République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A/Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Mémoire de fin de cycle
en vue de l'obtention du diplôme de master recherche en informatique
Option : **R**éseaux et **Sy**stèmes **D**istribués

Thème

Problème de recherche des services dans les réseaux P2P

Cas : graphes de Cayley

Présenté par :
✓ M^r **BEGREDJ** Farid.
✓ M^{elle} **MADAOUI** Lamia.

Proposé et encadré par :
✓**Dr.AMAD** Mourad.

Devant le jury :

✓**Président : Mr.KABYL** Kamel.
✓**Examinateur** 1 **: Mr. KHENOUS** Lachemi.
✓**Examinateur** 2 **: Mlle.MORDJI** Zouweyna.

* * *Promotion 2012/2013* * *

Table des matières

List of algorithmes

Table des figures

Liste des tableaux

Introduction générale

Au cours de ces dernières années, la technologie n'a pas cessé d'évoluer et de progresser en particulier la technologie informatique. Dans le domaine des réseaux informatique, de nouveaux outils sont apparus et permettent l'échange d'informations entre les différents utilisateurs *(ordinateurs)* à travers le monde entier tel que **Napster, Kazaa**...etc, Ce qui a mené a l'apparition d'un nouveau concept qui est le " peer to peer " (*abrégé " p2p "*), que l'on peut traduire en français par " *poste à poste* " ou " *égal à égal* " ou encore " *pair à pair* ".

Les systèmes pair à pair existaient bien avant les applications dites P2P d'aujourd'hui. Aux Balbutiements de l'Internet, tous les ordinateurs étaient des pairs, parce qu'ils n'y avaient pas un système de noms de domaines **DNS** *(**D**omain **N**ame **S**ystem)* En place, et une machine devait connaitre une autre à l'avance pour pouvoir établir une communication avec elle. C'est seulement aujourd'hui que l'on peut commencer à construire des systèmes constitués par de millions d'hôtes interconnectés. Les systèmes P2P sont la conséquence du développement de l'Internet.

Ces derniers s'appuient sur des concepts des systèmes distribués qui reposent sur un modèle décentralisé. Contrairement au modèle Client Serveur où chaque élément ne peut être que client ou serveur d'un service. En effet, le système P2P représente un réseau dont les noeuds *(peers)* sont équivalents en fonctionnalités c'est-à-dire ils peuvent être clients et serveurs à la fois. Un noeud peut aussi télécharger des ressources à partir d'un autre noeud, tout en fournissant des ressources a un autre noeud. Mais cette décentralisation a conduit inévitablement à l'apparition de nombreux problèmes comme le problème de recherche des services, car l'absence du serveur centrale ne permet pas d'avoir une vue globale du réseau.

La recherche et la localisation des services *(ressources)* dans ce type de réseau, représente un problème majeur, car c'est difficile de situer les noeuds. Donc on doit disposer de structure ou bien de topologie au niveau overlay *(présence des liens entre les noeuds)*. Pour améliorer et trouver des solutions pour la recherche et la localisation, des ressources des systèmes décentralisés structurés sont apparus.
Les graphes de Cayley représentent l'un des plus récents exemples de cette classe de systèmes et constitue la structure du modèle des réseaux étudiés dans ce mémoire.
Notre mémoire est organisé comme suit :
Le chapitre I : représente une étude générale sur les réseaux pair à pair à savoir leur

historique, caractéristiques, objectifs, avantages, limites...etc.

Le chapitre II : représente l'étude et la description détailler des graphes de Cayley à savoir leurs caractéristiques, leurs propriétés, leur technique de routage et enfin, la comparaison avec d'autres topologies.

Le chapitre III : consiste a présenter une nouvelle méthode de recherche et de localisation des ressources basée sur les graphes de Cayley, et proposer un nouvel algorithme de routage dans le but de trouver le plus court chemin d'une source donnée vers une destination quelconque dans le réseau.

Le chapitre IV : consiste la réalisation d'une application qui simulera la nouvelle méthode proposée dans le chapitre III.

Et enfin, viendra la conclusion générale qui va clore le mémoire en résumant les points essentiels abordés au cours de ce travail et puis, les perspectives et les améliorations pour les travaux à venir.

Chapitre 1

Généraltés sur les réseaux P2P

1.1 Introduction

Dans le domaine des réseaux, les applications suivent principalement deux modèles, à Savoir : *le client/serveur* et le modèle distribué. Ce dernier permet de remédier aux lacunes existantes dans le premier modèle, offrant une plus grande disponibilité et autonomie des sources d'information **[1].**

L'expression *" réseau peer to peer " (P2P)*, que l'on traduit généralement par réseau de *" poste-à-poste "*, *" pair à pair "* ou encore *" égal-à-égal "* désigne une architecture de réseau où les postes connectés communiquent dirèctement entre eux et partagent leurs ressources. Tous les postes ont un rôle équivalent, à la fois client et serveur **[2].**

Les réseaux P2P apparaissent actuellement comme un moyen populaire pour communiquer et partager de l'information - *des fichiers, plus souvent* -, mais également des calculs, des données plus structurées. Les technologies P2P se sont montrées très efficaces dans une certaine philosophie de partage. Elles permettent à différents participants *(individus, organisations)* de maintenir leurs propres ressources en les échangeant avec les autres participants dans un environnement distribué. Ce chapitre est consacré à l'étude des réseaux P2P.

1.2 Concepts de base des systèmes peer-to-peer

1.2.1 Historique et définitions

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

La définition la plus stricte est celle donnée au modèle dit **pur** P2P :*" c'est un système totalement distribué dans lequel tous les nœuds sont équivalents en termes de fonctionnalités et de tâches à résoudre "*. Cette définition ne concerne pas d'autres modèles P2P comme **KAZAA** qui est largement connu comme un système P2P dit **hybride**.

Une deuxième définition plus large a été donnée par **Shirky**[1] en *2000 :" le P2P est une classe d'applications qui exploite des ressources de stockage, de traitement, à travers le réseau Internet avec une présence humaine "* **[3].**

Cette définition est plus générale, car elle englobe les systèmes P2P qui se basent sur des serveurs centralisés comme *SETI@home*[2] et **Napster** et une définition plus significative est :*" Les systèmes pair à pair sont composés d'un ensemble d'entités partageant un ensemble de ressources, et jouant à la fois le rôle de serveur et de client. Chaque nœud peut ainsi télécharger des ressources à partir d'un autre nœud, tout en fournissant des ressources à un troisième nœud "* **[3].**

1.2.2 Caractéristiques des systèmes P2P

un vrai système peer-to-peer se reconnaît par les caractéristiques suivantes :

✓ **Décentralisation** : le contrôle n'est pas centralisé et il n'y a pas une vue globale de tous les pairs dans le réseau. La plupart des systèmes P2P ne sont pas purement

1. C.Shirky"Clay Shirky on P2P", Novembre 2000,
http ://www.scripting.com/Davenet/2000/11/15/clayShirkyOnP2p.html

2. Désigne le projet mondial qui veut étudier les résultats du télescope Arecibo pour y déceler des traces de vie extra-terrestre.

distribués, mais ils sont hybrides (*centralisé/ décentralisé*) **[4]**.

✓ **Dynamisme** : les systèmes supportent le dynamisme, c'est-à-dire les pairs peuvent joindre ou quitter le système de manière continue, sans qu'ils affectent le fonctionnement de ce dernier **[5].**

✓ **Connectivité Ad Hoc** : le modèle P2P se caractérise par une connectivité intermittente des pairs qui le composent, il vise à faire communiquer les différents nœuds dans les différents types de réseaux.

✓ **Tolérance aux fautes** : la tolérance aux fautes est la capacité des systèmes de continuer à fournir des services d'une manière régulière malgré la présence des fautes dans le software ou le hardware **[4].**

✓ **Sécurité** : la sécurité est la capacité des systèmes de gérer et protéger des informations sensibles en assurant les points suivants : -la confidentialité : une information doit etre consultée uniquement par les personnes autorisées. -l'intégrité : l'information doit etre transmise (*envoi et reception*)et/ou sauvegardée en totalité. - l'authentification : s'assurer de l'identité d'une entité afin de lui donner accés a des ressources ou a des données. -la non-répudiation : une entité ne peut pas nier qu'elle est l'origine d'une information. **[4].**

✓ **Anonymat** : l'anonymat est défini comme étant le degré pour lequel le système P2P tient compte des opérations non identifiées **[1].**

✓ **Scalabilité** : le système doit pouvoir gérer le nombre d'utilisateurs, puisque ces derniers augmentent d'une manière imprévisible dans les réseaux P2P **[1].**

✓ **Robustesse** : la robustesse doit être maintenue dans les trois composantes des systèmes P2P suivantes : sécurité, l'agrégation des ressources et la fiabilité **[6].**

En résumé, il existe plusieurs types de réseaux P2P avec plusieurs architectures, protocoles et mode de fonctionnement. Cependant, quelques caractéristiques sont communes à tous ces

systèmes, les plus importantes sont :

- ✓ Un système P2P doit être constitué d'au moins de deux pairs.
- ✓ Les nœuds d'un réseau P2P sont généralement de nature volatile *(les pairs peuvent joindre ou quitter le réseau en toute liberté).*
- ✓ un ou plusieurs nœuds centralisés peuvent jouer le rôle des serveurs dédiés dans un système P2P, selon la nature de l'application. Ces nœuds sont connus sous le nom de " Super-Pairs ". Les systèmes P2P qui ne contiennent pas de serveurs dédiés sont dits systèmes P2P purs.
- ✓ les pairs peuvent appartenir à différents propriétaires. Dans le cas de " clusters ", un pair peut être membre de plusieurs groupes en même temps.

1.2.3 Objectifs des systèmes P2P

Les systèmes P2P ont pour principal objectif de répondre aux exigences des utilisateurs. Les réseaux P2P sont apparus pour résoudre quelques problèmes rencontrés dans le paradigme client/serveur.
Les atouts pour lesquels les internautes préfèrent l'utilisation des systèmes P2P sont :

- ✓ Partage et réduction des coûts entre les différents peers ;
- ✓ Fiabilité et passage à l'échelle, l'absence d'élément centralisé pour l'échange des données permet d'accroître la fiabilité en supprimant tout point central de panne et d'améliorer le passage à l'échelle en évitant les goulots d'étranglement ;
- ✓ Agrégation des ressources et inter-opérabilité, en mettant en commun des ressources individuelles comme la puissance de calcul ou de l'espace de stockage,
- ✓ Accroissement de l'autonomie en l'absence d'une autorité centrale, il est de la responsabilité de chacun de partager ou non des fichiers,

- ✓ Anonymat pouvant être assuré par certaines applications, en utilisant par exemple des algorithmes de routage qui rendent quasiment impossible le pistage d'une requête,

- ✓ Communication ad-hoc et collaborative **[7].**

1.3 Mode client/serveur vs mode P2P

1.3.1 Mode Client/serveur (c/s)

Il décrit la relation entre deux programmes d'ordinateurs. Dans cette architecture le programme client fait une demande de service, à l'autre programme qui est celui du serveur. Les fonctions standards du réseau telles que l'échange de mails, l'accès aux web et aux bases de données sont basées sur le modèle client/serveur.

Par exemple, un navigateur web et un programme client qui se trouve sur le PC de l'utilisateur et qui permet l'accès â n'importe quel serveur web dans le monde. Le navigateur transmet la demande du client au serveur web qui, à son tour transmet la réponse au navigateur web qui affiche les informations **[8].**
La *figure*.1.1 montre un exemple de l'architecture c/s dans laquelle tous les ordinateurs sont reliés à un serveur central. Ces ordinateurs n'ont aucune connaissance du réseau et chacun d'eux peut communiquer uniquement avec le serveur central sans même savoir que les autres ordinateurs existent.

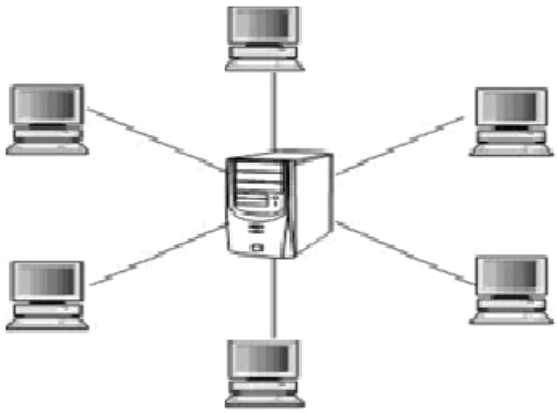

FIGURE 1.1 – Architecture Client/Serveur

1.3.2 Mode P2P

Il désigne un ensemble de techniques permettant de mettre en place un système d'échange de données avec plusieurs utilisateurs entre eux.
Les échanges passent dirèctement d'un ordinateur à l'autre. On peut établir une communication Peer to Peer en donnant à chaque utilisateur, dans un même protocole de communication, les capacités d'être client et serveur simultanément *(notion de servent)*, dirèctement ou à travers un réseau comme l'Internet. *Napster, Gnutella, FastTrack*[3] sont des exemples de ces types de réseaux **[9].**

- **Avantages**

On peut résumer les avantages d'un réseau peer to peer dans les points suivants :

- ✓ Réseaux très extensible,
- ✓ Responsabilité distribuées,
- ✓ Différent canaux de communication possibles,
- ✓ Utilisation de toute la bande passante,
- ✓ Haute disponibilité,
- ✓ Résistant aux pannes,
- ✓ Calculs distribués,
- ✓ Espace de stockage distribué **[10].**

3. Le réseau peer-to-peer (poste a poste) le plus utilise grâce aux clients KaZaa, et Grokster.

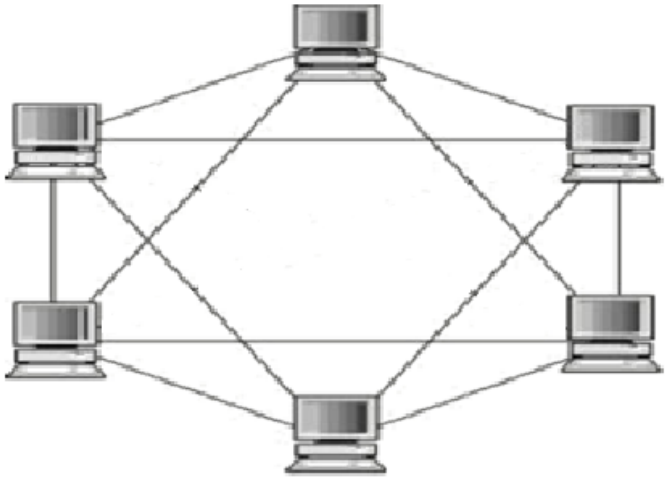

FIGURE 1.2 – Architecture peer to peer

- **Limites**

Le peer to peer présente néanmoins des limites les plus essentielles sont les suivantes :

- ✓ Réseaux redondant,
- ✓ Requête pour une information, ∗Différents résultats, ∗Sans réponse,
- ✓ Apparition/Disparition de ressource à tout moment,
- ✓ Attaques distribuées,
- ✓ Problème de sécurité,
- ✓ Authentification, Confidentialité et Intercepteurs **[11].**

1.3.3 Comparaison entre C/S et P2P

Traditionnellement, l'échange de services entre ordinateurs est fondé sur la technique*client/serveur*, selon cette architecture, il n'y a qu'une seule entité centrale très puissante, le serveur, et plusieurs entités généralement de puissances inférieures, les clients. Le serveur est le seul fournisseur des services aux clients. Un client consomme les services exécutés par le serveur, sans partager aucune de ses propres ressources.

