe	Chord	Un-niveau	Non	Non	-TF originale	Maintenir TF (logN) et liste des pred (r) + (r*2) dans le cas de churn.	Fournir une haute disponibilité de données pour réduire la latence.	Chercher dans la TF, l'ID le plus proche à key.
	Symmetric replication [Gho07]	Fixe	Tout protocole de DHT	Un-niveau	Non	Oui (optionnel)	Table de hachage locale à deux dimensions: présente les items avec IDs.	Implique r nœuds dans la maintenance de réplication et de TR.	Réduire l'Overhead de maintenance de degré de réplication et accroitre la sécurité.	Envoyer plusieurs requêtes simultanées et choisir la première réponse qui arrive.
	ID-replication [Sha12]	Fixe	Chord	Un-niveau	Non	Non	TF originale de Chord où les entrées sont les groupes.	Log N + impliquer tous les nœuds du groupe pour maintenir la réplication.	Fournir une tolérance aux fautes en face d'un haut taux de churn.	-Chercher dans TR: le groupe le plus proche à key. -la requête peut être routée vers un nœud aléatoire du groupe.
	DistHash [Dob09]	Fixe	Tout protocole de DHT	Deux-niveaux	RAgent	Oui	Catalogue de métadonnées (comme une table de hachage) pour chaque RAgent.	Impliquer tous les agents de chaque cluster pour maintenir le catalogue de métadonnées et la liste des répliques.	Réaliser l'équilibrage de charge des grands ensembles de répliques.	Le demandeur contacte un RAgent qui va chercher dans son catalogue de métadonnées, une key et l'un de ses agents responsables.
	A dynamic popularity-aware load balancing [Sol12]	Fixe	Chord	Deux-niveaux	LoadDir + repDir	Oui	TF originale de Chord.	Maintenir: TF, nœuds de répliques, loadDir et RepDir.	Réaliser l'équilibrage de charge en fonction de la popularité des items.	Chercher le succ de firsthash(key) dans sa TF, si ça n'existe pas: chercher le succ de sechash(key)…

6.3. Discussion générale (Tableaux comparatifs) [Che16]

Cette partie présente une discussion sur les travaux déjà mentionnés, afin de montrer ce que chaque catégorie offre comme points forts et points faibles aux systèmes de DHT.

Les Tableaux2.1, 2.2 et 2.3 présentent une étude comparative des solutions DHT en fonction des principaux points suivants : topologie réseau, protocole d'overlay, structure hiérarchique, point unique de défaillance, sensibilisation de la localité, tables de routage, cout de maintenance, l'objectif principal et le processus de recherche.

Des acronymes utilisés dans les tableaux comparatifs :

Acronyme	***Description***
N	La taille réseau
s	La taille de la liste des successeurs
r	Le degré de réplication
k	Clé de la ressource cherchée
Succ	Successeur
Pred	Prédécesseur
Net	Réseau
TF	Table de finger
TR	Table de routage
MR	Routeur maillé (mesh router)

L'objectif commun des trois catégories consiste à :

- optimiser la performance de recherche : la première et la deuxième catégorie visent à réduire le nombre de sauts ainsi que la longueur du chemin de recherche.

La première catégorie concerne les sauts logiques, en augmentant la possibilité de résoudre la requête de recherche localement en donnant plus d'informations sur le système, tandis que la seconde catégorie, concerne les sauts physiques, en plaçant les nœuds les plus proches physiquement comme des voisins sur l'overlay. Bien que l'objectif principal de la troisième

catégorie n'est pas la réduction de nombre de sauts, cela peut être réalisé lorsque la méthode de recherche utilisée permet d'arriver à un nœud de réplique avant le nœud racine.

Réduire le nombre de sauts, que ce soit dans le système logique ou physique est très important pour le processus de recherche, ce qui réduit la longueur du chemin de recherche ainsi que le temps de réponse. Cependant, la méthode utilisée ne devrait pas provoquer un trafic réseau élevé, durant : le processus de maintenance des TF étendues, l'obtention de la location actuelle ou la restauration des répliques, ce qui est un point d'une importance capitale pour les chercheurs et doit être pris sérieusement en considération. Dans la troisième catégorie, les deux dernières sous-catégories présentent des méthodes qui réduisent le coût de maintenance par rapport à la première sous-catégorie.

De plus, les nœuds défaillants entrainent une perte de ressources, ce qui fait de la réplication des copies dans d'autres nœuds : un aspect très important pour optimiser la tolérance aux pannes. La robustesse du système est l'objectif principal de la troisième catégorie. C'est également un point considérable dans la deuxième catégorie : réalisé par l'accélération de processus de mise à jour des tables de routage afin d'éviter des entrées ayant expirées et des requêtes de recherche perdues ou erronées.

7. Conclusion

Nous avons dressé un état de l'art sur les réseaux mobiles sans-fil en général et sur les MANet en particulier, en décrivant les protocoles de routage, les contraintes d'énergie et les techniques de conservation de l'énergie au niveau du routage, de la couche MAC et physique.

Une étude détaillée et critique sur les DHTs fixes et mobiles est élucidée. Dans ce contexte, nous avons introduit puis expliquer le concept de déploiement de DHT sur les réseaux mobiles et les défis correspondants. Les problèmes à aborder sont déterminés par les travaux de DHT fixe et mobile.

Nous avons proposé une classification des optimisations existantes de DHT en fonction de la technique utilisée. Des tableaux comparatifs d'un certain nombre de travaux sont présentés suivis de larges discussions. Cette étude détaillée et critique est appelée à servir comme un guide et comme base pour la conception de l'approche qui est introduite dans le chapitre suivant. Elle

consiste à associer le domaine de P2P à base de DHT au domaine des réseaux mobiles, en prenant en considération les issues mentionnées ci-dessus et notamment celle de l'énergie.

Chapitre 3

RepMChord : vers une approche de Chord-mobile plus efficace en énergie

1. Introduction

Le déploiement des DHT sur les MANET offre l'opportunité de bénéficier de l'efficacité de partage, stockage et localisation de données présentée par les DHT et de la mobilité sans infrastructure présentée par les réseaux MANET. Cette association présente les défis à prendre en considération tels que, la topologie dynamique, la bande passante limitée et l'énergie limitée.

En conséquence, notre approche associe le protocole Chord de DHT au réseau MANET, en proposant une nouvelle contribution de réplication de données qui sert à optimiser le temps de recherche, le taux de succès de la recherche et la consommation de l'énergie.

A notre connaissance, au vu de ce qu'on trouve dans la littérature, cette tentative qui utilise une méthode de réplication pour optimiser la consommation de l'énergie dans un protocole P2P mobile, est assez originale.

Ce chapitre, est consacré au développement de l'approche de réplication sur MANET appelée RepMChord. Dans un premier temps nous énumérons quelques observations sur les recherches existantes et sur ce qui concerne la DHT mobile et la réplication de données dans les DHT, afin d'expliquer les motivations qui nous ont conduit à proposer et à construire RepMChord.

Les travaux connexes les plus populaires de la réplication sur DHT, sont mentionnés et classifiés par catégories. Une discussion détaillée est élaborée à partir de leurs points faibles et de leurs avantages. Une description et une explication détaillée de l'approche proposée, étoffée des algorithmes et des figures utilisés est présentée et ce, en mentionnant les principales fonctionnalités de la méthode RepMChord.

2. Observations et motivations

Dans cette partie, nous expliquons le problème abordé dans cette approche en termes de critères suivants :

a) De churn (topologie dynamique)
b) De disponibilité des données

c) De trafic d'overhead, lors de maintenance ou de la recherche
d) D'énergie.

Les recherches sont orientées vers le déploiement de P2P sur les réseaux mobiles sans-fil en général et sur les MANET en particulier [daH09][Lee13][Kha14][Wou15][AlM15], afin de bénéficier de l'efficacité du partage de contenu et de la mobilité sans infrastructures. Les P2P structurés et les MANET partagent certaines caractéristiques en commun comme : (1) entièrement distribué où chaque nœud joue le rôle d'un client et d'un serveur au même temps, (2) auto-organisé contre les arrivées et les départs imprévisibles des nœuds (topologie dynamique), (3) auto-guérison contre les défaillances des nœuds, (4) les exigences de base sont l'évolutivité et l'efficacité de routage. De plus, comme les P2P sont destinés au premier lieu aux réseaux fixes filaires, ils présentent certains défis sur les MANETs qui doivent être pris en considération par les chercheurs, comme l'énergie et la bande passante limitées et la topologie dynamique [Cha08][Kel08][Cho12][Abi15].

Dans les DHT traditionnels, le fichier est identifié par le hachage de son nom pour obtenir ce qu'on appelle la *clé,* et le nœud est identifié par le hachage de son adresse IP pour obtenir un ID (identifiant). Cependant, la fonction de hachage utilisée, ex. SHA-1 ou SHA-3, doit être de bonnes propriétés (un faible risque de collision et une inversion impossible). La *clé* d'un fichier et l'ID du nœud qui stocke réellement ce fichier représentent la ressource ou la donnée qui sera stockée par la DHT. Chaque ressource a un nœud responsable unique appelé nœud de racine ou nœud maitre, qui a le plus proche identifiant (clé proche à ID). Dans Chord, quand un nœud veut quitter le réseau, il informe le successeur pour qu'il soit responsable de ses ressources, et informe le prédécesseur pour qu'il mette à jour son voisinage. Autrement, dans le cas d'un départ imprévisible de nœud, la ressource sera perdue, et donc les données ne sont pas toujours disponibles, ce qui est un problème connu, surtout dans la topologie hautement dynamique comme les systèmes mobiles sans-fil (ex. en raison des batteries épuisées).

Pour faire face à ce problème, les données doivent être répliquées dans plus d'un seul nœud (appelés nœuds de répliques) pour augmenter la disponibilité des données, la tolérance aux fautes, la fiabilité et l'évolutivité. Cette solution s'appelle la réplication de données.

Les solutions de réplication proposées sont motivées par l'augmentation de la disponibilité des données, la réalisation d'un équilibrage de charge ou la réduction de la latence de recherche. Outre ces avantages, la réplication de données génère plus de trafic réseau dans le processus de maintenance, où les nœuds de répliques mettent à jours leurs répliques (les données répliquées) à chaque fois qu'un nœud joint ou quitte le réseau (i.e. quand un processus de churn est effectué). Ces messages de mise à jour sont échangés entre les nœuds de répliques eux-mêmes ou avec le nœud racine, ce qui provoque un trafic réseau élevé. Cependant, le nombre de messages de maintenance dépend de la méthode de réplication utilisée, mais le trafic excessif n'est pas adaptable aux dispositifs mobiles avec énergie et bande passante limitées [AlM16].