L'architecture *pair-à-pair* se pose comme une solution de rechange à l'architecture *client/serveur* en offrant plusieurs avantages par rapport aux autres basés sur le paradigme client/serveur, la *table 1.1* montre bien les différences entre les deux modèles **[4].**

Critère	Modèle Client-serveur	Modèle P2P
Gestion	Supervisée	Auto-organisée
Présence	Permanente	Ad Hoc
Accès aux ressources	Recherche	Découverte
Organisation	Hiérarchique	Distribuée
Mobilité	Statique	Mobile
Disponibilité	Dépendante du serveur	Indépendante des pairs
Nommage	DNS	Indépendant
Modèle de programmation	RPC	Asynchrone

TABLE 1.1 – Comparaison entres les infrastructures Client/Serveur et P2P.

1.4 Architectures du P2P

1.4.1 Architecture centralisée

Dans cette architecture, un client *(un logiciel utilisé par les membres)* se connecte à un ou plusieurs serveurs qui gèrent les partages, la recherche, l'insertion d'informations, bien que celles-ci transitent dirèctement d'un utilisateur à l'autre.

Toutes ces informations sont collectées a l'aide d'un système d'indexation appelé la *" table de hachage distribuée "* qui permet en théorie d'éviter la multiplication de fichiers inutiles **[4].**

En effet, les fichiers peuvent être identiques sans pour autant posséder le même nom et inversement. On associe donc un identifiant unique *(" hash ")* à chaque pair, chaque mot clé et chaque fichier en fonction de son contenu et non de son nom. Ce qui permet par exemple de récupérer le bon fichier même si le nom est incorrect.

L'avantage de l'architecture centralisée est bien sûr d'avoir une vue globale et très complète du réseau, à condition que le nombre d'utilisateurs connectés soit limité pour ne pas se retrouver submergé par une liste interminable d'informations si on ne recherche rien

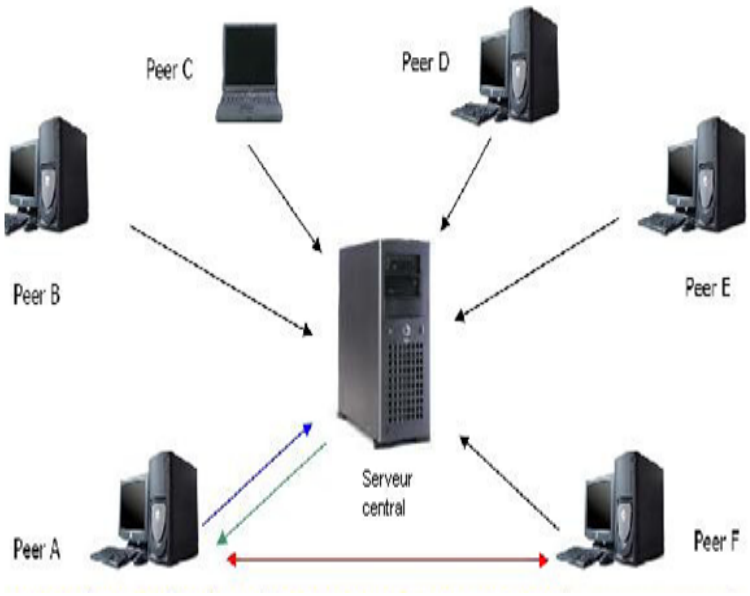

FIGURE 1.3 – Architecture centralisée

de particulier.

Mais dans le cas où notre recherche de fichiers est ciblée, le réseau centralisé se montre très rapide. Le serveur reçoit notre requête, nous indique quel utilisateur est susceptible d'héberger le fichier recherché avant de nous connecter dirèctement au PC concerné. À aucun moment le fichier ne passe par le serveur central.

Malheureusement, le réseau centralisé est très vulnérable. En cas d'arrêt du serveur, c'est le réseau tout entier qui disparaît **[12].**

-Avantages

Cette architecture présente quelques avantages, dus a la centralisation du serveur, la méthode de recherche ainsi qu'a la tolérance aux fautes, on cite :

✓ Avantages habituels d'un serveur central,

1. Facile à administrer et à contrôler,

✓ Evite les recherches coûteuses sur le réseau,

1. Efficacité de la recherche par indexation centralisée,
2. Pas de routage,

3. Planification de la gestion des utilisateurs,

✓ Tolérance aux fautes par un sondage régulier des peers connectés, état cohérent,

-Limites

La centralisation du serveur provoque certaines imperfections et défauts comme :

1. Pas d'anonymat partout,
   - ✓ L'utilisateur est connu du serveur,
   - ✓ ainsi que des peers sur lesquels il télécharge,
2. Limites habituelles d'un serveur central,
   (a) Réseau complètement dépendant du serveur,
   (b) Disponibilité
   (c) Problème du passage à l'échelle :
   - ✓ Saturation de la bande passante,
   - ✓ Saturation du nombreux processus,

   (d) Très facile de fermer le service, car localisé,
3. Mauvaise information du débit des peers pour ne pas être sollicités.

-Exemple du réseau Napster

Il a été crée en *1999* par un certain *Shawn Fanning*, dans le but de pouvoir échanger des fichiers *MP3* avec ses amis, suite a l'augmentation des débits, la baisse des prix des fournisseurs d'accès et des ordinateurs qui facilitent le partage de fichiers.

Napster permet le téléchargement de *MP3* sur Internet en s'appuyant sur la technologie Peer-to-Peer centralisée. Des la première semaine, *15 000* personnes ont téléchargé le logiciel, puis *23* millions en Juillet *2000* **[4].**

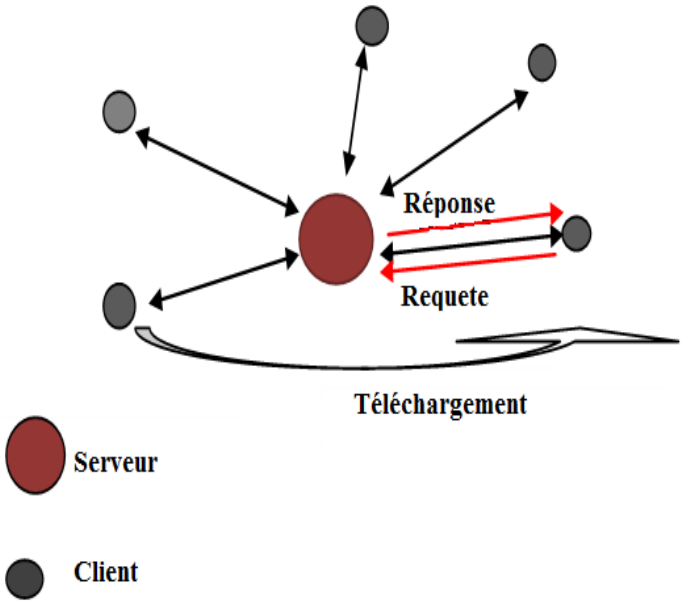

FIGURE 1.4 – Architecture du réseau Napster

La facilité d'utilisation de **Napster** lui assure un succès rapide. Des lors, le téléchargement illégal fait partie intégrante du phénomène Internet et les logiciels Peer-to-Peer deviennent les nouvelles applications à la mode. Toutefois la *" disponibilité temporelle "* est encore faible.

En effet, les forfaits Internet ne sont pas illimites et les utilisateurs ne laissent pas leurs ordinateurs connectés en permanence. Le comportement des utilisateurs est donc défini par des facteurs technologiques et financiers. Il y a un décalage entre la technique et l'utilisation.

L'apparition du commerce illégal de CD gravés est un facteur d'utilisation important de **Napster**. Déjà sous le coup d'attaques des majors de la musique et d'artistes, **Napster** est condamné à ne plus permettre l'échange de fichiers protégés par des droits à travers leurs serveurs, il a été fermé définitivement en *2002* pour raisons juridiques **[13].**

1.4.2 Architecture distribuée ou décentralisée

Étant donné qu'il n'ya plus de serveurs centraux, ce sont tous les nœuds qui assurent ce rôle, et c'est pour cette raison qu'il est appelé un réseau **pur peer to peer**.

Lorsqu'un client souhaite se connecter, il va donc être nécessaire d'envoyer un message *broadcast* afin de savoir quelles autres personnes du réseau sont actives. Seules ces personnes répondront au message *broadcast (qui est une methode de transmission de données a l'ensemble des machines du réseau)*. On est alors connecté au réseau.

Afin de garder des informations cohérentes, un utilisateur n'est pas connecté dirèctement à plus de 3 ou 4 nœuds, il connait tous les utilisateurs avec une profondeur d'arbre de 07 sauts généralement. Chaque nœud a une vision limitée du réseau **[10]**.

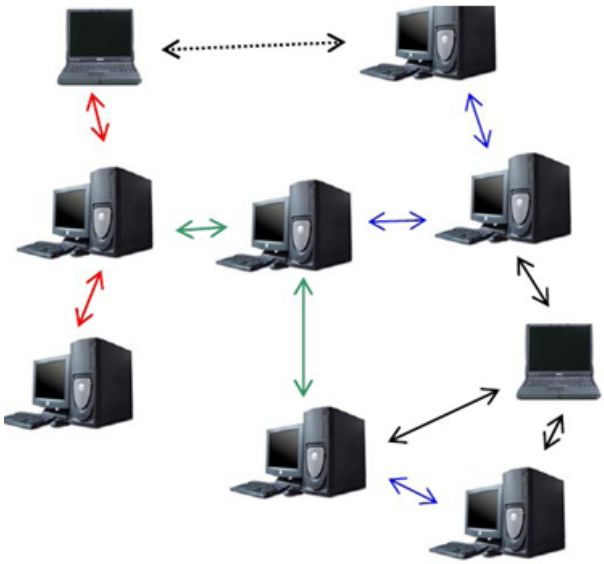

FIGURE 1.5 – Architecture distribuée ou décentralisée

-Avantages

1. Administration simple et mutualisée,
2. Topologie évolutive (taille illimitée en théorie),
3. Disponibilité du réseau, on ne peut l'arrêter,
4. S'adapte bien à la dynamique du réseau (*allées et venus des pairs*),

-Limites

1. Gros consommateur de bande passante à cause des broadcasts envoyés,

2. Pas de garantie de succès, ni d'estimation de la durée des requêtes *(utilisation de TTL*[4] *Time To Live)* **[7].**

-Exemple du réseau Gnutella

C'est un protocole de fichiers partagés. La première version *(version 0.4)* date de mars *2000*, il fut par Justin Frankel et Tom Pepper.

Les applications qui implémentent le protocole **Gnutella** autorisent les utilisateurs à rechercher et charger des fichiers sur tous les autres utilisateurs connectés à la communauté Gnutella, avec plusieurs dizaines de milliers d'utilisateurs simultanés.
Il existe plusieurs implémentations compatibles apportant des extensions : Limewire[5], Toad-Node[6], BearShare **[7].**

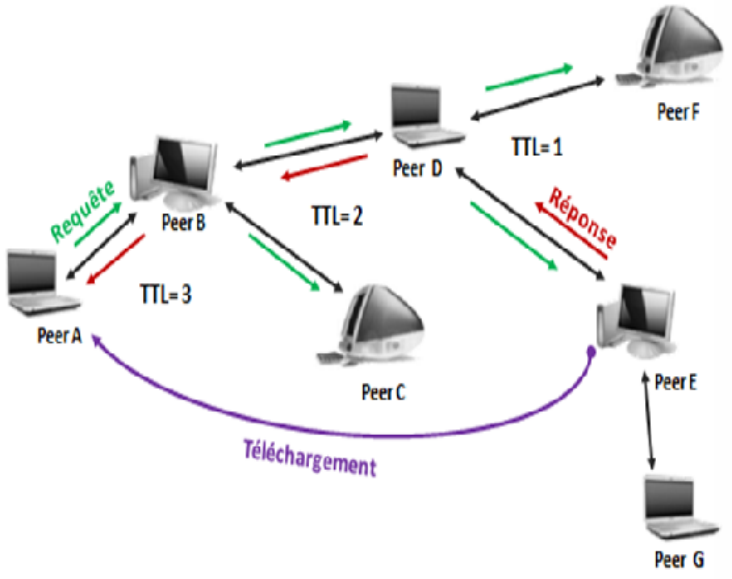

FIGURE 1.6 – Architecture du réseau Gnutella

Il utilise le mécanisme de TTL *(Time To Live)* pour borner le nombre de sauts des messages dont il est décrémenté par chaque voisin. Lors d'une recherche, le peer envoie la

4. Acronyme pour Time To Live *(période pendant laquelle va vivre le paquet dans un réseau)*http ://fr.wikipedia.org/wiki/Time-to-Live.

5. Un logiciel de peer to peer, ou tu télécharge, musique, films...etc. illégalement ou légalement.

6. Est une application de partage de fichiers peer-to-peer (P2P) qui facilite la recherche et le partage de fichiers, http ://www.slyck.com/story337-Toadnode-Not-so-Dead.

requête à ses voisins, qui font de même et ainsi de suite, jusqu'à atteindre les nœuds à distance 7 *(TTL = 7)* du demandeur, (7 est le nombre de sauts maximal d'une requête http).
La figure 1.6, illustre un exemple de **Gnutella** avec TTL = 3 **[1].**

1.4.3 Architecture P2P hybride

Le réseau hybride est plus complexe à mettre en oeuvre, il combine à la fois le réseau centralisé et décentralisé. En effet, sa structure permet de diminuer le nombre de connexions sur chaque serveur, et ainsi d'éviter les problèmes de bande passante.

D'autre part, le réseau de serveurs utilise un mécanisme issu des réseaux décentralisés pour tenir à jour un annuaire client et un index de fichiers à partir des informations provenant des autres serveurs. Un serveur peut donc proposer à n'importe quel client toutes les informations contenues sur le réseau **[10].**

Cette solution, rendant le système un peu moins robuste, elle est employée dans les systèmes *FastTrack*, comme **KaZaA**. Les nœuds du réseau peuvent alors devenir super-nœuds et vice-versa, selon les besoins du système ou de leur propre choix **[14].**

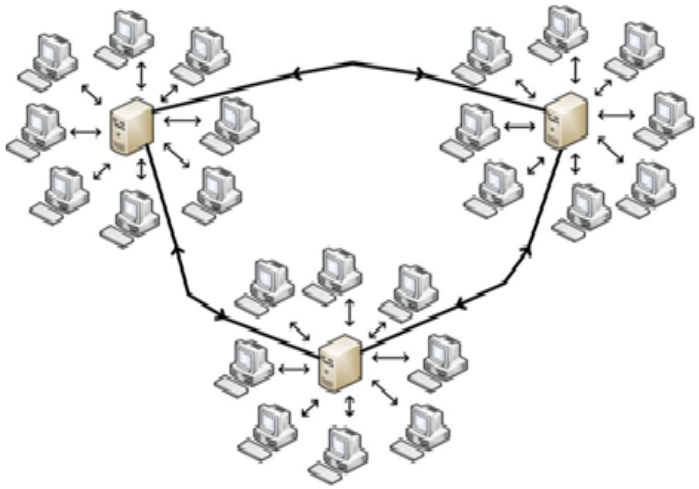

FIGURE 1.7 – Architecture hybride

-Avantages

- ✓ Présence d'un serveur central : facile à administrer, et donc facile à contrôler,
- ✓ Evite les recherches coûteuses sur le réseau : pas de routage et planification de la gestion des utilisateurs,
- ✓ Tolérance aux fautes en sondant régulièrement les pairs connectés et en maintenant un état cohérent,
- ✓ Service novateur qui généralise le P2P.