Dans ce présent travail, nous proposons une approche de réplication qui vise à améliorer les performances de Chord, soit dans un environnement statique ou mobile. Cependant, nous nous intéressons par l'environnement mobile car il y a plus de points à prendre en considération, ceux qui sont causés par les caractéristiques de mobilité et de sans-fil. Par conséquent, pour s'adapter avec ces caractéristiques, nous nous intéressons dans notre travail par les points suivants :

- Réduire l'overhead de maintenance par rapport aux travaux de réplication existants. De plus, réduire l'overhead lors de processus de recherche par minimiser le nombre de pairs impliqués dans chaque requête de recherche (nombre de sauts), et donc le nombre de messages envoyés/reçus par chaque pair est minimisé. Cette réduction de l'overhead permet à l'approche de s'adapter avec le problème de limitation de bande passante et d'énergie dans les réseaux mobiles.
- Accélérer le processus de recherche, c.à.d. réduire la latence de recherche. Ainsi, la requête peut arriver au nœud de destination avant que ce dernier quitte le réseau à cause de la mobilité ou de l'épuisement de batteries, et donc ça permet de s'adapter avec la topologie dynamique.

Autrement dit, notre approche de réplication vise non seulement à accroitre la disponibilité des données mais aussi à minimiser le nombre de messages générés et à réduire la latence de recherche pour défaire face aux caractéristiques de MANET.

Cette approche paraît assez originale du fait qu'elle diffère des autres approches de réplication existantes dans : (1) la considération de trafic de maintenance, (2) la considération et la réduction de la consommation d'énergie, et (3) l'application d'une méthode de réplication sur un Chord mobile. A notre connaissance, l'utilisation de la réplication pour optimiser la consommation de l'énergie dans un protocole de P2P mobile, est une des toutes premières tentatives dans cette thématique de recherche.

3. Travaux connexes

Dans la littérature, il existe un nombre considérable de techniques de réplication proposées pour les P2P structurés. Ces techniques sont intéressantes dans l'optimisation de la disponibilité de données et la tolérance aux fautes dans les réseaux fixes sans déploiement, dans les environnements mobiles. Les approches existantes peuvent être classifiées en trois catégories en fonction de la méthode utilisée:

a) *Réplication de liste de voisinage:* dans cette méthode, le nœud réplique ses données dans les membres de sa liste de voisinage. Comme dans [Sto01], les auteurs proposent de répliquer dans les *r* nœuds de la liste des successeurs de Chord. Et donc, dans le cas d'un départ imprévisible d'un nœud, la requête sera résolue par son successeur immédiat. Une autre méthode de [Ben13] qui fait la réplication dans les *r* nœuds de la liste des prédécesseurs de Chord, dans le but de réduire la latence de recherche quand la requête arrive à l'un des prédécesseurs responsables avant d'arriver au nœud racine. Les auteurs de Pastry proposent de répliquer dans les membres de la liste de leaf-set (r/2 prédécesseurs et r/2 successeurs).

Concernant le processus de maintenance de cette catégorie, le churn implique le changement de plusieurs listes de voisinage, et donc un nœud joignant implique $(r \times 2)$ nœuds à échanger de

messages de mise à jour et un nœud quittant implique $(r \times 2) - 1$ nœuds, ce qui génère un trafic réseau élevé et présente un inconvénient considérable pour ces approches.

b) *Réplication de groupes associés :* dans cette méthode, chaque ressource est destinée à être répliquée dans un groupe avec un identifiant bien spécifique. Comme dans la solution de réplication symétrique [Gho07], les auteurs proposent une structure de donnée (formule) pour définir les nœuds de répliques correspondants à chaque ressource, dont la clé est associée avec d'autres *r* identifiants formant une classe (un groupe). Une autre solution appelée ID-Replication [Sha12], utilise la notion de groupe à la place de la notion de nœud. Comme cette solution est appliquée sur Chord, le groupe responsable de la ressource *clé* est celui avec ID, égal ou immédiatement supérieur à *clé,* puis tous les *r* nœuds de ce groupe répliquent cette ressource. Dans chaque groupe, les nœuds utilisent le gossiping pour synchroniser la vue du groupe, et les groupes entre eux utilisent périodiquement un processus de stabilisation pour rafraîchir les tables de routage.

Concernant le processus de maintenance de cette catégorie, le churn implique l'échange de messages de mise à jour entre les *r* membres du même groupe, ce qui est réduit par rapport aux approches de la catégorie précédente. La méthode d'ID-Replication invoque plus de messages de maintenance par le rajout des messages de stabilisation entre les groupes.

c) *La réplication de multi fonctions de hachage :* l'idée consiste à appliquer des différentes fonctions de hachage sur chaque ressource, pour avoir différentes *clés*, et puis assigner cette ressource aux différents nœuds de répliques en fonction de la *clé* calculée. On peut citer l'exemple d'un travail basé sur cette méthode, à savoir celui de [Sol12], où le nœud surchargé par la popularité d'un item A, applique une deuxième fonction de hachage sur A, pour avoir une nouvelle clé. Celle-ci est assignée à un nouveau nœud de réplique avec l'ID le plus proche. Dans le but, de réaliser un équilibrage de charge sur les requêtes de recherche en fonction de la popularité des items où la ressource avec une haute popularité va avoir plus de répliques.

3.1. Discussions

Chaque catégorie et chaque méthode en particulier présente des avantages ainsi que des inconvénients, mentionnés ci-après. La réplication de liste des successeurs évite d'avoir des requêtes perdues et donc d'overhead perdu. La requête destinée à un nœud partant sera résolue par l'un de ses successeurs. Cependant, elle présente certains inconvénients, comme le goulot d'étranglement chez le nœud maitre, puisque toutes les requêtes seront résolues par le nœud maitre dans un premier lieu. Ce problème est moins courant dans la réplication de leaf-set, puisque la requête peut être résolue par l'un des voisins et pas toujours par le nœud maitre.

Dans la réplication de la liste des prédécesseurs [Ben13], les auteurs minimisent ce problème de goulot d'étranglement, et réduisent la latence de recherche puisque c'est possible d'atteindre l'un des nœuds prédécesseurs avant le nœud maitre, mais ce n'est pas toujours le cas comme ils n'ont pas appliqué des changements sur le processus de recherche, et donc les nœuds ne peuvent pas savoir qui sont les nœuds de répliques des autres ressources dont ils ne sont pas responsables. Cette réduction de latence peut être réalisée par la réplication de leaf-set quand un nœud prédécesseur est choisi.

L'avantage des deux secondes catégories est de réduire le nombre de messages de mise à jour (de maintenance) par rapport à la première catégorie. L'inconvénient de la méthode symétrique est la nécessité de manipuler une structure de donnée complexe à chaque réplication et recherche de données. Cette méthode réalise un équilibrage de charge de facteur de réplication (le nombre de répliques) comme il est fixe pour toute ressource, différemment à ID-replication où ce facteur varie entre deux valeurs rMin et rMax. L'avantage de cette dernière est la non nécessité de manipuler une structure de donnée complexe, mais d'un autre coté elle génère plus de messages lors de la stabilisation des groupes et la réalisation d'équilibrage de charge entre eux.

L'inconvénient de la réplication de multi fonctions de hachage apparait quand un nœud demandeur lance une requête de recherche d'un item A, avec une clé *key* calculée par firstHash, et le nœud responsable correspondant n'est pas disponible (il est parti imprévisiblement). Donc, la requête sera perdue sans résultat. Ensuite, le nœud demandeur lancera une autre requête en

utilisant une nouvelle *key* avec une deuxième fonction de hachage secHach et ainsi de suite jusqu'à arriver à A, ce qui entraîne une perte d'overhead.

Généralement, l'inconvénient majeur des méthodes de réplication est l'overhead généré lors du processus de maintenance, quand les nœuds mettent à jour leurs répliques. Ce nombre de messages générés diffère d'une méthode à une autre. Pour cela, notre approche proposée vise en premier lieu à minimiser l'overhead généré par le processus de maintenance, et aussi par le mécanisme de recherche. De plus, la proposition permet d'éviter d'autres inconvénients susmentionnés.

4. L'approche proposée [Che17]

Dans la littérature, les approches de réplication proposées s'intéressent par l'augmentation de la disponibilité des données et la tolérance aux fautes face au churn. Ici, l'approche proposée s'intéresse non seulement à ces deux points, mais aussi à la réduction du nombre de sauts de la recherche (les pairs impliqués dans la recherche) et les messages de maintenance, afin de consommer moins d'énergie. Cette technique de réplication P2P qui s'intéresse aux nœuds mobiles et à la consommation d'énergie, nous l'appelons RepMChord (Replication in Mobile Chord).

L'objectif de RepMChord est de considérer les points de critères expliqués dans la section d'« observations et motivations » de ce chapitre, afin d'optimiser la disponibilité des données tout en évitant le trafic excessif et adapter ainsi le système au problème de churn. Nous supposons un système distribué avec un ensemble de pairs communiquant à l'aide des messages de send-receive en passant par un réseau de communication, comme suit :

- send (sender: receiver: Message(arg1,arg2,…))
- receive (sender: receiver: Message(arg1,arg2,…))

Les variables utilisées dans la suite sont décrites dans le tableau ci-dessous :

Tableau 3.1 Description des variables utilisées.

Variable	*Description*
ID	Identifiant d'un nœud
Key/clé	Identifiant de ressource
N	La taille réseau : nombre des nœuds
m	La longueur des identifiants en bits, appelée l'espace d'adressage.
r	Le nombre de répliques d'une seule ressource, appelé : facteur de réplication

Dans RepMChord, une ressource avec un identifiant *key* est stockée dans un nœud maître *n* qui est le nœud successeur le plus proche à *key* comme dans Chord de base. De plus, *key* sera répliquée dans *r* nœuds prédécesseurs définis par la formule (1) et qui peuvent être trouvés dans la liste des prédécesseurs de *n*, qui sont les nœuds avec un ID appartenant à l'intervalle $[key - r(mod\ m), key]$. Le paramètre *r*, s'appelle le facteur de réplication égal à $m - 1$, et *m* est l'espace d'adressage de l'anneau de Chord. Comme montré dans la figure 3.1, la clé 10 est stockée dans le nœud 10 et répliquée dans les nœuds : 9, 8 et 7.

$$respNode = (key - r)\ mod\ m \qquad \dots Formule\ (1)$$

Tel que : $1 \leq r \leq m - 1$

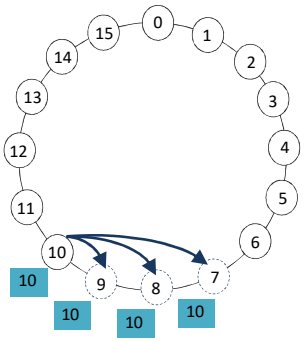

Figure3.1 La réplication proposée, appliquée sur la clé 10 (m=4, r=3)

L'exemple suivant montre comment travailler avec la formule (1) :

```
for (i=1 to r) do {
//Le nœud responsable est celui avec ID égal à ((key-r)mod m)
respNodeID:= (key-i)mod m              ... formule(1)
```

```
//Le nœud maître n informe le respNode par les ressources dont il est responsable.
send(n:respNode:addkey(key,value))
}
```

Un point important, est que *m* dans RepMChord, est défini en fonction de la taille réseau N, i.e. quand le réseau est large, *m* va avoir une haute valeur correspondante à N (et l'inverse, quand N est petit), de façon que la valeur de *m* soit égale approximativement ou précisément au logarithme binaire de N ($m = log_2 N$), afin d'avoir des voisins d'overlay avec des valeurs d'ID proches. Par exemple, quand N=70 nœuds (pairs), la valeur de *m* sera définie par 6, puisque $log_2(70) = 6$. De plus, pour s'adapter à la grande échelle, nous ne définissons pas *r* par une valeur fixe comme dans [Gho07] mais elle est ajustée en fonction de la taille réseau N et donc l'espace d'adressage *m*, tel que $r = m - 1$.

Dans le but d'éviter les messages de mise à jour causés par le changement de la liste de prédécesseurs (liste de voisinage) à chaque fois qu'un prédécesseur quitte ou joint le réseau, différemment à [Ben13] nous ne répliquons pas dans les prédécesseurs successifs du nœud maitre (les membres de la liste des prédécesseurs), mais dans les prédécesseurs définis par la formule (1). Ces derniers peuvent être une partie ou une totalité de la liste des prédécesseurs dépendant de leur disponibilité dans le réseau. Donc, le changement de la liste des prédécesseurs n'entraine pas un contact entre tous les anciens nœuds de répliques et les membres de la nouvelle liste de voisinage. Dans le cas d'un nouveau nœud prédécesseur, quand il est informé par le nœud maitre sur les ressources dont il est responsable, il sera ainsi informé sur les répliques dont il est responsable en fonction de la formule définie, et ça n'entraine pas de nouveaux messages. Autrement dit, avec l'arrivée de la nouvelle *clé* à stocker, le nœud maître calcule l'ID du nœud de réplique selon la formule (1) et cherche dans sa liste de prédécesseurs si ce nœud existe ou pas. Ce qui signifie qu'il est disponible dans le système et puis, il lui transmet la réplique à stocker. Par ailleurs, si ce nœud n'existe pas dans la liste des prédécesseurs, ce qui signifie qu'il n'est pas disponible dans le système (déconnecté), on ne le remplace pas par un aucun autre nœud, seulement si un nœud avec cet ID se connecte au système. Par conséquent, la réplique sera stockée sur le nœud avec exactement le même ID calculé et non sur l'un de ses voisins. Cette idée sera mieux clarifiée dans la suite de cette section.

Un autre avantage de l'utilisation d'une formule mathématique pour définir le respNode (le nœud responsable) est que tous les nœuds dans le système peuvent savoir qui est le nœud maître et qui sont les nœuds de répliques par l'usage de la formule, et peuvent choisir le plus proche en termes de la valeur d'ID la plus proche. Même si le nœud responsable le plus proche (le prédécesseur) n'est pas disponible pour le moment, la requête sera transmise à son successeur comme dans Chord de base, et donc nous évitons les requêtes perdues, comme indiqué dans la figure3.2, où le respNode le plus proche de la clé 10 est 7, qui n'est pas disponible, et donc la requête de recherche est transmise à son successeur 8.

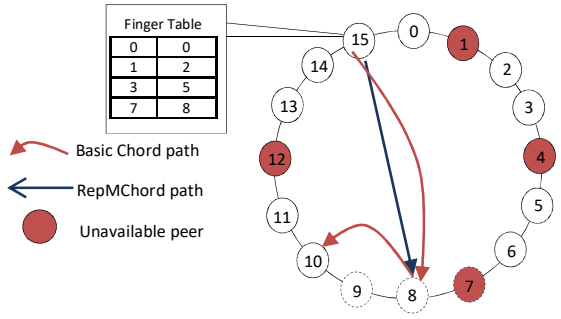

Figure3.2 Le nœud 15 choisit le nœud responsable disponible le plus proche de la clé 10.

Les auteurs de la réplication symétrique [Gho07], utilisent une formule pour calculer les répliques et puis définissent le nœud successeur de la réplique *key* comme un responsable (nœud de réplique). Par exemple, cette formule définit le nœud 11 comme un nœud de réplique de key=3, si ce nœud 11 n'est pas disponible donc l'un de ses successeur comme le nœud 12 sera le nœud réplique de key=3, et donc le nœud 12 sera un nœud de réplique pour un nombre illimité de clés, i.e. le nœud 12 sera inclus dans plus d'un groupe de réplique, ce qui réduit l'équilibrage de charge des zones de responsabilités et augmente le nombre de messages de mise à jour envoyés et reçus par ce nœud. Cependant, notre contribution évite de définir de nœud de réplique par son successeur, mais par le nœud qui a le même ID comme défini par la formule (1), comme on réplique dans le nœud prédécesseur et donc son nœud successeur est déjà responsable de cette *key*.

Algorithme1: Le stockage d'une nouvelle clé

```
1. key= hash(resource)
2. masterNode= succ(key); //Le successeur de key
3. for (i=1 to r) do {
   respNodeID:= (key-r)mod m //celui avec ID=((key-r)mod m)
   send(masterNode:respNode:addKey(key,value))
   }
```

- Espace de stockage : Tout d'abord, nous spécifions que lorsqu'on parle de l'espace de stockage dans les DHT, on signifie le stockage des clés et leurs valeurs associées (ex. information de localisation de cette clé), et donc ce n'est pas le stockage des fichiers entiers. Concernant l'espace de stockage supplémentaire des méthodes de réplication, il dépend du facteur de réplication *(r)* de la méthode utilisée, qui est le nombre des nœuds de répliques de chaque ressource. Par conséquent, l'espace de stockage des méthodes de réplication est supérieur à celui de Chord de base sans réplication mais il est le même dans notre méthode que dans les autres travaux de réplication populaires comme : réplication de liste des prédécesseurs [Ben13] et la réplication de liste des successeurs [Sto01] puisque *r* dans ces travaux égal à : $m - 1 = log_2 N - 1$ (*N* est la taille réseau). Dans d'autres travaux comme [Sol12], l'espace de stockage peut être plus haut ou plus bas puisque le facteur de réplication augmente avec l'augmentation de popularité de la ressource. En outre, dans les travaux de réplication de [Sha12] et [Tri16] le facteur de réplication n'est pas déterminé par une valeur fixe ou une heuristique bien définie, puisque le réseau est divisé en zones et les nœuds membres d'une seule zone stockent les même répliques, et donc la valeur du facteur de réplication varie selon le nombre de membres de chaque zone.

La Figure 3.3 montre une vue globale de RepMChord. Pour assurer l'efficacité de l'approche nous nous intéressons principalement aux cinq processus suivants :

- Le stockage d'une nouvelle ressource.
- L'arrivée d'un nouveau nœud.
- Le départ du nœud.
- Le processus de recherche.
- La mobilité du nœud.

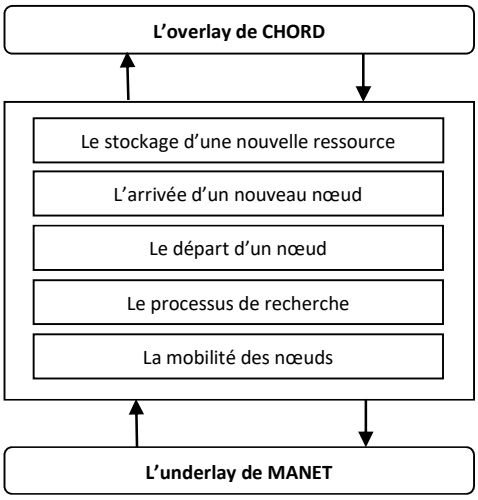

Figure3.3 Une vue globale sur l'approche proposée.

4.1. L'arrivée d'un nouveau nœud

Avec l'arrivée d'un nouveau nœud *n* dans Chord de base et après l'obtention de son ID, *n* contacte son successeur pour récupérer les clés dont il est responsable, ce qu'on appelle les clés originales (original keys) de *n*. Dans RepMChord, le successeur fait la même chose comme dans le Chord de base, en plus d'informer le nouveau nœud de ses répliques correspondantes. Cela n'exige aucun nouveau message car le successeur les possède dans sa zone de responsabilité et utilise le même message de addKey. C'est comme l'exemple présenté dans la figure3.4(b), où le nœud 12 joint le système et contacte son nœud successeur 14. Puis, le nœud 14 informe le nœud 12 de sa clé originale qui est key=12 et ses répliques qui sont keys= (13, 14 et 15) qui sont déjà stockées dans le nœud 14, comme montré par la figure.