-Limites

- ✓ Pas d'anonymat partout, car chaque pair est connu du serveur et des pairs sur lesquels il télécharge,
- ✓ Limites habituelles d'un serveur central : problème de disponibilité, de passage à l'échelle *(saturation de la bande passante et du nombre de processus)*,
- ✓ Certains pairs peuvent mentir sur leur débit pour ne pas être sollicités.

-Exemple du réseau KAZaA

Il suit le même principe que **Napster**, mais avec absence de serveurs fixes. L'utilisation des pairs avec une connexion Internet rapide et processeur puissant : *Super Pairs*. Ces nœuds particuliers du réseau se comportent comme les serveurs **Napster** : connaissance des fichiers présents chez les autres utilisateurs, et adresses de ces fichiers. Chaque *Super Pair* réfère une recherche aux autres *Super Pairs*.

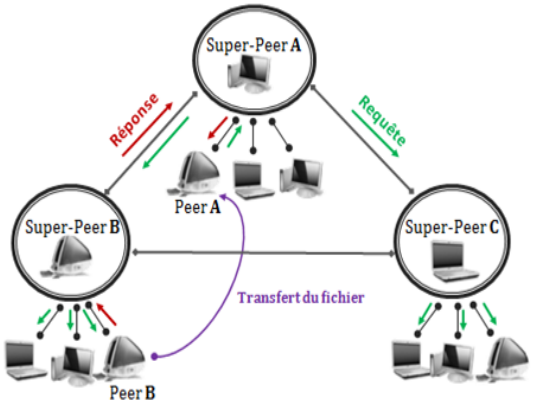

FIGURE 1.8 – Architecture du réseau Kazaa

La recherche est donc plus rapide, on ne cherche que dans les fichiers indexés par le *Super Pair* auquel on est connecté. Ce modèle *Super Pair* est une variante du modèle hybride.

1.5 Applications de la technologie P2P

Les réseaux " peer to peer " peuvent être classifiés par leur degré de centralisation comme nous l'avons vu précédemment. Dans ce qui suit, nous allons décrire les différentes applications du *" peer to peer "* .
Ainsi, nous pourrons observer que le **" peer to peer "** ne se réduit pas seulement au partage de fichiers.
Les différents types d'application que nous allons traiter sont :

- ✓ le partage de fichiers,
- ✓ Le calcul distribué *" Grid Computing "*,
- ✓ Système de sauvegarde distribué,
- ✓ Des programmes de messagerie,
- ✓ Streaming P2P,
- ✓ Plateformes de développement.

1.5.1 Partage de fichiers

L'application principale concerne l'échange de contenu, elle est fortement connue par Napster, Gnutella, Morpheus, Freenet (*anonymat des sources et intégrité des documents*), **KaZaa et BitTorent**, (*transfert parallèle de plusieurs sources, possibilité d'arrêt et de reprise d'un transfert*). D'autres projets consistent à établir un système de fichiers distribué dans une communauté, comme **PAST, OceanStore** (*plus de 6 millions d'utilisateurs*), **CAN ou Chord.**

Les bases de données distribuées et les tables de hachage distribuées commencent aussi à utiliser les protocoles P2P *(Mariposa)* **[13].**

1.5.2 Calcul distribué *"Grid Computing"*

Consiste à utiliser les machines connectées à l'internet pour faire des petites portions d'un grand calcul, en exploitant les ressources *(CPU, mémoire,etc)*inutilisées des PC du réseau en

vue d'accroitre le potentiel réseau, comme exemple le projet : *SETI@home* (*Search for Extra Terrestrial Intelligence*) **[4].**

1.5.3 Système de sauvegarde distribué

Le système de sauvegarde réparti s'appuie sur la coopération des pairs à mettre à disposition leurs espaces de disques inutilisés, un utilisateur peut sauvegarder d'une manière transparente et sécurisée une copie de ses données dans les autres pairs du réseau P2P afin de les récupérer en cas de perte ou de dégâts occasionnés aux données locales.
Nous pouvons citer des projets comme *Wuala*[7] , *DisPairSe*[8]... ,etc **[2].**

1.5.4 Programmes de messagerie

De nos jours, il existe des services de messagerie électronique basés sur le principe P2P, les utilisateurs peuvent envoyer et recevoir des e-mails d'une façon sécurisée, pas besoin d'un serveur central pour stocker temporairement les messages, ce qui assure la confidentialité des correspondants. Un système d'authentification et de cryptage est utilisé afin de protéger leurs contenus, un bon exemple est **Jeftel.com [9].**

Des logiciels de messagerie instantanée peuvent aussi être vus comme P2P, on citera des exemples comme *ICQ*[9] qui certes utilisent un serveur mais juste pour la résolution d'adresses *(les adresses utilisées peuvent être des alternatives aux adresses IP).*

1.5.5 Streaming P2P

Le streaming P2P est le fait de regarder en direct des flux produits et/ou relayés par d'autres pairs du réseau afin d'éviter ou du moins diminuer la congestion qui pourrait se produire sur les serveurs de téléchargement, tout se déroule entre les personnes qui veulent accéder au fichier, *Swarmplayer*[10], qui permet de lire des vidéos en streaming en utilisant Bittorrent, s'annonce comme une vraie révolution dans le domaine **[9].**

7. Est un stockage sécurisé pour tous vos fichiers.
8. Acronyme de Disque Pair à Pair Sécurisé.
9. Un système propriétaire de messagerie instantanée.
10. Application qui permet de visionner des vidéos en cours de téléchargement sans attendre le téléchargement complet grâce au protocole BitTorrent, http ://swarmplayer.p2p-next.org/dev.html.

1.5.6 Plateformes de développement

La plupart des logiciels P2P ont été développés de manière spécifique sans référence à des standards propres au P2P, dans le but d'uniformiser les réseaux P2P, des plates-formes sont implémentées pour servir de base au développement des applications P2P, Elles assurent les fonctionnalités de base telle que : gestion des pairs, attribution des identifiants, découverte des ressources, communication entre pairs et sécurité. On peut citer la plate-forme **JXTA** [11] développée par *Sun MicroSystems* [12]**[5].**

1.6 Conclusion

Aujourd'hui, le peer-to-peer est un phénomène de société. Quel internaute ne connait pas une application P2P ? Grâce au développement et l'accessibilité de l'Internet haut débit, mais aussi avec le vaste choix d'applications de partage de fichiers, proposées.

Dans ce chapitre, nous avons donné quelques définitions, aspects et caractéristiques des systèmes pair-à-pair. L'avenir du P2P est dans les architectures complètement décentralisées ou hybrides qui pourraient résoudre le problème de l'asymétrie de l'*ADSL* [13]. Même s'il rèste encore des recherches à faire sur le domaine, le " peer to peer " est l'avenir du partage d'informations (*Fichiers, audio, vidéo...etc.*). De nombreuses recherches sont menées pour fournir des services essentiels et améliorer la qualité et les performances des applications Peer-to-Peer.

De nouvelles études sont en cours pour améliorer l'efficacité des systèmes Peer-to-Peer en termes de routage et localisation de services, comme le cas des graphes de *Cayley* qu'on présentera dans le prochain chapitre.

11. Est un projet Open Source lancé par Sun Microsystems en avril 2001. Il vient du mot anglais " Juxtapose " est un spécification d'un ensemble de 6 protocoles, qui permettent le partage, la communication et la collaboration entre les peers, http ://fr.wikipedia.org/wiki/JXTAv.
12. Enterprise a l'origine du développement du langage Java en 1995.
13. Terme signifiant : " Asymetric Digital Subscriver Line " ou " ligne asymétrique numérique ", http ://fr.wikipedia.org/wiki/Asymmetric-Digital-Subscriber-Line.

Chapitre 2

Généralités sur les graphes de Cayley

2.1 Introduction

Les réseaux généraux tels que Internet sont devenus de plus en plus complèxes et les graphes de Cayley et les graphes de Coset sont d'excellents modèles pour des réseaux d'interconnexion étudies en liaison avec le traitement en simultanéité et le calcul distribué.

Les graphes de Cayley par exemple, ont été proposés pour l'élaboration d'un système pour soutenir les bases de données muti-systèmes ($DBMS$) dans le cloud (*nuage*), les tables de hachages distribuées ($DHTs$), pour une symétrie pour la topologie virtuelle de réseau, le calibrage du potentiel d'un système de gestion de données,etc.
Plusieurs réseaux d'interconnexion bien connus sont des graphes de Cayley ou de Coset. Par exemples :
-l'hypercube : dans un hypercube Q_n, chaque sommet porte une étiquette de longueur n sur un alphabet $A = \{0, 1\}$, et deux sommets sont adjacents si leurs étiquettes ne diffèrent que d'un symbole.
-le cube-connected cycles : dans la théorie des graphes, le ccc est un graphe non orienté cubique, formé en remplaçant chaque sommet d'un graphe hypercube par un cycle.
sont des graphiques de Cayley alors que :
-De Bruijn : est un graphe orienté qui permet de représenter les chevauchements de longueur $n-1$ entre tous les mots de longueur n sur un alphabet donné
est un graphe de *Coset.* Ce dernier est un graphique associé à un groupe G et à un groupe générateur tel que $\{x_i : i \in I\}$, et à un sous-groupe $H \leq G$.

Les graphes de Cayley ont été employés pour expliquer et unifier des réseaux d'interconnexion avec beaucoup d'avantages. Le présent chapitre sera consacré pour quelques définitions et propriétés des graphes de Cayley.

2.2 Préliminaires

rappelons d'abord quelques définitions de la théorie des graphes, de façon à fixer le vocabulaire et les notations.

2.2.1 Groupe

Un groupe est un ensemble G muni d'une opération $*$, c'est-a-dire d'une application G×G→G, qui a de plus les propriétés suivantes :

- ✓ Pour tous x, y, z dans G, on a $(x*y)*z = x*(y*z)$,
- ✓ Il y a un élément neutre e qui vérifie pour tout x de G, $xe = ex = x$,
- ✓ Pour tout x de G, il y a un inverse y tel que $x*y = y*x = e$ **[15].**

2.2.2 Graphe(*non orienté*)

C'est un graphe construit a partir de deux ensembles V et E, noté $G(V,E)$ où E est un ensemble de paires de points de V. V constitue l'ensemble des sommets de G et E celui de ses arêtes.

Deux paires x et y∈V sont voisins dans G, et on note (x∼y), ou $deg_G x$ si $x, y \in E$.
Le degré d'un sommet x, noté d(x), est le nombre de ses voisins dans G. x et y sont reliés via une arête dans G, ce qui est noté $x \leftrightarrow_G y$ s'il existe une suite $x_1, \ldots, x_n = y$ de sommets de G tel que, pour $i = 1, \ldots, n-1$,$x_i \sim x_{i-1}$.

Définition 1. Un graphe orienté $G = (V(G), E(G))$ est un ensemble de sommets VG et un ensemble des arêtes orientées EG (ou arcs) $(u, v) \in VG \times VG$ **[16].**

Lorsque tous les couples de sommets de G sont reliés, G est dit connexe. Les composantes connexes d'un graphe G sont les classes d'équivalence de sommets pour la relation $\leftrightarrow_G$, si $x \in V$, la composante connexe est notée $K_G(x)$.

Si x et y sont deux sommets de G reliés entre eux, la distance $d_G(x, y)$ entre x et y est la longueur n de la plus courte suite $x = x_1, \ldots, x_n = y$ avec $x_i \sim x_{i+1}$, pour $i = 1, \ldots, n-1$ **[16].**

Définition 2. Soit $G = (V, E)$ un graphe où $V = V_1$, V_2,... V_n est l'ensemble de sommets et $E = E_1$, E_2,....E_m est l'ensemble des arcs dans le graphe G. Deux sommets V_i et V_j dans le graphe G sont dit connectés, s'il existe une série consécutive d'arcs $E = E_1$, E_2,....E_t tel que : E_1 commence à partir du sommet V_i et E_t se termine au niveau du sommet V_j. La série des arcs commençant de V_i à V_j est appelée un chemin. La longueur P d'un chemin est le nombre des arcs constituant ce dernier **[17].**

Définition 3. Un graphe orienté G est fortement connexe si entre deux sommets quelconques x et y, il existe un chemin de x vers y et un chemin de y vers y. Le nombre de connectivité du graphe G (notée λ) est le nombre minimal de sommets dont l'enlèvement déconnecte le graphe. Puis, la tolérance aux pannes d'un graphe est $(\lambda - 1)$ **[17].**

Un sous-graphe de $G = (V, E)$, est un graphe (V', E) ou $V' \subset V$ et $E' = \{\{x, y\} \in E \mid x, y \in V'\}$. La donnée d'un sous-graphe équivaut donc à celle de ses sommets ou de ses arêtes **[16].**

Graphe transitif : soit le graphe $G = (V, E)$.

le graphe transitif est obtenu comme suite :$(x_i, x_j) \in E$, $(x_j, x_k) \in E \Rightarrow (x_i, x_k) \in E$

Un morphisme de graphes de $G = (V, E)$ dans $G' = (V', E')$ est une application $f : V \to V'$ telle que pour tous $x, y \in V$, $x \sim_G y \Rightarrow f(x) \sim_{G'} f(x)$.

Un isomorphisme de graphes est un morphisme de graphes bijectif dont l'application réciproque est également un morphisme de graphes, et on parle d'automorphisme de graphes lorsque les graphes de départ et d'arrivée sont les mêmes **[16].**

Remarque :un isomorphisme de graphes induit naturellement une bijection $(E \to E')$ que l'on notera de la même manière. Un graphe $G = (V, E)$ est transitif si le groupe de ses automorphismes agit transitivement sur ses sommets.

Un cycle dans un graphe G est une suite $x_1, \ldots, x_{n-1}x_n$ de sommets distincts de G, avec

$n \geq 3$, telle que, pour $i = 1, \ldots, n-1, x_i \sim x_{i+1}$ et $x_n \sim x_1$.

Un arbre est un graphe connexe et sans cycles. Un arbre couvrant d'un graphe G est un sous-graphe T de G qui est un arbre et qui contient tous les sommets de G. Une forêt couvrante d'un graphe G est un sous-graphe F de G sans cycle *(c'est donc une réunion disjointe d'arbres)* et qui contient tous les sommets de G **[16].**

2.3 Graphes de Cayley

2.3.1 Historique

Les graphes de Cayley ont été construits la première fois en 1878 par le mathématicien A. Cayley. La construction de ces graphiques est décrite par la théorie de groupe algébrique finie. Un groupe $(V, *)$ se compose d'un ensemble V qui est fermé en vertu de l'inversion et d'une loi de composition simple $(*)$, également connu sous le nom de multiplication de groupe.
Il existe également un élément d'identité $I \in V$. Un groupe est fini s'il y a un nombre fini d'éléments dans V **[18].**

2.3.2 Définition

Soit Γ un groupe de type fini et S une famille génératrice finie symétrique (*i.e.* $g^1 \in S$, *si et seulement si* $g^{-1} \in S$), ne contenant pas l'élément neutre e. Tout élément de Γ est donc représenté par au moins un mot en S. On associe à (Γ,S) un graphe $G = (V, E)$, appelé graphe de Cayley de Γ par rapport au système de générateurs S.

L'ensemble des sommets V s'identifie aux éléments de Γ et $\{g_1, g_2\} \in E$ si et seulement si, $g^1 g_2 \in S$. Ce graphe est connexe. La distance combinatoire de e à g dans ce graphe est le nombre minimal de lettres de S qu'il faut pour une écriture de g.