Algorithme2: L'arrivée d'un nouveau nœud

```
1.//Le successeur cherche dans sa zone de responsabilité, toutes les clés
  dont le nouveau nœud est responsable
2.for (i=0 to zoneResp.getSize-1) do {
3.//Trouver les clés originales et répliquées, correspondant au nouveau nœud
  (predID)
4.if (zoneResp.getKey.isBetween]actualNode.predID,actualNodeID+r]){
5.correspondingKeys[i]=zoneResp.getKey;
6.}
7.//Le successeur informe le nouveau nœud sur ses clés correspondantes.
8.send(succ:newNode:addKey(correspondingKeys))
```

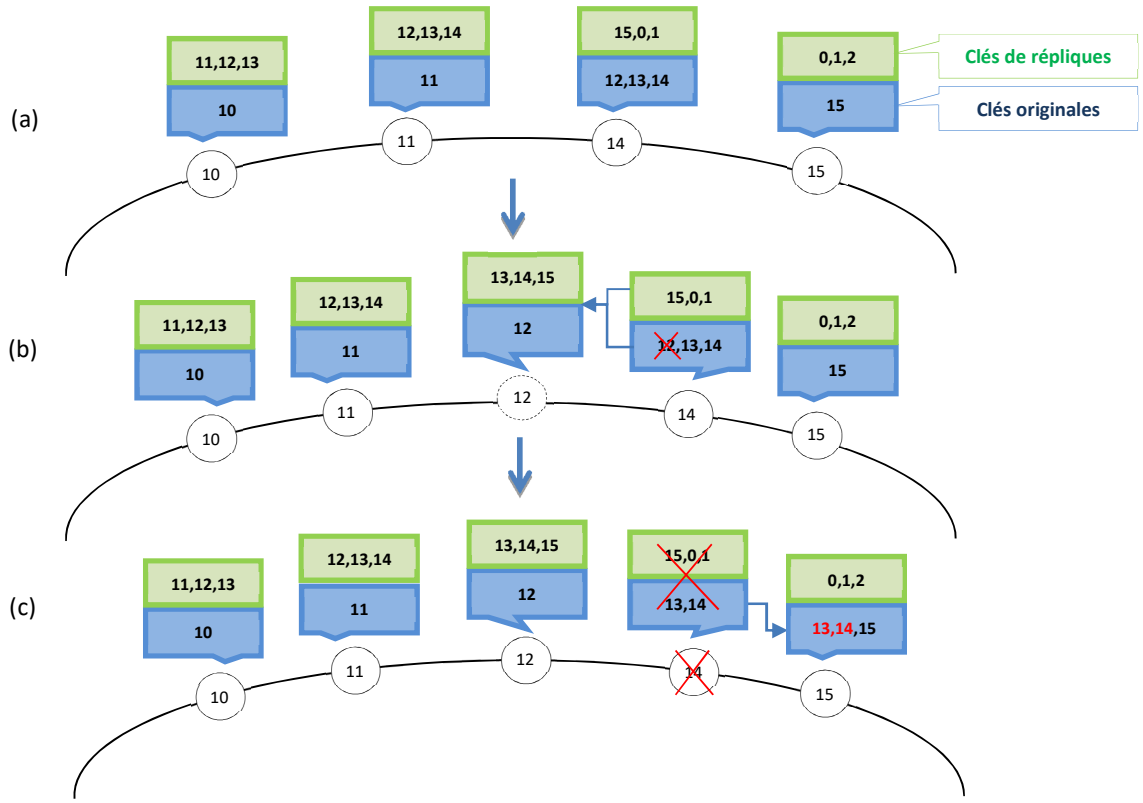

Figure3.4 (a) Une partie de système de Chord dans l'état stable où chaque nœud stocke ses clés originales et répliquées, m=4 et r=3; (b) l'arrivée du nœud 12; (c) le départ du nœud 14.

4.2. Le départ d'un nœud

Chaque nœud dans le système est responsable de certaines clés, comme « nœud maître » s'il est le successeur le plus proche de cette *key* (appelée clé originale), et comme « nœud de réplique » si son ID correspond à la formule (1) de cette *key* (appelée réplique). Quand un nœud quitte le système, il stocke ses clés originales dans son successeur comme dans Chord de base. Mais il ne stocke pas ses répliques dans aucun autre nœud, en raison de l'idée de stocker la réplique dans le nœud avec exactement le même ID calculé par la formule (1) et non dans l'un de ses voisins comme nous l'avons proposé dans notre méthode. Ce principe est une sorte de

tradeoff entre perdre une réplique et éviter les messages excessifs de mise à jour présentés dans la réplication de liste de voisinage [Row01b] [Ben13]. De plus, la probabilité de perdre toutes les répliques de *key* ou un grand nombre d'elles, est égale à la probabilité de n'avoir aucun nœud avec ID appartenant à l'intervalle [key-r, key[, qui est une probabilité ignorée surtout quand *m* (le bit d'adressage) est égal à log_2N, dont *N* est la taille réseau, soit, avoir des nœuds avec des valeurs d'ID proches, comme il a été expliqué en début de cette section.

La Figure3.4 (c) présente un exemple d'un processus de départ du nœud 14, où il envoie ses clés originales [keys= (13 et 14)] à son successeur pour être le nouveau responsable, mais il n'envoie pas ses répliques (15, 0 et 1) à aucun autre voisin.

Algorithme3: Le départ d'un nœud

```
1.//Parcourir la zone de responsabilité du nœud actuel quittant le réseau
2.for (i=0 to zoneResp.getSize-1) do {
3.//Trouver les clés originales
4.if (zoneResp.getKey.isBetween]actualNode.predID,actualNodeID]
5.originalKeys[i]=zoneResp.getKey;
6.}
7.//Envoyer les clés originales au nœud successeur
8.send(actualNode:actualNode.succ:addKey(originalKeys))
```

La figure3.4 (b) (c) montre que les processus d'arrivée et de départ d'un nœud invoquent le même nombre de messages de mise à jour comme dans Chord de base.

4.3. Processus de recherche

Dans RepMChord, nous utilisons la table de finger originale de Chord de base, mais nous ne suivons pas le processus de recherche original (différemment à [Ben13] [Gho07]), dans le but de tirer plus d'avantages de la méthode de réplication proposée. Ceci est expliqué comme suit :

- En premier lieu, le demandeur cherche dans sa zone de responsabilité, s'il est responsable de cette *key* (comme nœud maitre ou de réplique) en utilisant un intervalle défini à l'aide de la formule proposée (Algorithme4),
- En deuxième lieu, si ce n'est pas le cas, il cherche dans sa liste des successeurs l'un des nœuds responsables de *key,* et choisit celui le plus proche avec la valeur d'ID minimale si plusieurs responsables sont trouvés,
- En troisième lieu, si aucun nœud responsable n'est trouvé, le demandeur cherche dans sa table de finger, le nœud prédécesseur le plus proche de $(key - r)$, afin de trouver le responsable disponible le plus proche, sinon trouver son nœud prédécesseur le plus proche qui va exécuter le même algorithme dès le début.

Ces points sont repris et expliqués dans l'algorithme 4.

Contrairement à [Gho07], où le nœud demandeur et les nœuds impliqués dans le processus de recherche doivent manipuler une structure de données complexe à chaque récupération d'une clé, ce qui présente un inconvénient. Dans notre technique nous remplaçons tout cela par une simple vérification: si le nœud actuel veut savoir s'il est responsable, il vérifie si la clé recherchée appartient à l'intervalle :]actualNode.predecessorID, actualNodeID + r].

Algorithme4: Le processus de recherche

```
1. //Le demandeur vérifie s'il est responsable de key
   IsResponsibleFor(nodeID,key) {key.isBetween]nodeID.predID,NodeID+r]}
2. //chercher dans la liste des successeurs, le responsable le plus proche
   for (i=0 to succList.getSize-1) do {
   if (IsResponsibleFor(succList.getSuccID,key))then
   respNode=succList.getSuccID
   else
   respNode.isUnspecified
}
```

```
3. //chercher dans la table de finger, le responsable le plus proche ou son
nœud prédécesseur
   for (i=fingerTable.getSize-1 downto 0) do {
   if (fingerTable.getFingerID.isBetween[nodeID,key-r]) then
   send(nodeID:fingerTable.getFinger:LookupRequest(key))
}
```

La Figure3.2 présente aussi un processus de recherche lancé par le nœud 15 pour chercher key=10, où dans Chord de base, la recherche prend 2 sauts du nœud 15 au nœud 8 et puis au nœud 10. Dans notre proposition de Chord, elle prend seulement 1 saut du nœud 15 au nœud 8, qui a aussi choisi le responsable le plus proche et donc un chemin plus court.

Si on utilise le processus de recherche originale de Chord à l'aide de notre technique de réplication, on va chercher directement dans la table de finger, le nœud maitre ou son prédécesseur proche, ce qui permet de trouver directement le nœud maitre. Ce qui fait qu'on ne réduit pas le chemin de recherche par le fait de manquer le prédécesseur responsable le plus proche du nœud demandeur.

4.4. La mobilité des nœuds

Lors du traitement de la mobilité des nœuds, on peut considérer deux cas:

- Le premier cas se présente quand la mobilité entraine un changement dans seulement, la topologie physique i.e. une perte de connexion entre les nœuds physiques, mais sans changement de l'adresse IP. Ce qui exige un réajustement dans le réseau physique par l'utilisation d'un protocole de routage, comme par exemple OLSR dans MANET.
- Le deuxième cas, se présente quand la mobilité entraine un changement de l'ID logique du nœud (ex. changement de l'adresse IP), ce qui exige un réajustement de la topologie logique d'overlay, pris en charge par le processus de maintenance de churn, puisque le nœud mobile ici est considéré comme un nœud quittant.

5. Conclusion

L'approche proposée baptisée RepMChord est motivée par l'observation de plusieurs points, tels que : la perte de données, notamment dans le cas des réseaux mobiles où le nombre de possesseurs d'une ressource est très faible et les travaux de réplication qui ne prenant pas en considération le trafic réseau et l'énergie consommée.

Cette approche permet d'optimiser non seulement la disponibilité de données dans le Chord mobile mais aussi, elle permet de réduire le nombre de messages en circulation et optimise ainsi l'énergie consommée. Chacune des fonctions de l'approche est suffisamment détaillée à l'aide de figures et d'algorithmes qui aideront aisément le lecteur de cette thèse à la compréhension de la méthode. La mise en œuvre de cette approche fait l'objet du chapitre qui suit.

Chapitre 4

Implémentation et validation de RepMChord

1. Introduction

Après avoir détaillé la solution de réplication de Chord sur MANET, appelée RepMChord, nous allons présenter la mise en œuvre de cette stratégie, sous le simulateur réseau Omnet++ et les deux plateformes Oversim et Inet.

Avant d'entamer l'implémentation, il est nécessaire de présenter les hypothèses de simulation utilisées puis nous allons décrire les métriques des performances utilisées pour prouver l'efficacité de ce travail. Ces résultats sont ensuite comparés à ceux obtenus à l'aide de Chord de base mobile et à ceux obtenus dans un autre travail où les performances sont prouvées et établies. Les résultats obtenus sont présentés sous forme des graphes d'évaluation et de comparaison avec les deux travaux mentionnés ci-dessus.