On peut évidemment étendre la définition à un espace X ou Γ agit. Le graphe associé $G = (V, E)$ à pour sommets les points de X et $\{x,y\}$ est une arête si et seulement si il existe $g \in S$, tel que $gx = y$. Ce graphe est de degré constant égal au nombre d'éléments de S **[19].**

-Exemple :

G est le groupe additif de $\mathbb{Z}/13\mathbb{Z}$, $S = \{1, 5, -1, -5\}$, A est le sous-groupe du groupe multiplicatif de $\mathbb{Z}/13\mathbb{Z}$, engendré par 5 (*A est donc formé par des multiplications par 1, 5, -1, -5 dans*$\mathbb{Z}/13\mathbb{Z}$). On a ainsi, un graphe de Cayley arête-transitif *(figure 2.1)* **[15].**

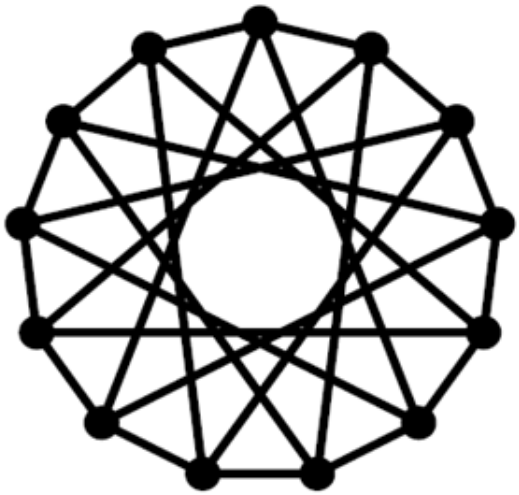

FIGURE 2.1 – Graphe de Cayley arêtes-transitif â 13 sommets

2.3.3 Propriétés

la connexité du graphe de Cayley $Cay(G, S)$, équivaut au fait que S engendre G, en d'autres termes que tout élément de G est produit d'élément de S. Si un sous-groupe A d'automorphisme de G vérifie $AS = S$, alors A induit un automorphisme de graphes de $Cay(G, S)$. Si de plus l'opération de A sur S est transitive (*c'est-à-dire ne produit qu'une orbite*), alors $Cay(G,S)$ est arête-transitif.

Il peu y avoir que le graphe de Cayley ait plus de symétries que ne laissait prévoir les automorphismes du groupe. Par exemple le graphe de la *figure2.3*, il est définit à partir de trois symétries-points, au tour des trois points non alignés de $(\mathbb{Z}/3\mathbb{Z})^2$.
Il ya 216 automorphismes, est pas seulement les 108=18∗6 prévisibles au vu des 6 bijections

affines conservant l'ensemble des 3 points **[16].**

Les graphes de Cayley sont sommet-transitifs (*vertex-symmetric*), c'est-à-dire que n'importe quelle permutation des sommets est homomorphe au graphe original. Visuellement, si on trace le graphe du réseau à partir de la donnée des relations de voisinage (*tables de routage*), on peut échanger les noms des sommets comme on veut (*sur le même dessin*), le graphe obtenu sera toujours conforme aux tables de routage.

Plus pragmatiquement, d'un point de vue P2P, cela permet que l'algorithme de routage ait le même comportement, quel que soit son point de départ. Donc tout pair peut être pris comme point de départ d'une recherche, cette recherche aura le même comportement, même coût, même qualité - *on parle qualité et coût moyen, minimal ou maximal*- **[20].**

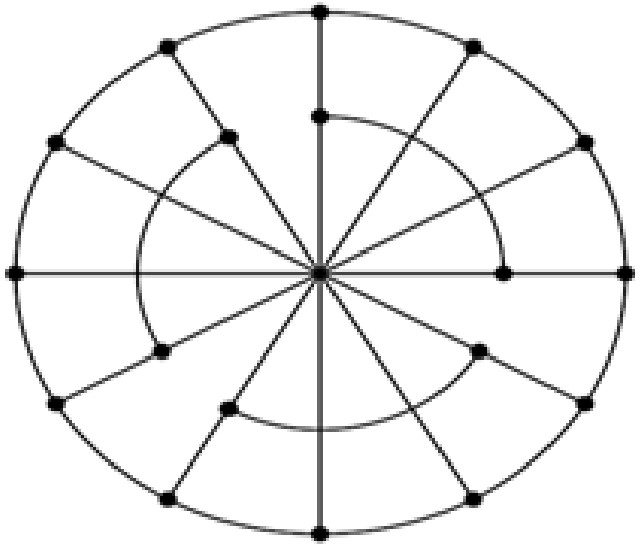

FIGURE 2.2 – Graphe de Cayley très symétrique

2.3.4 Catégories et graphes de Cayley

Graphes de Cayley coloriés associés à la présentation d'un groupe

Soit P la présentation d'un groupe Γ et on dénote par $C_p(\Gamma)$, ou par convention $C_\Delta(\Gamma)$ ou Δ est l'ensemble générateur dans P, le graphe de Cayley colorié de P pour Γ **[21].**

A chaque présentation du groupe est associé un graphe : chaque vertex correspond à un élément du groupe et les liens sont déterminés et coloriés à partir des générateurs. Si les vertex *(sommets)* v_1, v_2 correspondent aux éléments g_1, g_2 alors il y a un lien direct (*auquel on associe la couleur* h) de v_1 vers v_2 si et seulement si $g_1 h = g_2$. On a donc que $C_\Delta(\Gamma)$ est un graphe dirigé et colorié avec ensemble de vertex *(sommet)* $XC_\Delta(\Gamma)$ et l'ensemble de liens $EC_\Delta(\Gamma)$.

$$XC_\Delta(\Gamma) = \Gamma$$
$$EC_\Delta(\Gamma) = \{(g, g_\delta) | g \in \Gamma, \delta \in \Delta\}$$

Le graphe $C_\Delta(\Gamma)$. dépends non seulement du groupe, mais de l'ensemble générateur choisi pour un même groupe. L'effet d'un élément minimal ou redondant dans l'ensemble générateur est différent dans la construction du graphe comme le précise le théorème suivant :

Théorème 1 : Soit Γ un group fini *(infini)*. Un générateur h est redondant si et seulement si le résultat d'enlever tous les liens colorés avec h dans $C_\Delta(\Gamma)$ est un graphe direct fortement *(faiblement)* connecté **[18].**

Graphes de Cayley non coloriés et non orientés

Soit Δ un ensemble générateur du groupe Γ avec les conditions suivantes :

- ✓ $e \notin \Delta$, e l'élément identité du groupe.
- ✓ $\delta \in \Delta, \delta^2 \neq e, \delta^{-1} \notin \Delta$
- ✓ $\delta \in \Delta, \delta^2 = e$, chaque paire $\{g, g_\delta\}$ et $\{g_\delta, g\}$ de liens dirigés est remplacé par un seul lien non dirigé $\{g, g_\delta\}$.

Alors le pseudo graphe obtenu du graphe de Cayley colorié $C_\Delta(\Gamma)$ supprimant toutes les directions et toutes les couleurs est un graphe (*sans boucles et sans liens multiples*). Ce graphe est appelé un graphe de Cayley et est dénoté $C_\Delta(\Gamma)$ **[21].**

2.4 Exemple d'algorithme à base des graphes de Cayley

2.4.1 Diamètre et chemin optimal

Le processus du cheminement optimal d'un sommet arbitraire au sommet d'identité est équivalent à assortir une permutation donnée qui utilise les opérateurs disponibles *(générateurs)*.
Un algorithme *(SR)* qui a été déjà proposé permet d'atteindre le sommet d'identité en construisant simplement le sommet d'identité, il ne tient pas compte si le sommet de source déjà est partiellement assorti. C'est pourquoi il n'est pas optimal le plus souvent. L'algorithme suivant permet de trouver le chemin optimal.

2.4.2 Présentation des paramètres

- ✓ Les générateurs f et f^{-1} fournissent la complémentation des symboles tandis que les générateurs g et g^{-1} fournissent les décalages circulaires sans complémentation.
- ✓ Les paramètres m_1 et m_2 mesurent la plus longue séquence inachevé de symboles de gauche et droit pour un symbole spécial T_1 (*il est spécial parce qu'il occupe la première position dans le sommet d'identité de destination* I).

- ✓ Les générateurs f et g fournissent les décalages circulaires gauches tandis que les générateurs f^{-1} et g^{-1} fournissent les décalages circulaires de droite.

- ✓ $D_L(s)$ spécifie la distance gauche par rapport au sommet de départ s.
- ✓ $D_R(s)$ spécifie la distance gauche par rapport au sommet de destination **[18].**

2.4.3 Pseudo-code de l'algorithme

Dans cet algorithme, la procédure **Opt-Rout** donne correctement la distance entre le sommet de départ et le sommet d'identité (*destination*). Elle permet de déterminer le chemin optimal. On premier on compare les distances gauches et droites du sommet s par rapport au sommet de destination, donc le déplacement commence selon la plus courte distance.La longueur du chemin d'un sommet arbitraire est toujours $\leq[3n/2]$ et le diamètre du graphe est égale à $[3n/2]$ **[18].**

algorithme 1 Procédure *Opt_Rout* (s,m_1,m_1,J_R, J_L, $D_R(s)$,$D_L(s)$)

if $D_R(S) < D_L(S)$ **then**
begin
for ($i = 1$ to (J_L - 1)) **do**
Se déplacer au voisin de g du sommet courant.
for ($i = 1$ *to* $(J_L - 1) + (n - k + 1)$)**do**
if le dernier symbole du sommet courant est inachevé (*non complémenté*)**then**
se déplacer au voisin de g^{-1}.
else
se deplacer au voisin de f^{-1}.
for ($i = 1$ to ($k - J_L$ - m_1)) **do**
se déplacer au voisin de g^{-1}.
for ($i = 1$ to ($k - J_L$ - m_1)) **do**
if le premier symbole du sommet courant est inachevé ou non complémenté **then**
se déplacer au voisin de g.
else
se déplacer au voisin de f.
end.
else
begin
for ($i = 1$ to ($n - J_R$ - m_2)) **do**
se déplacer au voisin de g^{-1} du sommet courant.
for ($i = 1$ to ($n - J_R$ - m_2 + k -1)) **do**
if le dernier symbole du sommet courant est inachevé **then**
se déplacer au voisin g.
else
se déplacer au voisin f.
for ($i = 1$ to (J_R - k)) **do**
se déplacer au voisin de g du sommet courant.
for ($i = 1$ to (J_R - k)) **do**
if le premier symbole du sommet courant est inachevé **then**
se déplacer au voisin g^{-1}.
else
se déplacer au voisin f^{-1}. **end.**

Exemple : soit le graphe G_3 de la (*figure 2.3*), pour $n = 3$, qui est constitué de 24 sommets, ($n \times 2^n$ *sommets*). Les différents paramètres sont calculés comme suites :

- ✓ k : la position du premier caractère de l'identifiant du sommet identité (*i.e pour ce cas on cherche a déterminer la position de la lettre "a" dans la chaine identifiant le sommet source*), et déterminer aussi si "a" est d'une forme complémentée ou non complémentée (c.-à-d. *a ou bien* $\bar{a}$).

- ✓ m_1,m_2 : déterminent les longueurs des sous-chaines à gauche et à droite et de même forme que le premier caractère identifiant le sommet identité. (i.e. si $s = f\bar{g}hija\bar{b}cd\bar{e}$ *la source dans le graphe* $G_{10}, m_1 = 3$ (*en raison de la sous-chaine "hij"*) et $m_2 = 2$ (*en raison de la sous-chaine "cd"*).

- ✓ $D_L(s)$,$D_R(s)$: se calculent en fonction de n,k,m_1 et m_2 tel que :
  1. $D_L(s) = 2(k - m_1 - 1) + (n - k + 1)$.
  2. $D_R(s) = 2(n - k - m_2) + (k - 1)$.
  3. $D(s) = min(D_L(s), D_R(s))$.

- ✓ J_L,J_R : représentent les positions des premiers éléments des sous-chaines à droite et à gauche du premier élément de l'identifiant du sommet identité, tel que ces sous-chaines sont de même forme que ce dernier. (*i.e. pour l'exemple précédent* $s = f\bar{g}hija\bar{b}cd\bar{e}$, ($J_R$ =8) *en raison de la position du caractère "c"* et ($J_L = 3$) *en raison de la position du caractère "h"*).

2.4.4 Exécution d'un exemple

Partons du sommet ($b\bar{c}a$), pour atteindre le sommet identité (abc) du graphe précédent. Les valeurs des différents paramètres précédents sont ($s = ca\bar{b}, n = 3, k = 2m_1 = 1, m_2 = 0, J_R = 0, J_L = 1, D_R(s) = 2, D_L(s) = 3$). On a $D(s)$=min($D_R(s)$=3, $D_L(S)$=3)=2, donc on va exécuter la ligne **n°15** de l'algorithme **Opt-Rout**.

- ✓ **Ligne n°20 :** pour i :=1 $g^{-1}(ca\bar{b})$=($\bar{b}ca$), pour i :=0 $g^{-1}(\bar{b}ca)$=$a\bar{b}c$.
- ✓ **Ligne n°22 :** pour i :=1 $g(a\bar{b}c)$=($\bar{b}ca$), pour i :=0 $g(\bar{b}ca)$=$ca\bar{b}$.

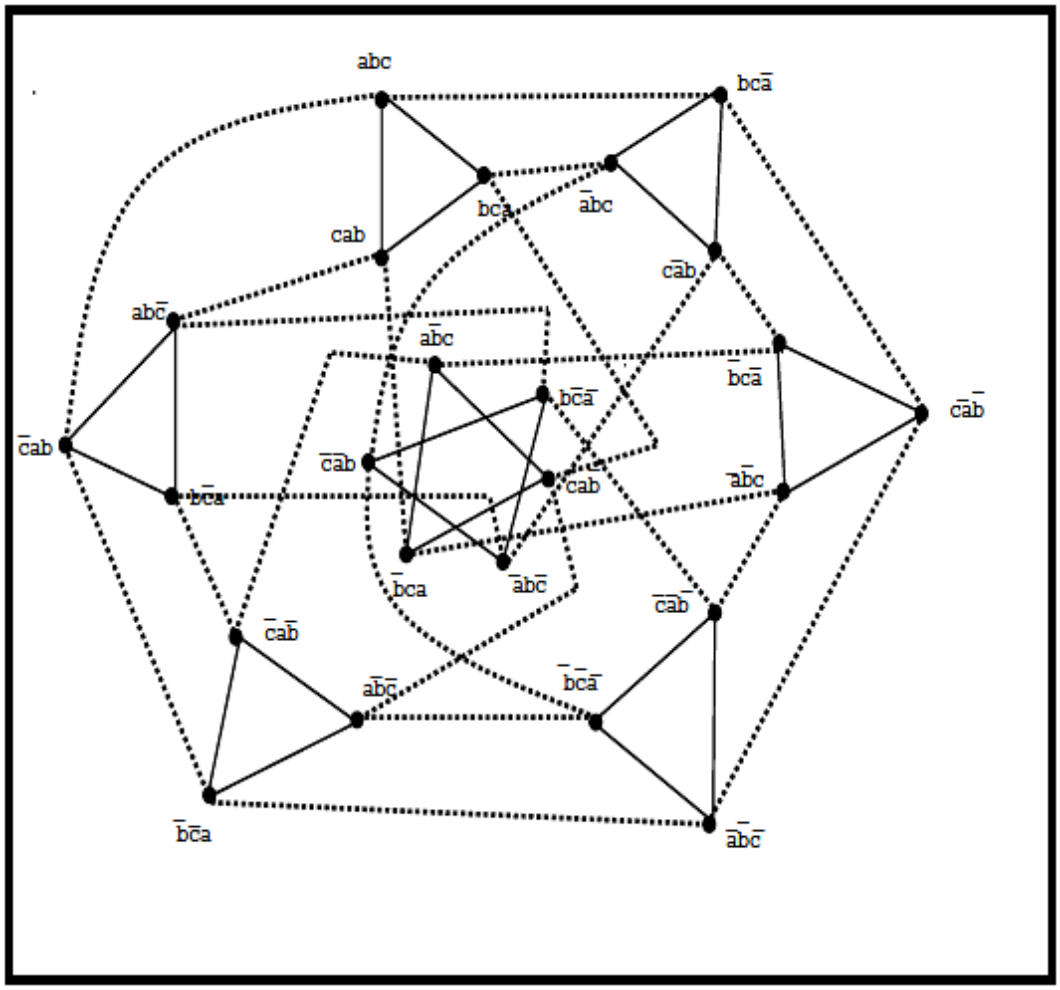

FIGURE 2.3 – Graphe de Cayley pour $n = 3$ (24 sommets)

- ✓ **Ligne n°27 :** pour i :=1 $g(ca\bar{b})$=$(a\bar{b}c)$, pour i :=0 $g(a\bar{b}c)$=$\bar{b}ca$.
- ✓ **Ligne n°29 :** pour i :=1 $f^{-1}(ca\bar{b})$=(bca), pour i :=0 $g^{-1}(bca)$=abc .