2. Environnement et paramètres de travail

Pour évaluer les performances de l'approche proposée, nous avons implémenté RepMChord sous les utilitaires suivants :

- Le simulateur réseau Omnet++4.6 [Var08] [Omn14],
- La plateforme Oversim [Bau09] pour l'overlay,
- La plateforme Inet [Ine12] pour *l'underlay*.

Comme les protocoles d'overlay existants ne sont disponibles seulement sur les réseaux fixes filaires, nous avons commencé par implémenter l'overlay Chord sur *l'underlay* MANet en utilisant OLSR [Jac01] comme protocole de routage et Ieee 802.11, comme interfaces. L'implémentation a été effectuée avec toutes les fractions correspondantes expliquées dans le chapitre précèdent.

Nous avons comparé la méthode proposée avec l'une des approches de réplication les plus récentes, qui est la réplication de liste de prédécesseurs (PredChord) [Ben13]. Cette dernière a prouvé son efficacité en termes de réduction de la latence de recherche par rapport à d'autres travaux. Afin de prouver l'efficacité de l'approche proposée sur MANet sans *overhead* excessif,

nous la comparons à l'approche de Chord de base appliquée sur MANet, qui ne génère pas d'*overhead* de maintenance de répliques.

Les étapes de simulations ainsi que les résultats correspondants sont publiés dans [Che17].

Paramètres de simulation :

- Pour *l'overlay*, nous utilisons le mode d'arrivée agressif *(agressive join mode)* avec un délai de 10s, un processus de stabilisation chaque 20s et une vérification du nœud prédécesseur chaque 5s.
- L'évaluation est effectuée avec une taille réseau variable et dans certains cas avec une durée de vie de *churn* variable.
- La vitesse de mobilité de l'*underlay* est définie entre 8mps et 20mps, avec une distribution truncnormal.

Ces paramètres sont illustrés dans le Tableau ci-dessous :

Tableau 4.1 Paramètres de simulation.

Paramètre	Valeur
Taille réseau	10…40…80…120
Protocole de routage MANET	OLSR
Temps de simulation (s)	1000
Vitesse de mobilité	Truncnormal (20mps, 8mps) = (72kph, 28.8kph)
Temps de vie de Churn (s)	500…1000…5000…10000
Délai d'arrivée des nœuds de Chord (s)	10
Délai de stabilisation de Chord (s)	20
Délai de vérification des preds (s)	5

3. Métriques de performances

Les métriques de performances utilisées pour prouver l'efficacité de RepMChord sont les suivantes :

- *Nombre de sauts de recherche :* le nombre moyen de sauts ou pairs parcourus par chaque processus de recherche depuis le lancement de la requête par le demandeur jusqu'à l'atteinte de nœud cible.

- *La latence de recherche :* le temps moyen pris par chaque requête de recherche émise dans le système.
- *Requêtes de recherche échouées :* le taux moyen de requêtes non-résolues. Ça peut être en raison de perte de données, un départ imprévisible des nœuds ou une entrée expirée de la table de finger.
- *Messages de maintenance :* le nombre moyen de messages de contrôles circulés dans l'***overlay***. Ces messages sont utilisés par chaque pair pour maintenir les tables de finger, les listes de successeurs ou les répliques (si une méthode de réplication est utilisée) dans le cas de churn ou d'une manière périodique pour s'adapter à la dynamicité réseau.
- *Énergie consommée :* l'énergie consommée par chaque pair ou l'énergie moyenne consommée par le réseau entier, ce qui reflète la durée de vie de réseau.

4. Résultats expérimentaux

4.1. Nombre de sauts de recherche

Le nombre de sauts de recherche est l'un des métriques les plus importantes utilisées pour prouver l'efficacité d'un nouveau protocole DHT en comparaison avec d'autres approches existantes. Ici, nous comparons le nombre de sauts moyen de notre approche avec Chord de base et avec l'approche de réplication de liste des prédécesseurs (PredChord). Ce nombre est donné par :

$$\mathrm{H} = (\sum_{i}^{L} h_{\mathrm{i}})/\mathrm{L} \quad \ldots \textit{Equation (A)}$$

Tel que : L est le nombre total de requêtes de recherche et h_i est le nombre de sauts parcourus par chaque requête.

La figure4.1 présente le changement de nombre de sauts dans les trois approches en variant la taille réseau. Nous remarquons d'abord que le nombre de sauts augmente avec l'augmentation de la taille réseau, ce qui est logique avec plus de pairs à parcourir par les requêtes de recherche.

Cependant, le nombre de sauts de recherche dans l'approche proposée est inférieur à celui de Chord de base mobile, ce qui prouve que la nouvelle méthode est plus adaptable au churn et

MANet (par atteindre le nœud cible avant qu'il quitte le réseau) et ça montre l'effet de la stratégie de réplication proposée sur la réduction du chemin de recherche.

De plus, nous expliquons, la petite réduction de nombre de sauts par rapport à l'approche de PredChord par la modification proposée appliquée sur le processus de recherche. Ce qui permet de mieux bénéficier de la méthode de réplication au lieu d'utiliser la recherche traditionnelle. En notons que PredChord est un travail qui a prouvé son efficacité dans la réduction de chemin de recherche par rapport aux travaux de réplication existants.

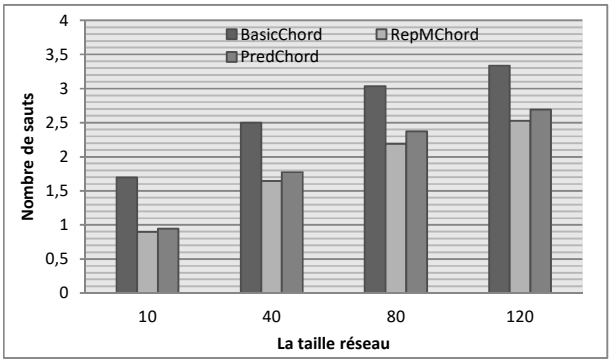

Figure4.1 le nombre de sauts moyen de recherche en fonction de la taille réseau.

4.2. La latence de recherche

La latence moyenne de recherche ou le délai de découverte de ressource est donné par:

$$T = (\sum_{i}^{L} t_i)/L \quad \text{... } \textit{Equation (B)}$$

Tel que : L est le nombre total de requêtes de recherche et t_i est le temps pris par une requête de recherche depuis la source jusqu'au nœud cible.

Réduire le temps de recherche ou la latence est l'une des exigences les plus importantes dans les protocoles de DHT, ce qui est réalisé par RepMChord et montré par la figure4.2. Nous remarquons que le temps de recherche de l'approche proposée est plus réduit que les deux autres approches, ce qui est expliqué par l'accroissement de la disponibilité de données par rapport à basicChord mobile, et la capacité de choisir le nœud responsable le plus proche en utilisant le mécanisme de recherche proposé par rapport à l'approche de PredChord.

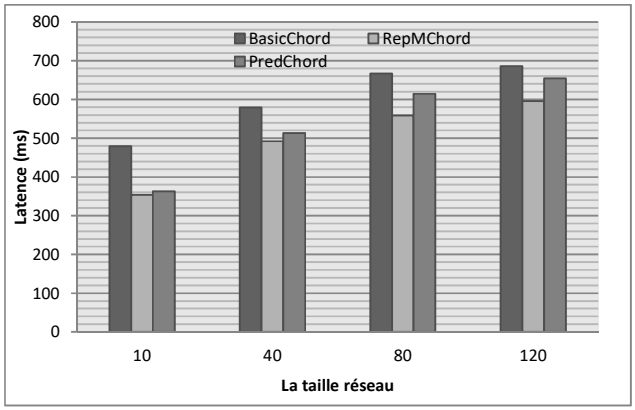

Figure4.2 La latence moyenne de recherche en fonction de la taille réseau.

4.3.Taux d'échec des requêtes

Ici, nous comparons le taux (le pourcentage) de requêtes de recherche non résolues. Ce pourcentage est donné par :

$$p = (fail \times 100)/L \quad \dots \textit{Equation (C)}$$

Tel que : *fail* est le nombre de requêtes de recherche échouées et *L* est le nombre total de requêtes de recherche émises.

L'augmentation de la taille réseau entraine un trafic réseau élevé, des collisions sérieuses et donc des problèmes de perte de paquets, ce qui explique l'augmentation de taux d'échec avec l'accroissement de nombre de nœuds, comme montré par la figure4.3.

Nous remarquons depuis la figure4.3 que le taux d'échec des requêtes dans RepMChord est considérablement inférieur que dans basic-Chord mobile et aussi inferieur que dans l'approche de liste des prédécesseurs, ce qui est expliqué par l'augmentation de la disponibilité de données et par conséquent le nombre de possesseurs de la même ressource augmente, et donc réduire la probabilité de perdre une ressource. De plus, notre modification appliquée sur le processus de recherche permet à la requête d'arriver à un des nœuds prédécesseurs, et si ce nœud quitte imprévisiblement, la requête sera résolue par l'un de ses successeurs, différemment à PredChord où la requête peut être destinée directement au nœud maitre en utilisant le processus de recherche

traditionnel, et donc si ce dernier n'est plus connecté, la requête ne peut pas être résolue par l'un de ses successeurs puisque ils ne sont pas des nœuds de répliques de la ressource recherchée.

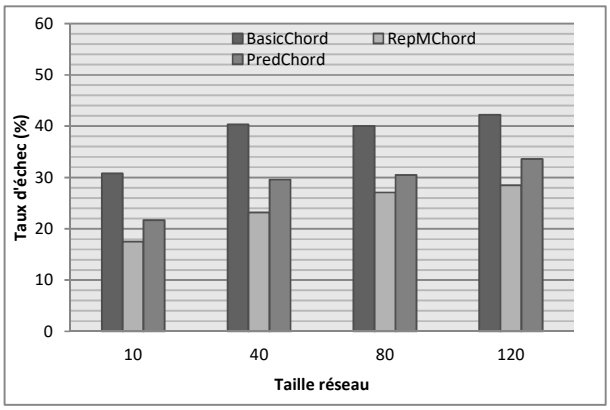

Figure4.3 Taux d'échec en fonction de la taille réseau.

4.4. Les messages de maintenance

Généralement, les messages de maintenance existants dans les méthodes de réplication appliquées sur Chord, sont ceux de mise à jour des listes de successeurs et des tables de finger comme dans Chord de base, plus les messages de mise à jour des répliques (des données répliquées).