Donc le chemin qui mène de la source $(ca\bar{b})$ vers le sommet identité (abc) est : $ca\bar{b}$→bca→abc.

2.5 Comparaison des graphe de Cayley avec d'autres topologies

La table suivante présente une comparaison entre le graphe de *Cayley* $(Cay(G, S))$ avec d'autre topologies connues comme : l'*hypercube* (H_n), le graphe *circulant rcursif* $(G_{2^n,4})$, le

Star (S_n), graphe de *DeBruijn* ($B_{n,k}$), graphe de *Knödel* ($W_{n,2^n}$) et le graphe de *pancakes* (P_n).

Graphes	**Sommet**	**Diamètre**	**Degré**
$Cay(G,S)$	n	$\log_2 n$	$\|S\|$
H_n	2^n	$\log_2(2^n) = n$	n
$G(2^n, 4)$	2^n	$(3n-1)/4$	n
S_n	$n!$	$(3(n-1))/2$	$n-1$
$B_{n,k}$	n^k	k	$2n$
$W_{n,2^n}$	2^n	$(n+2)/2$	n
P_n	$n!$	$\leq(n+1)/3$	$n-1$

TABLE 2.1 – Comparaison entre les graphes de Cayley et d'autres catégories de graphes.

2.6 Conclusion

Dans ce chapitre nous avons commencé par donner quelques définitions de base sur les groupes et les graphes, spécifiquement ceux appelés graphes de Cayley. Nous avons donné leurs caractéristiques, propriétés et citer un exemple d'algorithmes (*routage optimal*) sur ces derniers.

Dans le chapitre suivant, nous allons essayer d'exploiter les propriétés des graphes de Cayley, dans le but de proposer une nouvelle méthode de localisation de ressources (*ou sommets*) dans un réseau P2P, et cela en essayant de déterminer le chemin optimal menant d'un sommet source a un sommet destination, afin d'améliorer les performances du réseau.

Chapitre 3

Proposition d'une nouvelle méthode basée sur les graphes de Cayley

3.1 Introduction

Parmi les problèmes fondamentaux des réseaux P2P, la localisation des pairs (*nœuds*) qui stockent les profils des utilisateurs ainsi l'identification des nœuds offrants des services, car dans ce type des topologies réseaux (*P2P*), il n ya pas de serveur central ainsi que la volatilité des nœuds (*des nœuds peuvent le quitter à tout moment comme d'autres peuvent le rejoindre*).

Étant donné les caractéristiques sous-jacentes des graphes de Cayley, comme le fait que ces graphes sont :

- ✓ denses (*ce qui limite remarquablement le nombre de nœud intermédiaires*),
- ✓ symetriques (*le graphe peut être devisé en deux parties similaires*),
- ✓ sommet-transitifs (*permet l'exécution du même algorithme de routage sur tous les nœuds du réseau*).

Nous allons essayer d'exploiter ces propriétés afin de proposer un algorithme ou plus proprement dit, une nouvelle méthode de localisation de ressources (*nœuds*) dans un réseau P2P.

3.2 Problématique

Il est trivial que le principal objectif dans la recherche d'un nœud (*une ressource ou un service*) dans un réseau P2P, après la localisation bien évidement, est le fait de pouvoir minimiser et optimiser le nombre de sauts effectués par l'algorithme de recherche pour atteindre la ressource en un temps moindre (*c.-à-d. atteindre la ressource en suivant le plus court chemin*) et de ce fait ne pas surcharger vainement le réseau par des messages des requêtes.

Dans l'exemple précédent présenté dans le chapitre 2, l'algorithme de recherche ***Opt-Rout*** à effectué trois tentatives non fructueuses pour enfin trouver le nœud identité â la quatrième exécution.

Nous allons essayer d'exploiter les propriétés des graphes de Cayley, a fin de proposer une autre méthode de recherche basée sur une famille de ces graphes qui est de degré quatre (04). Cette méthode sera décrite en détail dans ce qui va suivre.

3.3 Proposition

L'idée est de modéliser le réseau P2P sous forme d'un graphe de Cayley sous jacent de l'ensemble G qui est le groupe additif de $\mathbb{Z}/n\mathbb{Z}$, avec $G = \{0, \ldots, n\}$de nombre entiers tout en fixant le sous ensemble S tel que $S = +1$, -1, $+2$, -2, ainsi, S garantie la symétrie du graphe grâce aux inverse des éléments $+1$ et $+2$.

Le choix s'est porté sur l'ensemble s=($+1$,-1,$+2$ et -2) afin de garantir que chaque nœud sera connecté a ses voisins directs de gauche et de droite et aussi des voisins a deux (02) sauts qui est une propriété très utile en cas d'une requête a un voisin proche ou bien en cas de défaillance. (*Ces deux propriétés seront plus explicites dans ce qui va suivre*).

Chaque nœud du réseau aura comme identificateur un entier naturel (*qui peut être écrit aussi sous sa forme binaire* $a_1a_2.... \ a_{n-1}a_{log_2(n)}$) car il faut éxactement $log_2(n)$ bits pour représenter n éléments.

Remarque : dans la définition même du graphe de Cayley, et plus particulièrement celle du sous ensemble S, cet ensemble doit être fini et symétrique, (*i.e. si un élément g est dans S donc son inverse* g^{-1} *est dans S*).

En remarque que deux des éléments de S sont négatifs, réellement " -1" correspond a $(-1+n)modulo(n)$ et " -2" correspond a $(-2+n)modulo(n)$, avec n : le nombre de nœuds du réseau (*ou éléments de G*). Tous les nœuds sont de même degré $d = 4$.
Dans ce qui suit, nous allons décrire l'algorithme de recherche et le dérouler avec plusieurs exemples.

3.4 Description de l'algorithme de recherche

- ✓ Nœud[i] : nœud actuel ou le nœud qui lance la localisation.
- ✓ *key* : désigne l'identificateur du nœud à localiser.
- ✓ $Ch[i]$: un tableau qui va contenir les nœuds constituants tout le chemin de la source a la destination.
- ✓ ID(identité) : identifiant du nœud N_0 considéré comme identité (*ou origine).*
- ✓ α :=(ID(identite)-ID(source)mod(n) : cette instruction permet de calculer la distance (*gauche*) entre la source et l'identité N_0.
- ✓ β :=(ID(identite)-ID(destination)mod(n) : cette instruction permet de calculer la distance (*gauche*) entre la destination et l'identité.
- ✓ $|\alpha$-$\beta|$: détermine quelle distance (*gauche ou droite*) entre la source et la destination, est la plus petite.
- ✓ $[ID(source) > ID(destination)]et[ID(source) < ID(destination)]$: permet de déterminer de quelle manière choisir le nœud suivant (*max ou min c.-à-d. (ligne 15) ou bien (ligne 22)*) de l'algorithme.

Ainsi, on détermine dans quelle partie du graphe se situent les deux ressources (*droite ou gauche par rapport à l'identité*) et le chemin à suivre sera le plus court de ces deux distances gauche et droite.

En terme binaire, si on affecte à chaque nœud un identificateur du type $(a_1a_2....a_{n-1}a_n)$, la détermination du prochain nœud â ajouter au chemin sera suite a la comparaison entre les bits du poids fort de tous les voisins directs c.-à-d. si $voisin_1$ a comme identificateur

algorithme 2 ***Recherche*** (nœud[i],*key*)

1: Début
2: **(**$\alpha := (ID(identite) - ID(source) mod(n)$**)** ;
3: **(**$\beta := (ID(identite) - ID(destination) mod(n)$**)** ;
4: **Si** nœud[i] (*c.-à-d. nœud actuel*) possède la clé (*key*) **alors**
5: **Retourner** (nœud[i]) (*nœud actuel*) -*destination trouvée*-
6: $ch[j] \leftarrow noeud[i]$;
7: **Sinon**
8: $ch[j] \leftarrow noeud[i]$;
9: **Si** (un des voisins directs du nœud[i] a la clé *key*) **alors**
10: $j \leftarrow j + 1$; $ch[j] \leftarrow voisin[i]$ -*mise a jour du chemin*-
11: **Retourner** (ch[i]) ;
12: **Sinon**
13: **Si [**(|α-β|<$n/2$) **et** [ID (source) <ID (destination)]**ou** (|α-β|> $n/2$)**et** [ID (source) > ID (destination)**] alors**
14: **Pour i** :=1 a 4 **faire** -*vérification des voisins directs de la source*-
15: *noeud_suivant* := **max**[$(voisin[i] - key)$]mod(n) ;
16: **Finpour ;**
17: $j \leftarrow j+1$; $Ch[j] \leftarrow noeud\_suivant$; -*mise à jour du chemin*-
18: **Recherche** (*noeud_suivant*, *key*) ; -*appel de la même fonction de recherche*-
19: **Sinon**
20: **Si [**(|α-β|<$n/2$)**et**[ID (source) >ID (destination)] **ou** (|α-β|>=$n/2$)**et** [ID (source) < ID (destination)**] alors**
21: **Pour i** :=1 a 4 **faire** -*vérification des voisins directs de la source*-
22: *noeud_suivant* := **min** [(voisin[i] - *key*)] mod(n) ;
23: **Finpour ;**
24: $j \leftarrow j+1$; $Ch[j] \leftarrow noeud\_suivant$; -*mise à jour du chemin*-
25: **Finsi ;**
26: ***Recherche*** (*noeud_suivant*, *key*) ; -*appel de la même fonction de recherche*-
27: **Finsi ;**
28: **Finsi ;**
29: **Finsi ;**
30: **Retourner** (ch[j]) ; -*tout le chemin de la source vers la destination*-
31: **Fin.**

(1010≡$a_1a_2a_3a_4$) et le $voisin_2$ a comme identificateur (1100≡$b_1b_2b_3b_4$), l'algorithme va comparer entre a_1 et b_1 ensuite a_2 et b_2 puis a_3 et b_3 et en fin a_4 et b_4.

Dans le cas **min**, par exemple, il va élire $voisin_1$ comme prochain nœud car a_1=b_1, mais a_2<b_2. Ici nous avons choisi d'utiliser les entiers, car ils sont plus faciles â manipuler et plus explicitent.

3.5 Propriétés

Cet algorithme à une complexité en termes d'opérations élémentaires :

- le meilleur des cas (*la destination est le nœud effectuant la recherche ou bien elle (i.e. la destination) se trouve parmi ses voisins directs*) : elle est de l'ordre de $O(1)$.
- le cas ou la destination est le nœud le plus loins du réseaux et le nombre de défaillance est nulle : $O(K[\log_2 n])$
- le pire des cas (*la destination est le nœud le plus éloigné par rapport à la source suite a deux défaillances successives*) : elle est de l'ordre de $O(K.2[\log_2 n] - 1)$.

K : le nombre total d'instruction de l'algorithme de recherche.

3.6 Déroulement de l'algorithme

1. Soit le groupe additif G=$(G,+)$, avec $G = \{0, \ldots, 15\}$, et soit S la famille génératrice tel que $|S|$= 4, (S= -1, +1, -2, +2). Notons que " -1" est l'élément inverse de l'élément " $+1$" qui est égale à "15" et " -2" est l'inverse de " $+2$" qui est égale à "14".

2. le choix s'est porté sur les éléments $+1$, -1, $+2$ et -2 afin de garantir que chaque nœud sera connecté à son successeur et prédécesseur, ainsi qu'aux voisins à deux sauts, ce qui garantie un court chemin en cas recherche d'un nœud très proche.

La construction du graphe est comme suite :

- ✓ Chaque nœud est identifié par un entier.
- ✓ Les nœuds voisins directs sont numérotés de 1 à 4 dans la table d'indices de chaque nœud.

- ✓ Le graphe contient $(2n)$ arêtes.
- ✓ Le diamètre du graphe (*la distance entre les deux nœuds les plus éloignés du réseau*) dans le cas ou tous les nœuds sont actifs (*aucune défaillance*)est de :$\log_2 n = \log_2 16 = 4$.
- ✓ Le diamètre du graphe dans le cas ou on aura deux nœuds défaillants et qui sont voisins dirècts (*c.-à-d. deux pannes successives*) sera égale à :$2[(\log_2 n)] - 1$.

Le graphe issu de ces conditions est illustré sur la figure 3.1 qui contient 16 nœuds de degré d=4.

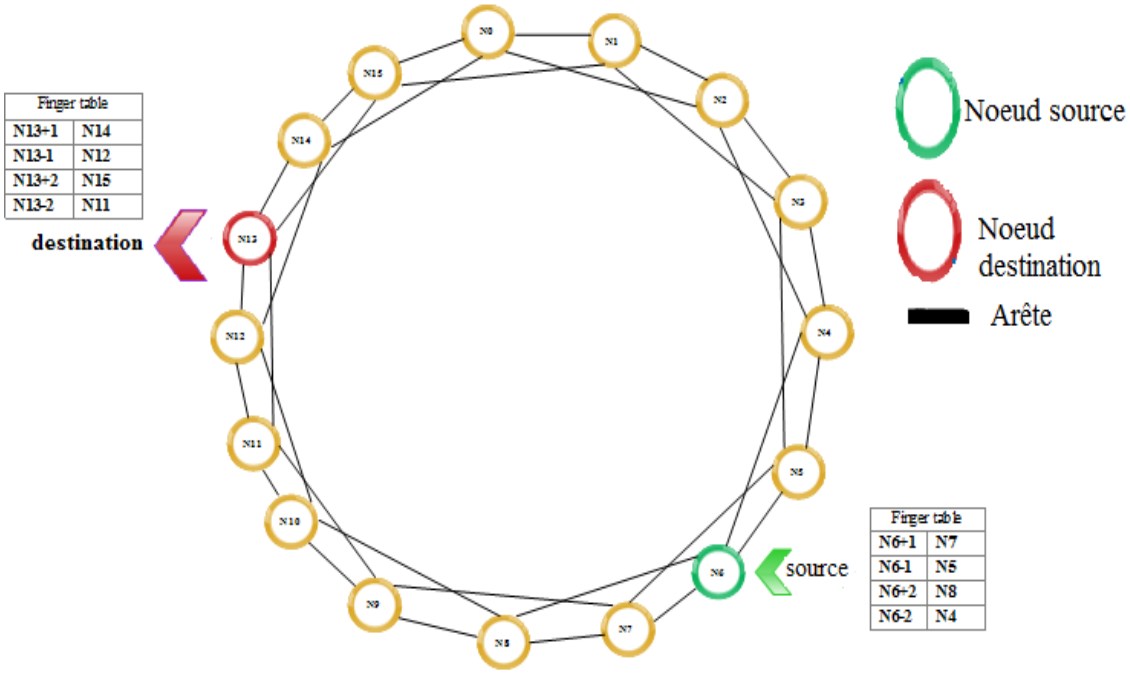

FIGURE 3.1 – Graphe de Cayley associe a $G_{16,4}$

Exemple 01 **:**(cas sans défaillance)

Soit le nœud N_6 cherche la clé ($key = 13$) c.-à-d. le nœud N_{13}.

- ✓ Dans la (ligne 4) de l'algorithme *Recherche*, (N_6 <> N_{13}), donc on va exécuter le **sinon** (ligne 7), le premier nœud du chemin est N_6 et les voisins dirècts de N_6 sont : N_4 ,N_5, N_7 et N_8.
- ✓ La condition de la (ligne 9) n'est pas satisfaite, donc on exécute le **sinon** (ligne 12).
- ✓ α :=(ID(N_0)-ID(N_6)mod(16)= 10 et β :=ID(N_0)-ID(N_{13})mod(16)=3.

- ✓ ($|\alpha$-$\beta|$= 7)<($n/2 = 16/2 = 8$) et ID(N_6)<ID(N_{13}) donc on va exécuter la boucle **pour** de la (ligne 14).
- ✓ **max**(voisin[i]-key)mod(n) :=max[$(4 - 13mod(16 = 7))$, $(5 - 13mod(16) = 8))$, $(7 - 13mod(16) = 10))$, $(8 - 13mod(16) = 11))$] qui est 11 qui correspond au nœud N_8 .
- ✓ Le nouveau nœud du chemin sera N_8.
- ✓ On refait la même chose avec le nœud N_8 (i.e. **max**(voisin[i]-key)mod(n) :=**max**[$(9 - 13mod(16) = 12)$, $(10 - 13mod(16) = 13)$,$(7 - 13mod(16) = 10)$,$(6 - 13mod(16) = 9)$] qui est 13 correspondant au nœud N_{10}.
- ✓ Le nouveau nœud du chemin est N_{10}.
- ✓ On refait la même chose pour le nœud N_{10} et le ***max*** correspond au N_{12}.
- ✓ Pour N_{12}, la condition de la (ligne 9) est vérifiée car N_{12} figure parmi ses voisins directes. La destination est trouvée (N_{13}) donc on termine l'algorithme on éxécutant l'instruction le la (ligne 30) qui va reconstituer tout le chemin qui mène de N_6 vers N_{13} :($N_6 \rightarrow N_8 \rightarrow N_{10} \rightarrow N_{12} \rightarrow N_{13}$).

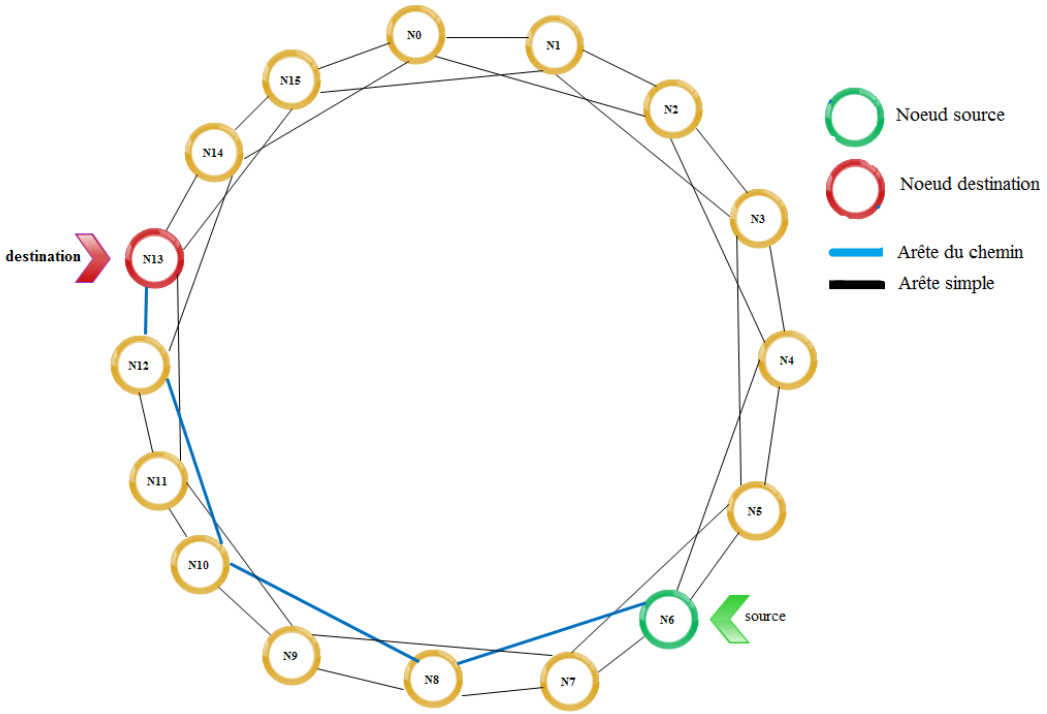

FIGURE 3.2 – Cas sans défaillance

Supposons maintenant que pour le même cas, (*recherche de la clé* 13 *en départ de* N_6) avec une défaillance de l'un des nœuds constituant le plus court chemin calculé précédemment. Par exemple le nœud N_{12} est en panne. La question a la qu'elle on doit répondre est : cet algorithme peut-il trouver un autre chemin qui soit le plus court en dépit de cette panne ?

Exemple 02 :(cas avec une (01)défaillance)

Dans cet exemple, le nœud N_{12} appartenant au plus court chemin entre N_6 etN_{13} est supposé défaillant.

- ✓ Dans la (ligne1), N_6<>N_{13}, donc on va exécuter le **sinon** (ligne 5), le premier nœud du chemin est N_6 et les voisins directs de N_6 sont N_4,N_5, N_7 et N_8.
- ✓ La condition de la (ligne 6) n'est pas satisfaite, donc on exécute le **sinon** (ligne 10).
- ✓ α :=(ID(N_0)-ID(N_6)mod(16)=10 et β :=ID(N_0)-ID(N_{13})mod(16)=3.
- ✓ ($|\alpha$-$\beta|$= 7)<(n/2=16/2=8) et ID(N_6)<ID(N_{13})donc on va exécuter la boucle **pour** de la (ligne 12).
- ✓ **max**(voisin[i]-key)mod(n) :=max[$(4-13) mod(16) = 7$), $(5-13 mod(16) = 8)$, $(7-13 mod(16) = 10)$, $(8-13 mod(16) = 11)$] qui est 11 qui correspond au nœud N_8.
- ✓ Le nouveau nœud du chemin sera N_8.
- ✓ On refait la même chose avec le nœud N_8 (i.e. max(voisin[i]-key)mod(n) :=max[(9-13mod(16)=12), (10-13mod(16)=13),(7-13mod 16=10),(6-13mod 16=9)] qui est 13 correspondant au nœud N_{10}.

- ✓ On refait la même chose pour le nœud N_{10}. (**max**[(9-13mod(16)=12), (11-13mod(16)=14), (8-13mod(16)=11), (12-13mod(16)=15)] correspond au nœud N_{12}.
- ✓ Dans ce cas N_{12} sera le nouveau nœud du chemin, mais on a N_{12} est un nœud défaillant alors on va choisir le deuxième max entre les résultats trouvés qui est N_{11}.
- ✓ Pour N_{11}, la condition de la (ligne 7) est vérifiée car N_{13} figure parmi ses voisins directes. La destination est trouvée (N_{13}) donc on termine l'algorithme on exécutant l'instruction de la (ligne 29) qui va construire le chemin qui mène de N_6 vers N_{13}(N_6→ N_{08}→N_{10}→N_{11}→N_{13}).

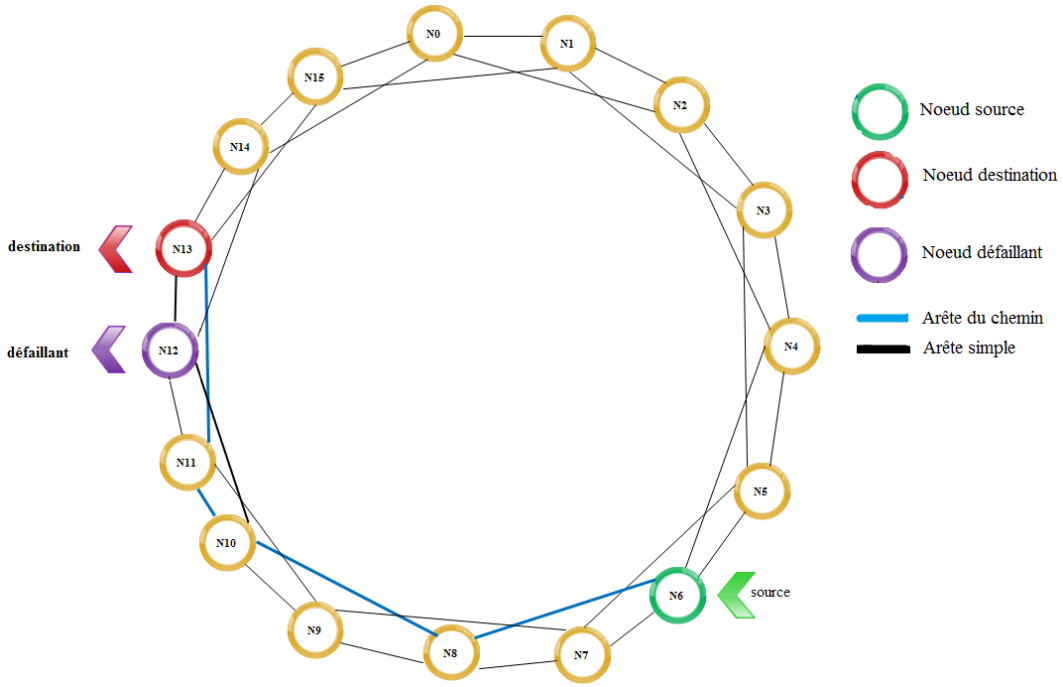

FIGURE 3.3 – Cas avec une défaillance

3.7 Comparaison avec l'algorithme *Opt_Rout*

Nous allons faire une comparaison entre l'algorithme recherche et l'algorithme *Opt-Rout* en se basant sur les critères suivants :

- ✓ Complexité :c'est le nombre d'instructions a éxécuter dans le pire des cas.
- ✓ Diamètre : c'est la plus courte distance entre les nœuds les plus éloignés du graphe.
- ✓ Tolérance aux fautes : le nombre maximum de pannes endépit des quels, l'algorithme fonctionnera toujours.
- ✓ Temps de réponse : le temps que prend (*nécessite*) l'algorithme pour trouver le chemin optimal lors d'une recherche.

Après avoir fait la comparaison entre les deux algorithmes (*Recherche, OptRout*) nous avons constaté que l'algorithme que nous avons proposé (*algorithme Recherche*) est plus efficace (*temps de réponse court et minimum d'arêtes à parcourir*) et repends plus au besoin en ce qui concerne la recherche du chemin optimal.

Critères	**Algorithme Recherche**	**Algorithme Opt-Rout**
Complexité	$O(K.\log_2 n)$	n
Diamètre	$\leq(\log_2 n)$	$\leq(3n/2)$
Tolérance aux fautes	Maximum (< 4 panes)	Maximum (< 4 panes)
Temps de réponse	Court	long(*a cause des retours arrières.*)
Nombre d'arêtes à parcourir	Le minimum d'arcs à parcourir	Plus d'arcs à parcourir

TABLE 3.1 – Comparaison entre l'algorithme recherche et l'algorithme optimal-route

3.8 Conclusion

Dans ce chapitre, nous avons exploité les différentes propriétés des graphes de Cayley(*symétrie, arête-transitif et sommet-transitif*), afin de proposer une solution pour le problème de la découverte et la localisation de ressources (*nœuds*) dans les réseaux P2P, ce qui a abouti à une nouvelle méthode (*nouvel algorithme de routage*) à base des graphes de Cayley. Cette dernière à aussi pour but de trouver le plus cout chemin entre deux nœuds dans le réseau en un temps remarquablement réduit. Dans le chapitre qui suit, nous allons réaliser une application pour simuler cette nouvelle méthode.

Chapitre 4

Conception et réalisation

4.1 Introduction

Après avoir mis au point, dans le chapitre précédent, une nouvelle méthode de localisation de nœuds dans un réseau Peer to Peer modélisé sous un graphe de Cayley, nous allons décrire les étapes de conception et de réalisation d'une application qui simulera cette méthode.

Pour la modélisation, nous avons opté pour le langage **UML** et pour la réalisation de l'application nous avons choisi un outil parmi les plus utilisés qui est le langage **Java** sous l"environnement ***Netbeans***, car il nous semble le plus adéquat, orienté objet et très explicite.

4.2 Outils de modélisation et de développement

Dans le processus de conception et de réalisation de l'application, nous avons eu recours à deux outils primordiaux qui sont :

4.2.1 Langage de modélisation unifié *(UML2.0)*

(*Unified Modeling Language*) est le langage d'analyse et conception orienté objet le plus en vigueur et le plus utilisé en ce moment. Comme son nom l'indique, il est issu de l'unification et la standardisation de plusieurs méthodes objet qui existaient avant lui et cela dans la vocation de créer un langage commun à la plus part des concepteurs **[22].**

Avant l'apparussions d'UML, il existait une cinquantaine de méthodes de conception objet, mais trois d'entre étaient les plus diffusées, qui sont : OMT (*Object Modeling Technique*)**[22]**, BOOCH et OOSE(*Object Oriented Software Engineering*) **[23]**, leurs fondateurs respectifs (*Jim Rumbaugh, Grady Booch,Ivar Jacobson*) se sont réunis au sein d'une société (*Rational Software*) dans l'objectif de fusionner leurs méthodes et c'est ainsi que naquit UML qui s'est amélioré au fur et à mesure que le temps passe pour paraitre dans plusieurs versions (*notamment la version 2*) et s'est imposé comme standard à utiliser en tant que langage de modélisation objet.

UML2 comporte treize (13) diagrammes qui sont utilisés dans la description d'un système, ces diagrammes sont regroupés dans deux grands ensembles :

- Diagrammes structurels : ils sont au nombre de six et ils représentent l'aspect statique d'un système.
- Ils sont sept et ils représentent la partie dynamique d'un système réagissant aux événements et aux attentes de résultats de la part des utilisateurs.

Dans notre modélisation, nous nous somme servi des trois diagrammes suivants : diagramme de séquences, diagramme de classe et le diagramme de cas d'utilisation.

4.2.2 Outils de développement

• Langage de programmation

Étant donné notre conception faite avec UML qui est orienté objet nous avons adopté le langage *JAVA*. La particularité principale de *Java* est que les logiciels écrits dans ce langage sont très facilement portables sur plusieurs systèmes d'exploitation tels qu'UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. C'est la plate forme qui garantit la portabilité des applications développées en Java.