Dans la figure4.4, nous varions les valeurs de churn pour définir une durée de vie de nœud entre (500s, 1000s, 5000s et 10000s) et nous fixons la taille réseau à 40 nœuds, où chaque nœud quitte le réseau après avoir passé, cette durée depuis sa création. Par exemple, dans la première phase : tout nœud quitte le réseau après avoir passé 500 secondes depuis sa création, c'est ce qu'on appelle la durée de vie du nœud.

Selon la figure4.4, en premier lieu nous remarquons que les messages de maintenance décroisent avec l'augmentation de la durée de vie de churn, puisque quand on a une petite valeur de durée de vie on aura plus de nœuds quittant le réseau et donc plus de messages de mise à jour pour réajuster la topologie d'overlay.

En second lieu, nous remarquons que RepMChord utilise presque le même nombre de messages de maintenance que celui de basicChord mobile et un nombre considérablement inférieur par

rapport à la méthode de liste des prédécesseurs, ce qui est expliqué par l'efficacité d'utilisation d'une formule mathématique pour définir les nœuds de répliques au lieu d'utiliser les listes de voisinage, et donc le nœud quittant/joignant ne nécessite pas la participation de tous les voisins $(r \times 2)$ dans la mise à jour des répliques. Cela nécessite seulement le processus de maintenance traditionnel appliqué dans Chord de base dans le cas de churn. De plus, la similarité du nombre de messages de maintenance entre Chord de base mobile et notre approche, est le résultat de l'utilisation des mêmes messages pour la mise à jour des listes des successeurs et des tables de finger (comme expliqué déjà dans la section 4 du chapitre 3). Même quand le successeur assigne une clé de réplique à un nouveau nœud joignant, il utilisera le message déjà utilisé dans Chord de base pour assigner les clés correspondantes (le message de AddKey comme expliqué déjà dans la section 4- L'arrivée d'un nouveau nœud- du chapitre précèdent).

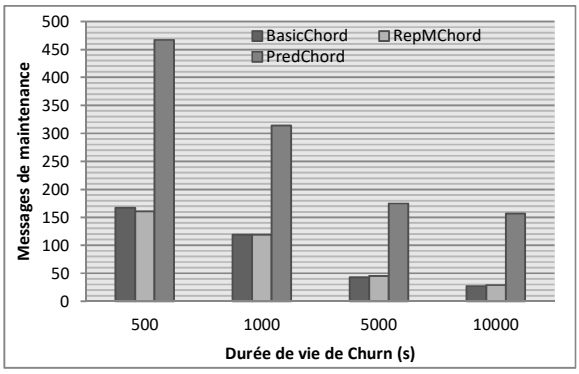

Figure4.4 Messages de maintenance en fonction de la durée de vie de churn.

4.5. L'énergie consommée

L'énergie consommée d'un nœud est en rapport direct avec le nombre de messages envoyés/reçus par ce nœud, ceux du processus de recherche et de maintenance (i.e. un nœud avec plus de messages envoyés/reçus consomme plus d'énergie). Le rapport entre ces deux facteurs (l'énergie et le nombre de messages) peut être exprimé par l'équation suivante :

$$ACE = (\sum_{i=1}^{N}(nbM_i \times c))/N \quad \dots \textit{Equation (D)}$$

Tel que : ACE est l'énergie moyenne consommée dans le réseau, nbM_i est le nombre de messages envoyés-reçus par le nœud i, c est le cout énergétique associé à un message et N est le nombre total de nœuds.

La Figure 4.5, présente l'énergie moyenne consommée dans un réseau avec une taille variable de 10 à 120, qui est une métrique très importante dans les réseaux mobiles sans-fil comme MANet, afin d'augmenter la durée de vie du réseau autant que possible.

La Figure 4.5, montre que les nœuds de RepMChord consomment moins d'énergie que dans Chord de base et de travail de la réplication de liste des prédécesseurs PredChord, ce qui clarifie l'efficacité de la méthode de réplication proposée dans la réduction du nombre de sauts et donc la réduction du nombre de nœuds impliqués dans chaque processus de recherche. Par conséquent, on observe la réduction du nombre de messages envoyés/reçus par chaque nœud.

La considérable différence de l'énergie consommée par rapport à la méthode de PredChord, s'explique par l'efficacité de notre méthode dans la réduction de messages de maintenance et la non génération d'un trafic excessif.

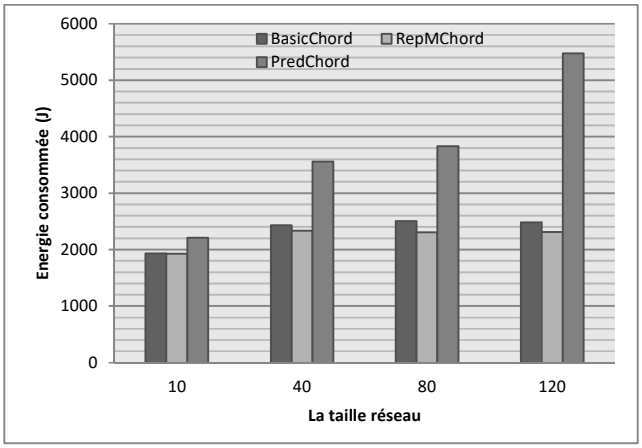

Figure 4.5 L'énergie moyenne consommée en fonction de la taille réseau.

5. Conclusion

L'implémentation mise en œuvre sur le simulateur Omnet++ et les plateformes Oversim et Inet, et les résultats que nous avons tirés, nous ont permis de tester et de valider notre contribution, et ce en la comparant avec :

- Chord de base mobile (BasicChord),
- et avec le travail de PredChord qui a prouvé ses performances dans l'optimisation de la recherche.

D'après les résultats obtenus, RepMChord présente de bons résultats en termes de réduction de chemin de recherche et d'augmentation de taux de succès de la recherche.

La non génération d'un nombre excessif de messages dans le processus de maintenance différemment à PredChord, permet à RepMChord de réduire la consommation de l'énergie.

Autrement dit, les résultats montrent que la technique proposée et son processus de recherche correspondant arrivent à :

- réduire le chemin de recherche (la latence et le nombre de sauts),
- augmenter le taux de succès de la recherche,
- et optimiser la disponibilité de données, tout en évitant l'inconvénient majeur des stratégies de réplication qui est l'overhead excessif généré.

Tous ces avantages ont contribué a optimisé la consommation de l'énergie. En conséquence nous avons obtenu un overlay Chord plus adaptable à l'underlay MANET, qui rappelons-nous, était le challenge ou l'objectif principal de ce travail.

Conclusion générale

Dans ce travail de thèse, nous nous sommes intéressés à l'adaptation du DHT aux réseaux mobiles MANET en prenant en considération la consommation de l'énergie des nœuds. Nous nous sommes fixés comme objectifs de trouver des réponses aux deux problématiques qui sont : comment stocker les données répliquées sur le réseau logique de façon à ce que les nœuds du système soient au courant de ces emplacements, et quel est le mécanisme de recherche à suivre afin de profiter de la stratégie de stockage proposée?

Pour analyser le problème de réplication et de stockage de données répliquées dans l'overlay, nous avons étudié les différentes propositions existantes telles que la réplication de liste des voisins, de groupes associés et de multi fonctions de hachage. Ces solutions existantes améliorent certes les performances du réseau, mais apportent à leur tour de nouveaux problèmes décrits dans la section 6.2.3 du chapitre 2, et la section 3.1 du chapitre 3.

Nous avons proposé une nouvelle approche de réplication, basée sur une formule mathématique pour déterminer l'emplacement des données dans l'overlay. L'utilisation de cette formule nous a permis d'atteindre les objectifs suivants :

- Éviter le trafic excessif généré par les approches de réplication existantes lors de processus de maintenance des répliques, et surtout celui généré par les méthodes de liste des voisins, où l'arrivée ou le départ d'un seul nœud entraine le changement des listes de voisinage de tous les voisins correspondants. Ce qui nécessite la maintenance de toutes les répliques stockées sur ces listes.
- Tout nœud peut appliquer la formule mathématique développée. Celle-ci lui permet de déterminer le nœud où la nouvelle ressource devra être stockée, quels sont les nœuds responsables d'une ressource donnée, et quel est le nœud responsable le plus proche.

Des modifications apportées au niveau du processus de recherche contrairement aux approches de réplication existantes qui utilisent le processus de recherche traditionnel du protocole utilisé (Chord, pastry, …), nous ont permis d'assurer l'efficacité de cette approche d'une part et de bénéficier de la stratégie de réplication, d'autre part.

Ces modifications ont permis :

- au nœud chercheur de choisir le nœud responsable le plus proche parmi les nœuds responsables existants, et donc de réduire le chemin et le temps de la recherche,
- de réduire le taux d'échecs des requêtes de recherche circulant dans l'overlay, grâce au mécanisme de choix des nœuds de répliques parmi les nœuds prédécesseurs du nœud maître, et donc si l'un de ces prédécesseurs est parti, la requête sera résolue par son successeur immédiat.

Nous avons déterminé la démarche des processus de : arrivée d'un nouveau nœud, départ d'un nœud, et gestion de la mobilité de manière à ce qu'ils soient adaptables à la stratégie proposée. Ces modifications appliquées sur ces processus sont indispensables pour avoir une contribution plus complète de Chord. L'originalité de notre approche réside dans sa simplicité, l'efficacité de recherche et son faible coût énergétique. C'est une des premières approches de réplication qui s'intéresse par l'optimisation de la consommation d'énergie dans un protocole P2P mobile.

Cette approche implémentée sous « Omnet++ » sous « Oversim » pour la construction de l'overlay et sous « Inet » pour l'underlay a subi une évaluation de performances. Les résultats obtenus sont comparés avec ceux de BasicChord mobile que nous avons implémenté en l'absence de la structure de mobilité sur les simulateurs de P2P existants, et avec ceux de PredChord connu performant dans l'optimisation de la recherche de Chord.

L'approche proposée présente de bons résultats en termes de réduction de chemin de recherche et d'augmentation de taux de succès de recherche en comparaison avec BasicChord et en concurrence avec PredChord. RepMChord génère un nombre réduit de messages dans le processus de maintenance différemment de PredChord : avantages indéniables qui lui permettent de réduire la consommation de l'énergie.

Des résultats obtenus, on peut conclure que la technique proposée et son processus de recherche arrivent à optimiser la disponibilité de données et offert plus d'efficacité au processus de recherche, tout en évitant les inconvénients liés à l'overhead excessif généré. C'est bien cela, qui optimise la consommation de l'énergie tout en obtenant un overlay Chord plus adaptable à l'underlay MANET : objectifs fixé lors du début de ce travail.