Le langage reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a été épuré des concepts les plus subtils du C++ et à la fois les plus déroutante, tels que les pointeurs et références, et l'héritage multiple remplacé par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de

sorte qu'en Java, tout est objet à l'exception des types primitifs (*nombres entiers, nombres à virgule flottante, etc*) **[23]**.

• **Environnement de développement**

NetBeans est un environnement de développement intégré (*EDI*[1]), placé en open source par *Sunmicrosysteme* en juin 2000 sous licence CDDL et GPLv2 (*Common Development and Distribution License*). En plus de Java, NetBeans permet également de supporter différents autres langages, comme *Python, C, C++, JavaScript, XML, Ruby, PHP et HTML*. Il comprend toutes les caractéristiques d'un IDE moderne (*éditeur en couleur, projets multi_langage, refactoring, éditeur graphique d'interfaces et de pages Web*).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (*sur x86 et SPARC*), Mac OS X ou sous une version indépendante des systèmes d'exploitation (*requérant une machine virtuelle Java*). Un environnement Java Development Kit JDK [2]est requis pour les développements en Java. NetBeans constitue par ailleurs une plate-forme qui permet le développement d'applications spécifiques (*bibliothèque Swing Java*). L'IDE NetBeans s'appuie sur cette plate forme. L'IDE Netbeans s'enrichit à l'aide de plugins **[24]**.

4.2.3 Edraw Max

est une application spécialisée dans le dessin vectoriel qui permet la conception des schémas graphiques, des plans et des diagrammes. Le logiciel agit spécialement sur le traitement de ces images, leur alignement, le rendu et autres. Grâce une riche bibliothèque on dispose d'une multitude de modèles qui nous ont servi d'inspiration. **[25].**

4.3 Description de l'application

Afin de bien apercevoir le résultat de recherche, nous avons modélisé le réseau Peer to Peer sous forme de graphes de Cayley et nous avons réalisé une application qui simule une requête

1. environnement de développement intégré.
2. Java développement kit.

de recherche dans ce type de réseaux. Nous avons implémenté l'algorithme de recherche décrit dans le chapitre précédent et l'application résultante comporte les fonctionnalités suivantes :

- ✓ Authentification de l'utilisateur par l'introduction d'un nom d'utilisateur et d'un mot de passe.
- ✓ Introduction des identificateurs (*clés*) noeuds du graphe de Cayley associé (*c.-à-d. les numéros des noeuds*).
- ✓ Introduction des numéros (*clés*) des noeuds qu'on voudra déclarer comme défaillants (*c.-à-d. le noeud est connecté, mais ne peu pas acheminer la requête et faire suivre le message*).
- ✓ Affichage du graphe de Cayley associé.
- ✓ Introduction de l'identifiant du noeud source ainsi que l'identifiant du noeud destination.
- ✓ Affiche le chemin complet de la source vers la destination (*tous les noeuds qui constituent le chemin*).

4.4 Modélisation avec UML2.0

Dans ce qui suit, nous allons modéliser la proposition avec le langage UML et cela on se servant des trois diagrammes suivants : le diagramme de séquences, le diagramme de classe et le diagramme de cas d'utilisation.

4.4.1 Diagramme de cas d'utilisation

il constitue une vue d'ensemble des différentes actions menées par l'utilisateur (*manipulation*) et les réactions du système a ces actions. Ce diagramme est illustré dans la figure 4.1.

4.4.2 Diagramme de séquences

Ce diagramme permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie d'utilisation des opérations en interaction avec les objets et la collaboration entre eux. La figure 4.2 illustre les séquences d'utilisation de l'application en commençant de l'authentification jusqu'à atteindre le résultat (*c.-à-d. afficher le chemin de la source a la destination*).

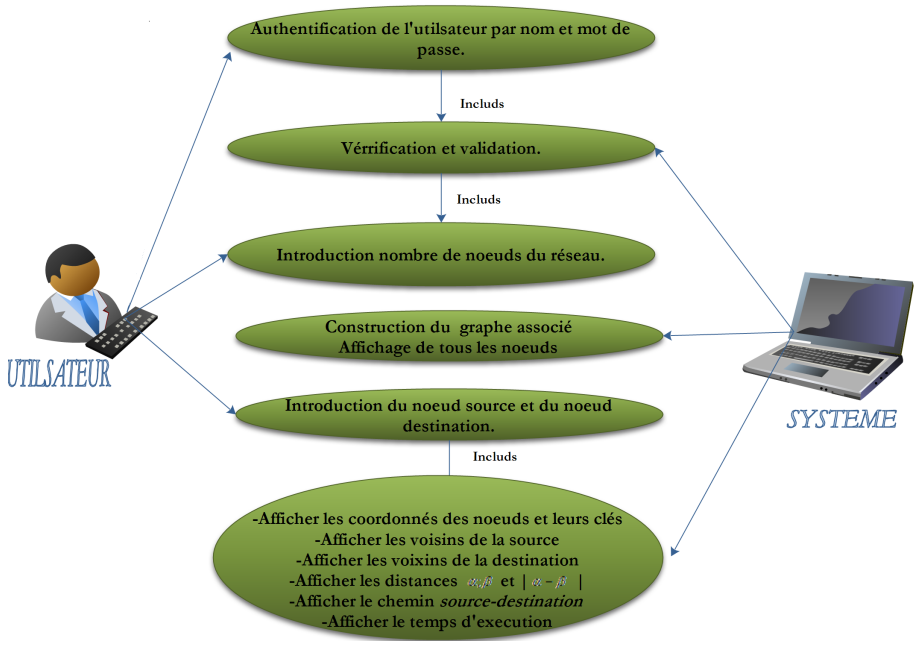

FIGURE 4.1 – Diagramme de cas d'utilisation.

4.4.3 Diagramme de classe

Il représente la description et la collection d'éléments statiques de la modélisation qui montre la structure d'un modèle de données et de traitements. Cette représentation est centrée sur les concepts de classes et d'associations, chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes. Les traitements sont matérialisés par des opérations.

- ✓ **Classe Utilisateur** : nous permet la saisie du nombre de nœuds du réseau, pour la gestion du positionnement des nœuds dans le graphe.
- ✓ **Classe Système** : cette classe permet de créer la topologie du réseau à l'aide du nombre de nœuds et d'arêtes associées (*construction du graphe*), récupère l'identifiant de la source et de la destination, affiche leurs coordonnées respectives ainsi que leurs voisins dirèctes.

  −Trouve le plus court chemin de la source vers la destination et calcule le temps d'exécution.
- ✓ **Classe Graphe** : contiens les informations relatives aux nombre de nœuds et d'arêtes du graphe.

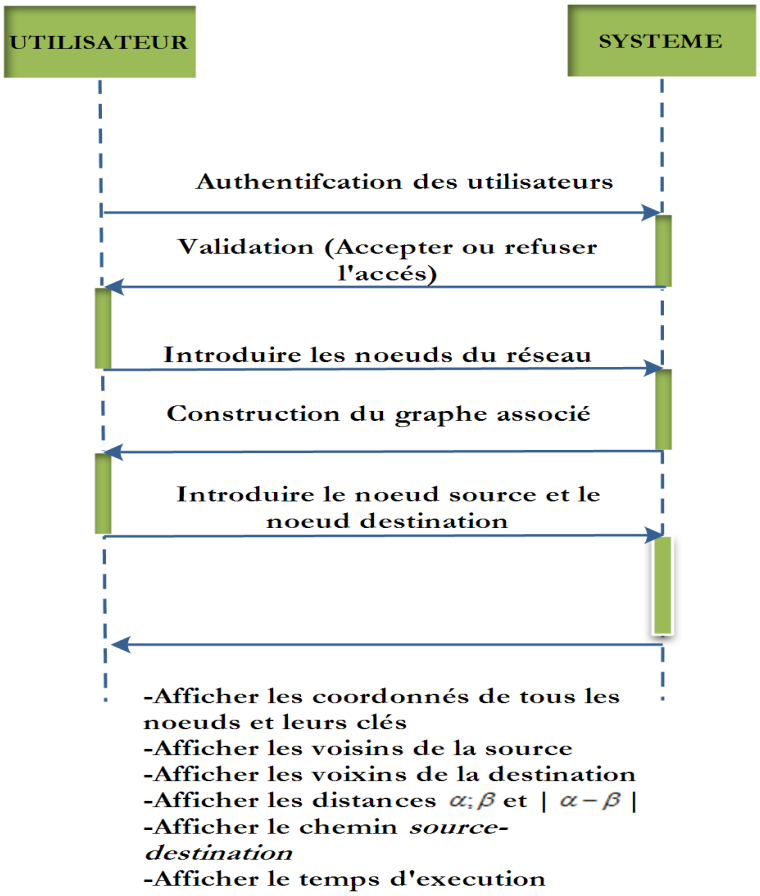

FIGURE 4.2 – Diagramme de séquences.

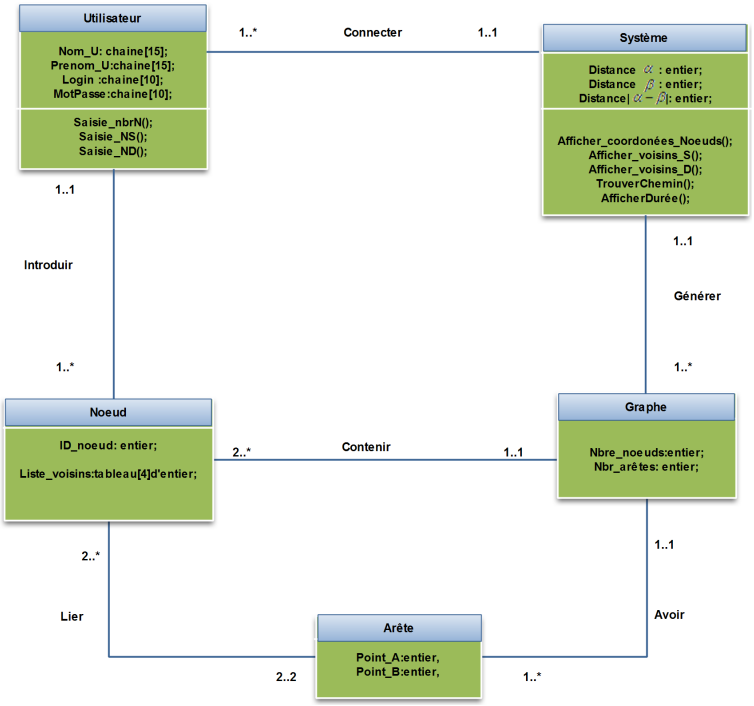

FIGURE 4.3 – Diagramme de classes.

- ✓ **Classe Nœud** :contiens les informations relatives a un nœud.
- ✓ **Classe Arête** : cette classe contiens les informations relatives aux deux nœuds aux quels elle est connectée dans le graphe.

4.5 Réalisation

Dans la phase conception, nous avons pu cerner les principaux objectifs de notre application, à présent nous allons commencer sa réalisation qui consiste à définir ces différentes interfaces et donner brievement leurs principes de fonctionnement.

4.5.1 Interface de la page d'accueil

Cette interface est illustrée dans la *figure 4.4*, elle contient quelques informations à propos de l'application et sons utilité ainsi que l'intiltulé de notre thème...etc.
Elle nous permet aussi les fonctions suivantes :

- ✓ **Suivant** : menu permettant d'accèder à l'espace d'authentification.

FIGURE 4.4 – Fenêtre de chargement.

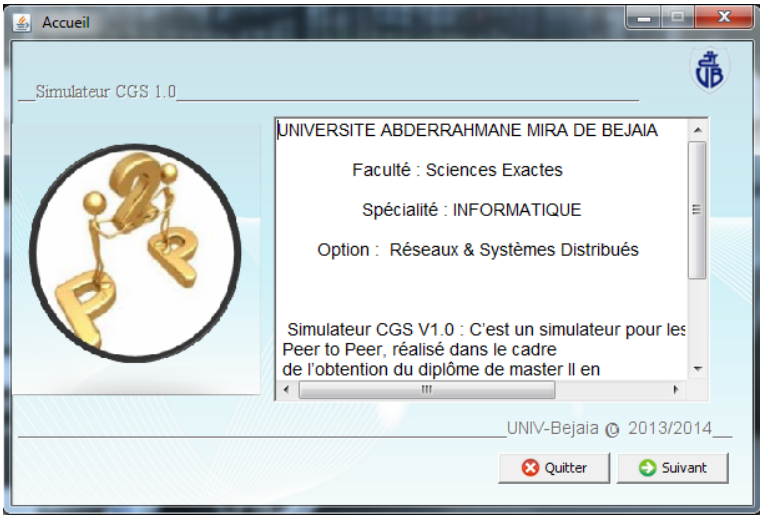

FIGURE 4.5 – Fenêtre d'accueil.

✓ **Quitter** : menu permettant de quitter et fermer définitivement l'application.

4.5.2 Interface d'authentification

Dans cette fenêtre, l'utilisateur peut s'authentifier afin d'accéder à l'interface de simulation. Elle est constituée de plusieurs zones de textes (*login et mot de passe*) qui permet à l'utilisateur de saisir ses informations avec les quelles il serra authentifié et pourra accéder a l'espace de simulation.

Elle offre aussi les choix suivant :

FIGURE 4.6 – Fenêtre d'authentification.

✓ **Annuler** : permet le retour a la fenêtre d'accueil.

✓ **Valider** : permet d'accèder a l'espace de simulation.

4.5.3 Interface de simulation et de recherche

Dans cette interface, l'utilisateur est déjà authentifié, actuellement il est capable d'introduire un nombre de nœuds afin de construire le graphe et d'effectuer une recherche.

Cette fenêtre comporte un volet a plusieurs boutons et champs de textes :

✓ **Nombre de station** :champs pour introduire le nombre de nœuds du réseau.

✓ **La Construction** :on a le choix entre les deux algorithmes, en cliquant les boutons :
- Notre algorithme, le système construit le graphe de Cayley associé au nombre de nœuds précédemment introduit relatif a notre algorithme, ou bien
- Opt-Rout, recherche selon l'algorithme **Opt-Rout**.

✓ **Champs Source -> Destination** :pour indiquer le nœud source et le nœud destination.

✓ **Simuler** : Ce bouton permet de simuler une recherche après avoir saisi le nœud source et le nœud destination.

✓ **Précédent** : ce bouton nous renvoi vers la fenêtre d'authentification.