Perspectives :

- Appliquer notre stratégie sur d'autres protocoles à base de DHT comme Pastry afin de prouver les performances qu'elle apporte et son adaptation à d'autres protocoles qui utilisent l'ID-Assignement (l'assignement à base d'identifiants) pour placer les ressources dans l'overlay.
- Combiner une approche de sensibilisation de localité avec notre approche de réplication pour tirer profit de la réduction du chemin physique, en plaçant les nœuds dans l'overlay en fonction de leur localisation physique, dans le but de minimiser la latence de recherche.

Communications et Publications

1. Sarra Cherbal, Abdellah Boukerram, Abdelhak Boubetra. "An Improvement of Chord Routing Protocol for Mobile P2P Networks", *International Conference on Advanced Networking, Distributed Systems and Applications, INDS'2014,* Bejaia, Algeria, June 17-19, pp. 1-4, 2014.

2. Sarra Cherbal, Abdellah Boukerram, Abdelhak Boubetra. "A Survey of Locality-Awareness Solutions in Mobile DHT Systems", *the 12th International Symposium on Programming and Systems, ISPS'2015,* Algiers, Algeria, April 28-30, pp. 1-7, 2015. (**Published in IEEE**).

3. Sarra Cherbal, Abdellah Boukerram, Abdelhak Boubetra. "A Survey of DHT Solutions in Fixed and Mobile Networks", *the International Journal of Communication Networks and Distributed Systems,* Vol. 17, No. 1, 2016. **Indexed in Scopus (Elsevier).**

4. Sarra Cherbal, Abdellah Boukerram, Abdelhak Boubetra. "RepMChord: A Novel Replication Approach for Mobile Chord with Reduced Traffic Overhead", *International Journal of Communication Systems*, doi: 10.1002/dac.3285, 2017. **Indexed in ISI Thomson, IF: 1.099.**

5. Sarra Cherbal, Abdellah Boukerram, Abdelhak Boubetra. "Locality-Awareness and Replication for an Adaptive Chord to MANET". *International Journal of Distributed Systems and Technologies*, Vol. 8, No. 3, 2017. **Indexed in Scopus, Web of Science Emerging Sources Citation Index (ESCI).**

Bibliographie

[Abi15] **Abid, S.A., Othman, N.A., Shah, N.** A Survey on DHT-Based Routing for Large-Scale Mobile Ad Hoc Networks. *Journal of ACM Computing Surveys,* Vol. 47, No. 2, 2015.

[AlM15] **Al Mojamed, M., Kolberg, M.** Design and evaluation of a peer-to-peer MANET crosslayer approach: OneHopOverlay4MANET. *International journal of Peer-to-peer Networking and applications*, pp.1-18, doi : 10.1007/s12083-015-0413-4, 2015.

[AlM16] **Al Mojamed, M., Kolberg, M.** Structured Peer-to-Peer overlay deployment on MANET: A survey. *Computer Networks Journal*, Vol 96, pp. 29-47, 2016.

[Ama10] **Amadou, I., Valois, F.** Pizza Forwarding : A Beacon-less routing protocol designed for realistic radio assumptions. *Proceeding of IEEE 4th International Conference on Sensor Technologies and Applications*, pp. 495-500, March 2010.

[Ama11] **Amadou, I., Chelius, G., Valois, F.** Energy-Efficient Beacon-less Protocol for WSN. *In 22nd IEEE Symposium on Personal, Indoor, Mobile and Radio Communications (PIMRC 2011)*, Toronto, Canada, September 2011.

[Amo12] **Amokrane, A., Langar, R., Boutaba, R., Pujolle, G.** A Green Framework for Energy Efficient Management in TDMA-based Wireless Mesh Networks. *In IEEE/ACM CNSM*, pp. 1–7, Las Vegas, USA, October 2012.

[And04] **Androutsellis-Theotokis, S., Spinellis, D**. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, Vol. 36, No. 4, pp. 335–371, 2004.

[Ban02] **Banerjee, S., Misra, A.** Minimum energy paths for reliable communication in multi-hop wireless networks. *In Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, MobiHoc '02*, pp. 146–156, Lausanne, Switzerland, 2002.

[Bau09] **Baumgart, I., Heep, B., Krause, S.** OverSim: A Scalable and Flexible Overlay Framework for Simulation and Real Network Applications. *IEEE*

Ninth International Conference on Peer-to-Peer Computing, Washington, USA, pp. 87–88, 2009.

[Bel07] **Belghith, A., Akkari, W., Bonnin, J. M.** Traffic aware power saving protocol in multi-hop mobile ad-hoc networks. *Journal of Networks*, Vol. 2, No. 4, pp. 1–13, August 2007.

[Bem12] **Bemmoussat, C., Didi, F., Feham, M.** Efficient routing protocol to support qos in wireless mesh network. *International Journal of Wireless and Mobile Networks (IJWMN)*, Vol. 4, No. 5, October 2012.

[Ben13] **Ben Guirat, F., Filali, I.** An efficient data replication approach for structured peer-to-peer systems. *IEEE Telecommunications (ICT)*, pp. 1–5, May 2013.

[Bit04] **Bittorent.** Available on : http://www.bittorrent.com/lang/fr/, First version on 2004.

[Bou12] **Bousia, A., Angelos, A., Luis, A., Verikoukis, C. V.** "green" distance-aware base station sleeping algorithm in lte-advanced. *In IEEE International Conference on Communications (ICC)*, pp. 1347–1351, June 2012.

[Bur04] **Burkhart, M., Rickenbach, P., Wattenhofer, R., Zollinger, A.** Does topology control reduce interference. *In Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '04*, pp. 9–19, Tokyo, Japan, 2004.

[Cap12] **Capone, A., Malandra, F., Sansò, B.** Energy savings in wireless mesh networks in a time-variable context. *Mobile Networks and Applications*, Vol. 17, No. 2, pp. 298–311, April 2012.

[Cha05] **Changsu, S., Young-Bae, K., Jai-Hoon, K.** Enhanced power saving for ieee 802.11 wlan with dynamic slot allocation. *In Proceedings of the First international conference on Mobile Ad-hoc and Sensor Networks (MSN'05),* pp. 498–507, Wuhan, China, 2005.

[Cha08] **Chan, H., Van, K., Hoang, G.** Characterizing Chord, Kelips and Tapestry algorithms in P2P streaming applications over wireless network. *In Proceeding of IEEE, International Conference on Communications and Electronics*, pp. 126-131, 2008.

[Cha12] **Chang, J.M., Lin, Y.H., Woungang, I., Chao, H.C.** MR-Chord: A Scheme for Enhancing Chord Lookup Accuracy and Performance in Mobile P2P

Networks. *Proceeding of the IEEE Intl. Conference on Communications (ICC 2012)*, Ottawa, Canada, June 2012.

[Che15] **Cherbal, S., Boukerram, A., Boubetra, A.** A Survey of Locality-Awareness Solutions in Mobile DHT Systems". *IEEE 12th International Symposium on Programming and Systems (ISPS'2015)*, Algiers, Algeria, pp. 1-7, April 2015.

[Che16] **Cherbal, S., Boukerram, A., Boubetra, A.** A Survey of DHT Solutions in Fixed and Mobile Networks. *The International Journal of Communication Networks and Distributed Systems,* Vol. 17, No. 1, 2016.

[Che17] **Cherbal, S., Boukerram, A., Boubetra, A.** RepMChord: A Novel Replication Approach for Mobile Chord with Reduced Traffic Overhead. *International Journal of Communication Systems*, doi: 10.1002/dac.3285, 2017.

[Chi97] **Chiang, C. C., Wu, H. K., Liu, W., and Gerla, M.** Routing in clustered multihop, mobile wireless networks with fading channel. *IEEE Singapore International Conference on Networks,* pp. 197-211, 1997.

[Cho12] **Chowdhury, F., Kolberg, M.** A Survey of Peer-to-Peer Solutions in Mobile and Cellular Networks. *In Proceeding of PGNet*, 2012.

[daH09] **da Hora, D. N., Macedo, D.F., Oliveira, L.B., Siqueira, I.G., Loureiro, A.A., Nogueira, J.M., Pujolle, G.** Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks. *Computer Communications Journal* Vol. 32, No. 13-14, pp. 1445–1459, 2009.

[Dav96] **David.B Jhonson, David.A Maltz.** Dynamic Source Routing in Ad Hoc wireless. In Mobile Computing Book, Imielinski, T. and Korth, H., Eds. Kluwer, Publisher: Springer US, Vol. 353, pp. 153-181, 1996.

[DeB46] **De Bruijn, N. G.** Combinatorial problem. *Proceedings of the Section of Sciences of the KoninKlijke Nederlands Akademie van Wetenschappen,* Amesterdam, Vol. 29, No. 7, pp. 758-764, 1946.

[Dob09] **Dobre, C., Pop, F., Cristea, V.** DistHash: A robust P2P DHT-based system for replicated objects. *Proceedings of 17th International Conference on Control Systems and Computer Science (CSCS 17)*, Bucharest, Romania, Vol. 1, pp. 453-460, May 2009.

[eMu02] **eMule.** Available on : http://www.emule.com/fr/, First version on 2002.

[Fan09] **Fantar, S., Youssef, H.** Locality-aware Chord over Mobile Ad-hoc Networks. *Proceedings of the IEEE Global Information Infrastructure Symposium (GIIS 2009),* Hammamet, Tunisia, June 2009.

[Fra06] **Fraigniaud, P., Gauron, P.** D2B: a de Bruijn Based Content- Addressable Network. *Theoretical Computer Science,* Vol. 355, No. 1, pp. 65–79, 2006.

[Gai04] **Gai A. T., Viennot L.** Broose: A Practical Distributed Hashtable Based on the De-Brujin Topology. [Research Rapport] RR-5238. France : Institut National de Recherche en Informatique et en Automatique (INRIA), pp. 16, 2004.

[Gho07] **Ghodsi, A., Alima, L.O., Haridi, S.** Symmetric Replication for Structured Peer-to-Peer Systems'. *In Proceedings of the international conference on Databases, information systems, and peer-to-peer computting*, 2007.

[Got12] **Gottron, C., Konig, A., Steinmetz, R.** A Cluster-based Locality-Aware Mobile Peer to Peer Architecture. *IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 643-648, March 2012.

[Gup00] **Gupta, P., Kumar, P. R.** The capacity of wireless networks. *IEEE Transactions on Information Theory*, Vol. 46, No. 2, pp. 388–404, March 2000.

[Hec04] **Heckmann, O., Bock, A., Mauthe, A., Steinmetz, R.** The eDonkey File Sharing Network. *Workshop on Algorithms and Protocols for Efficient Peer-to-Peer Applications, Informatik 2004*, September 2004.

[Ine12] **Inetmanet-2.0-inetmanet-2.2**. Available en ligne on: https://github.com/inetmanet/inetmanet, 2012.

[Jac01] **Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., Viennot, L.** Optimized link state routing protocol (OLSR). *In proceeding of IEEE International Multi Topic Conference (IEEE INMIC 2001)*, pp. 62-68, 2001.

[Jia05] **Jiang, J., Pan, R., Liang, C., Wang, W.** BiChord: an improved approach for lookup routing in Chord. *In Proc. of the 9th East European conference on Advances in Databases and Information Systems (LNCS 3631)*, pp. 338-348, 2005.

[Kaa03] **Kaashoek M. F., Karger D. R.** Koorde: A simple degreeoptimal distributed hash table. In: Kaashoek M.F., Stoica, I. (Eds) *International Workshop on Peer-to-Peer Systems (IPTPS)*, Vol. 2735 in LNCS, Springer, pp. 98–107, 2003.

[KaZ01] **KaZaA.** Available on : https://kazaa-lite-k.fr.softonic.com/, First version on 2001.

[Kel08] **Kelényi, I., Nurminen, J. K.** Energy Aspects of Peer Cooperation – Measurements with a Mobile DHT System. *IEEE Ineternational Conference on Communications Workshops (ICC'2008)*, pp. 164-168, 2008.

[Kha14] **Khan, M.A., Yeh, L., Zeitouni, K., Borcea, C.** MobiStore: Achieving Availability and Load Balance in a Mobile P2P Data Store. *IEEE 6th International Conference on Mobile Computing, Applications and Services (MobiCASE)*, pp. 171-172, 2014.

[Khi12] **Khirallah, C., Thompson, J. S.** Energy efficiency of heterogeneous networks in lte-advanced. *Journal of Signal Processing Systems*, Vol. 69, No. 1, pp. 105–113, October 2012.

[Kta09] **Ktari, S., Hecker, A., Labiod, H.** Exploiting Power-Law Node Degree Distribution in Chord Overlays. *In IEEE Next Generation Internat Networks*, pp. 1-5, 2009.

[Kum04] **Kumar, A., Merugu, S., Xu, J., Zegura E.W., Yu, X.** Ulysses: A Robust, Low- Diameter, Low-Latency Peer-to-Peer Network. *Journal of Transactions on Emerging Telecommunications Technologies*. Vol. 15, No. 6, pp. 571-587, 2004.

[LeD14] **Le-Dang, Q., McManis, J., Muntean, G.** Location-aware chord-based overlay for wireless mesh networks. *IEEE Transactions on Vehicular Technology*. Vol. 63, No. 3, pp. 1378–1387, 2014.

[Lee13] **Lee, S.B., Wong, S.H.Y., Lee, K.W., Lu, S.** Content management in a mobile ad hoc network: beyond opportunistic strategy. *International journal of communications networks and distributed systems.* Vol. 10, No. 2, pp. 123–145, 2013.

[Lop11] **Lopez-Peres, D., Ladanyi, A., Jüttner, A., Rivano, H., Zhang, J.** Optimization Method for the Joint Allocation of Modulation Schemes, Coding Rates, Resource Blocks and Power in Self-Organizing LTE Networks. *In Proceddings IEEE INFOCOM*, pp. 111-115, 2011.

[Mal02] **Malkhi, D., Naory, M., Ratajczak, D.** Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In ACM: *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, California, pp. 183-198 2002.

[Mur96] **Murthy, S., Garcia-Luna-Aceves, J. J.** An efficient routing protocol for wireless networks. Journal of *Mobile Networks and Applications*, Vol. 1, No. 2, pp. 183–197, 1996.

[Nap99] **NAPSTER** (1999) [online] http://www.napster.com.

[Omn14] **Omnetpp 4.6.** Available on: https://omnetpp.org/9-articles/software/3724-omnet-4-6-released, 2014.

[Pat09] **Patel, S., Elleithy, K., Rizvi, S. S.** Hierarchically Segmented Routing (HSR) Protocol for MANET. In: Proceeding of the *6th International Conference on Information Technology: New Generations ITNG 2009*, Nevada, USA, 2009.

[Per94] **Perkin, C. E., Bhagwat, P.** Highly Dynamic Destination-sequenced distance vector routing (DSDV) for mobile computers. *In: ACM SIGCOMM'94*, pp. 234-244, 1994.

[PER99] **Perkins, C. E., Royer, E. M.** Ad hoc on demand distance vector (AODV) algorithm. *In IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99),* pp. 90-100, 1999.

[Pla06] **Plancoulaine, S., Bachir, A., Barthel, D.** WSN node energy dissipation. Technical report, Technical report, France Telecom R&D, internal report, July 2006.

[Pla99] **Plaxton, C. G., Rajaraman, R., Richa, A.W.** Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*. Vol. 32, No. 3, pp. 241-280, 1999.

[Rat01] **Ratnasamy S. P., Francis P., Handley M., Karp R., Shenker S.** A scalable content-addressable network. *In: Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, US, pp. 161–172, 2001.

[Rhe04] **Rhea, R., Geels, D., Roscoe, T., Kubiaatowicz, J.** Handling Churn in a DHT. *In: ACM Proceedings of Usenix Annual Technical Conference*, USA, June 2004.

[Rip01] **Ripeanu, M.** Peer-to-peer architecture case study: Gnutella network. *IEEE 1st International Conference on Peer-to-Peer Computing (P2P)*, Suède, pp.99–100, 2001.

[Row01a] **Rowstron, A., Druschel, P.** Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: Guerraoui R. (eds)

Middleware'2001, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Vol. 2218, 2001.

[Row01b] **Rowstron, A., Druschel, P.** Storage management and caching in PAST, a large scale persistent peer-to-peer storage utility. *In proceeding of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP '01).* New York, NY, pp. 188-201, 2001.

[Sha12] **Shafaat, T. M., Ahmad, B., Haridi, S.** Id-replication for structured peer-to-peer systems. *Proceedings of the 18th European conference on Parallel Processing (Euro-Par'12)*, pp. 364–376, 2012.

[She02] **Sheetalkumar, D., Timothy, X. B.** Minimum energy routing schemes for a wireless ad hoc network. *In IEEE INFOCOM*, New York, NY, USA, pp. 1-11, June 2002.

[Sin12] **Singh. G., Singh. J.** MANet: issues and behaviors analysis of routing protocols. *Int. J. of Advanced Research in Computer Science and Software engineering,* Vol. 2, No. 4, April 2012.

[Sol12] **Soltani, N., Khaneghah, E. M., Sharifi, M., Leili, M.** A Dynamic Popularity-Aware Load Balancing Algorithm for Structured P2P Systems. In: Park J.J. et al (eds) *Network and Parallel Computing, NPC'2012, LNCS,* Vol. 7513, Springer, Berlin, Heidelberg, pp. 77–84, 2012.

[Sto01] **Stoica I., Morris R., Liben-Nowell D., Karger D. R., Kaashoek M. F., Dabek F., Balakrishnan H.** Chord: A Scalable Peer-topeer Lookup Protocol for Internet Applications. *In: Proceedings of SIGCOMM. ACM Press*, pp. 149–160, 2001.

[Tha11] **Thaalbi, M., Tabbane**, **N., Bejaoui, T., Meddahi, A.** An Enhanced chord-based P2P lookup protocol for Mobile Ad hoc Networks. *In IEEE IFIP Wireless Days (WD)*, pp. 1-5, October 2011.

[Tha12] **Thaalbi, M., Tabbane**, **N., Bejaoui, T., Meddahi, A.** Enhanced Backtracking Chord Protocol for Mobile Ad hoc Networks. *In IEEE ICCIT'2012,* Tunisia, pp. 191-195, 2012.

[Toh01] **Toh, C. K.** Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *Journal of IEEE Communications Magazine*, Vol. 39, No. 6, pp. 138–147, June 2001.

[Tri16] **Trifa, Z., Khamakhem, M.** A novel replication technique to attenuate churn effects. *International journal of Peer-to-peer Networking and applications*, Springer, Vol. 9, No. 2, pp. 344-355, 2016.

[Van03] **Van Dam, T., Langendoen, K.** An adaptive energy-efficient mac protocol for wireless sensor networks. *In Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03*, Los Angeles, California, USA, pp. 171–180, November 2003.

[Var08] **Varga, A., Hornig, R.** An Overview of the OMNet++ Simulation environnement. *In:Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and System,* Article No. 60, France, 2008.

[Wan12] **Wang, Y., Li, X.** AB-Chord: an efficient approach for resource location in structured P2P networks. *9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, pp. 278-284, 2012.

[Wan13] **Wang, Y., Wu, Q.** Research on the Chord Algorithm Based on Physical Location and Node Ability. *Fifth International Conference on Intelligent Human-Machine Systems and Cybernetics*, Vol.1, pp. 340-343, 2013.

[Wou15] **Woungang, I., Tseng, F. H., Lin, Y. H., Chou, L. D., Chao, H. C., Obaidat, M. S.** MR-Chord: Improved Chord Lookup Performance in Structured Mobile P2P Networks. *IEEE Systems Journal*, Vol. 9, No. 3, pp. 743-751, 2015.

[XuH01] **Xu, Y., Heidemann, J., Estrin, D.** Geography-informed energy conservation for ad hoc routing. *In Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom '01),* Rome, Italy, pp. 70–84, July 2001.

[Zha01] **Zhao B. Y., Kubiatowicz J. D., Joseph A. D.** Tapestry: An infrastructure for fault-tolerant wide-area location and routing. *In Technical Report* No. UCB/CSD-01-1141, Computer Science Division (EECS), University of California at Berkeley, California, USA, 2001.

[Zou09] **Zou, D., Gan, Y.** The research of a structural p2p model based on mobile cellular networks. *In IEEE International Conference on Networking and Digital Society (ICNDS),* Vol. 2, pp. 64–67, 2009.

[Zul12] **Zulhasnine, M., Huang, C., Srinivasan, A.** Towards an Effective Integration of Cellular Users to the Structured Peer-to-Peer Network. *Journal of Peer-to-Peer Networking and Applications*, Springer, vol. 5, No. 2, pp. 178-192, 2012.