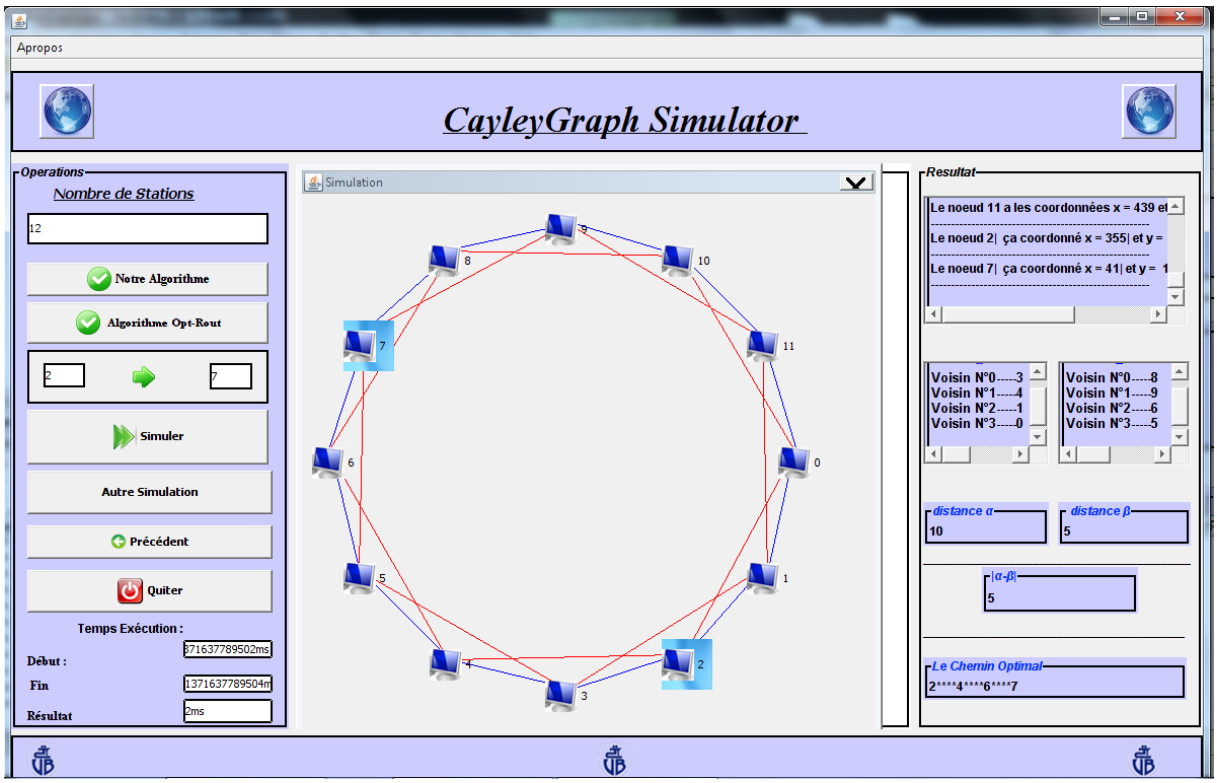

FIGURE 4.7 – Fenêtre de simulation.

✓ **Autre simulation** : Permet d'autres simulations (*construir d'autres graphes et effectuer une recherche sans avoir a redémarrer l'application*).

✓ **Quitter** :Permet de quitter la simulation et l'application en général.

✓ **Temps d'exécution** :il comporte trois champs, le premier pour récupérer le temps système lors de l'appel de l'algorithme, le second représente le temps courant du système a la fin de l'exécution de l'algorithme et le troisième donne la durée de l'exécution de

l'algorithme (*c.-à-d. le temps qu'a nécessité la recherche*).

Sur le front droit de l'espace de simulation, on vois le volet champs de textes, dans lesquels sont affichés :

- ✓ La totalité des noeuds du réseau avec leurs coordonnées et leurs clefs.
- ✓ Les voisins directs du noeud effectuant la requête ainsi que ceux du noeud destination.
- ✓ Les paramètres α, β ; et $|\alpha-\beta|$.
- ✓ La totalité du chemin optimal de la source vers la destination.

4.6 Évaluation de la solution à travers l'application

Dans cette partie du document, nous avons effectué une série de tests, de recherche dont le but est de faire apparaitre les performances de la méthode proposée.

4.6.1 Nombres de nœuds Vs temps d'exécution

Afin de mesurer les performances de l'algorithme proposé et de mettre en évidence l'influence du nombre de nœuds et du nombre de sauts sur le temps d'exécution (*ou temps de recherche du même nœud destination (*25*) en partant du même nœud source (*02*)*), nous avons effectué cinq 05 tests. Les résultats sont représentés dans la table (4.1).

Nombre de nœuds du graphe	**Temps d'éxécution** *(ms)*
$Test_1$=$G_{(64,4)}$ ($n = 64$ nœuds)	8
$Test_2$=$G_{(128,4)}$ ($n = 128$ nœuds)	8
$Test_3$=$G_{(192,4)}$ ($n = 192$ nœuds)	8
$Test_4$=$G_{(256,4)}$ ($n = 256$ nœuds)	8
$Test_5$=$G_{(320,4)}$ ($n = 320$ nœuds)	8

TABLE 4.1 – Le temps d'éxécution Vs nombre de nœuds.

Interprétation

- ✓ Dans le diagramme de la *figure 4.8*, on remarque clairement que le nombre de nœuds du réseau n'a pas d'influence sur le temps d'exécution de l'algorithme,

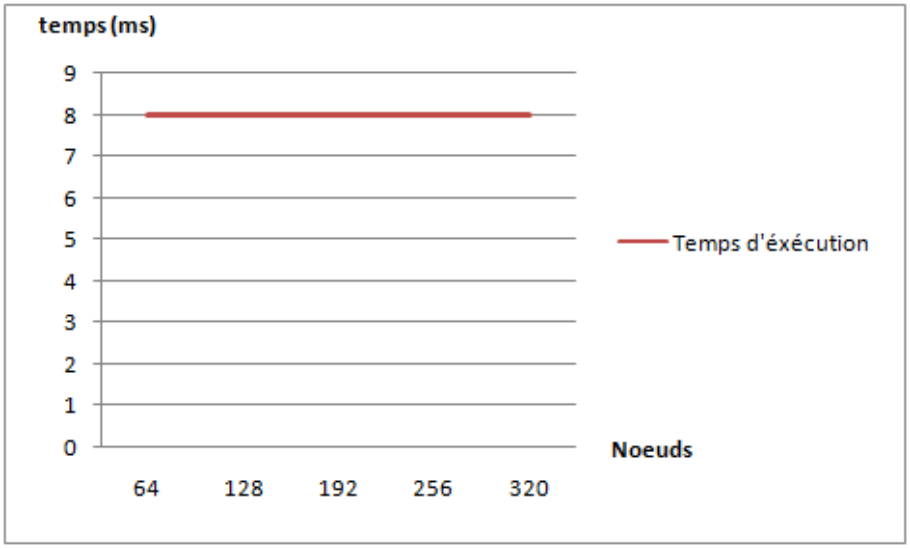

FIGURE 4.8 – Diagramme du temps d'exécution Vs nombre de nœuds.

car le nombre de sauts entre la source et la destination n'a pas changé (*même source, même destination*).

4.6.2 Nombres de sauts Vs temps d'exécution

Pour voir l'impacte du nombre de sauts sur le temps d'exécution de notre algorithme, noua avons effectuer différents tests avec tout en gardant le même nœud source, mais en variant le nœud destination (*c.-à-d. la distance ou nombre de sauts entre les deux nœud.*)
La table 4.2 récapétule les résultats obtenus.

Nombre de nœuds	**Source⇒Déstination**	**Nombre de sauts**	**Temps d'exécution** *(ms)*
G_1=$G_{(24,4)}$	2⇒11	5	4
G_2=$G_{(64,4)}$	2⇒30	14	9
G_3=$G_{(128,4)}$	2⇒55	26	15
G_4=$G_{(256,4)}$	2⇒120	59	41
G_5=$G_{(512,4)}$	2⇒250	124	219

TABLE 4.2 – Le temps d'exécution Vs nombre de sauts et nombre de nœuds.

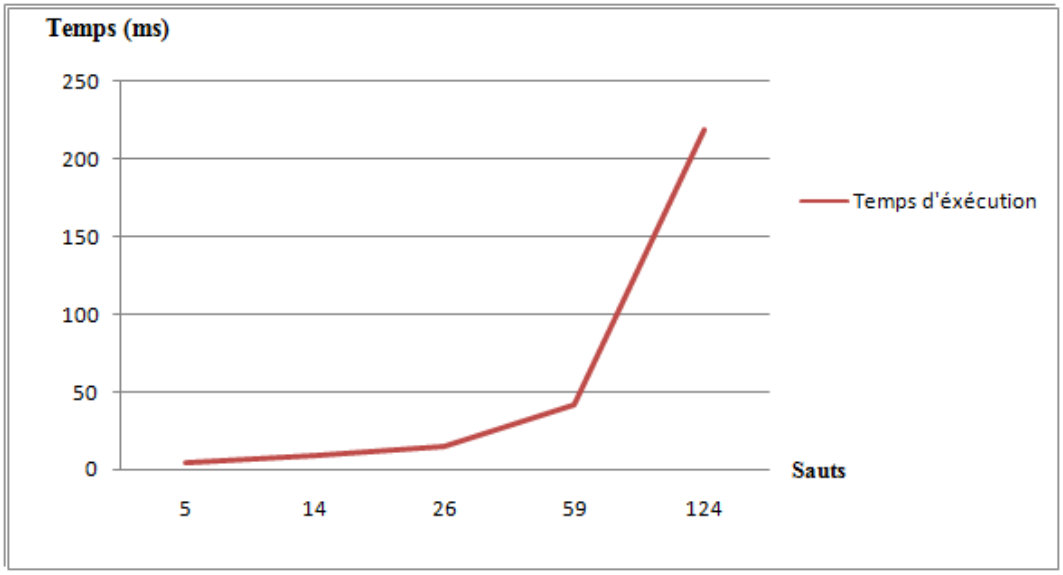

FIGURE 4.9 – Diagramme du temps d'exécution Vs nombre de sauts et du nombre de nœuds.

Interprétation

✓ Le diagramme de la *figure 4.9*, fait apparaitre un retard dans le temps d'exécution en dépassant un certain nombre de sauts (*60 sauts*), et cela est du probablement aux caractéristiques de la machine [3] utilisée.

4.7 Conclusion

Dans ce chapitre, nous avons décrit les étapes de la conception et de la réalisation d'une application qui simule la solution proposée dans le *chapitre III* pour le problème de recherche et de la localisation des services dans les réseaux P2P modélisés sous forme de graphes de Cayley. L'application issue nous permet d'effectuer une recherche et de palper le résultat et cela avec plusieurs graphes (*graphes avec différents nombres de nœuds*).
Nous avons décrit l'essentiel et le nécessaire requis pour sont utilisation ainsi que la majorité des résultats à obtenir suite à une simulation.

3. Testes réalisés avec une machine qui a : -3*Go* de RAM - Processeur AMD Turion(Tm)X2,DualCore Mobile,2.20 GHz

Conclusion générale et perspectives

Les réseaux peer-to-peer, longtemps délaisse, connaissent aujourd'hui un franc succès. Tout cela grâce au développement, l'accessibilité de l'Internet haut débit et grâce a l'approche décentralisée ainsi le vaste choix d'applications de partage de fichiers, proposés au grand public.
Le Peer-to-Peer ne doit être systématiquement associé aux applications de partage de fichiers connues du grand public. En effet, les entreprises s'intéressent, elles aussi, a ce type de réseaux, car il permet la mutualisation des ressources et le support d'un grand nombre d'utilisateurs.
Bien que le partage de fichiers soit le plus utilisé, les champs d'applications du peer-to-peer sont nombreux. Il s'étend à l'ensemble des domaines dans lesquels le groupe d'utilisateurs, la multiplication des ressources ou la puissance de calcul sont au centre des problématiques (*partage de connaissances, travail collaboratif*). Le modèle peer-to-peer est donc utilise pour construire des applications distribuées.

Actuellement les systèmes P2P soufrent de plusieurs problèmes tels que les fichiers truqués, virus et le problème de localiser l'emplacement des nœud qui permettent de stocker les profils des utilisateurs ainsi la localisation et la recherche des services (*ressources*).

Au cours de ce mémoire, nous avons traité le problème de localisation et de recherche de ressources à base des graphes de Cayley, et pour cela nous avons procédé comme suit : nous avons commencé par une présentation générale des réseaux P2P, puis on a fait une étude sur les graphes de Cayley a savoir leurs caractéristiques, leurs propriétés, ensuite nous avons proposé une nouvelle méthode recherche (*localisation*) et qui permet de trouver le plus court chemin entre deux nœuds du réseau, et enfin, nous avons terminé ce document par la réalisation d'une application pour cette nouvelle méthode en utilisant le langage de programmation *Java* sous l'environnement NetBeans IDE 7.1.2.
Comme perspectives, notre solution pourra être améliorée et complétée par :

- ✓ L'implémentation de la solution sur un réseau reel,
- ✓ La gestion des noeuds défaillants (*déconnectés*),
- ✓ Gestion des départs et arrivée des nœuds,
- ✓ L'implémentation de l'algorithme Optimal-Rout dans la même application afin de réaliser une comparaison entre les deux algorithmes.

Bibliographie

[1] AKZIZ Assia, BOUBKEUR Nawel, *" Problèmes de recherche des services dans les Réseaux P2P Cas : Graphe de Knödel ",* Mémoire master 2,Universite de Bejaia,2011.

[2] DIJOUX Alexandre, EMMA Samuel, *" Master Professionnel Système informatique et réseaux ",* Master professionnel Système informatique et réseaux, Université Claude Bernard LYON 1, 2006.

[3] GHARZOULI Mohamed, *" Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer ",* thèse doctorat, Université Mentouri Constantine,2011.

[4] HAFI Houda, *" Protocole pour la sécurité des réseaux sans fil peer to peer ",* thèse magistère, Université Kasdi Merbah - Ouargla,2010.

[5] AMGHAR Farouk, BENKERROU Makhlouf, *" Découverte et localisation des utilisateurs dans les réseaux P2P ",* Mémoire ingénieur, Université de Bejaia , 2009.

[6] MERABET Aimed, SEKHRIOU Noureddine, *" Problèmes de recherche des services dans les réseaux p2p ",* Mémoire ingénieur, Université de bejaia, juin 2010.

[7] GABRIELLE Feltin, GUILLAUME Doyen et OLIVIER Festor, *" Les protocoles Peer-to-Peer, leur utilisation et leur détection ",* LORIA - Campus Scientifique, 2003.

[8] TOUT Rabih, *" Sauvegarde des données dans les réseaux P2P ",* thèse doctorat, Université Claud Bernard Lyon1,2010.

[9] GUENOUNOU Farida, MAOUCHE Zineb, *" Problèmes de recherche et de localisation des services dans les réseaux P2P Cas : skip graphs ",* Mémoire d'ingénieur, Université Bejaia,2010.

Bibliographie

[10] DERBALI Mohammed ,EMBOUAZZA Fethi, *" Étude des réseaux peer to peer ",* mémoire master2, université paris 13, 2003.

[11] ABDELLI Nabila, AIT OUALI Kayssa, *" Decouverte et Localisation des Usagers dans un Reseau Peer To Peer ",* mémoire d'ingénieur, Université Béjaia, 2009.

[12] DILITRI Caron, DOMINIQUE Poure, *" l'empire du peer to peer ",* mémoire, Université paris dauphine, 2007.

[13] HUON Alexandre, JAAIDI Younes, MERCAT Philippe, PLAZE Aurélien, *" Les Réseaux Pair à Pair ",* Rapport de recherche, École nationale supérieure d'ingénieurs de CAEN, 2006.

[14] ABOUBACAR Thiam, KAARAR Hayat et DOUMA Wafaa, *" Adaptation des applications P2P dans les Réseaux Ad hoc ",* mémoire master2, université d'Avignon, 2008/2009.

[15] DELORME Charles, HEYDEMANN Marie-Claude, *" graphes de cayley ",*Rapport de recherche LRI 1186, Laboratoire de recherche en informatique, 2002.

[16] FRANCOIS Rousseau et LAURENT Tournier, *" Pércolation sur les graphes de Cayley ",* mémoire de maitrise (ENS)paris13,2005.

[17] Y. C. Tay, *"Paths to Stardom : Calibrating the Potential of a Peer-based Data Management System",* conférance internationale sur la gestion des données, School of Computing National University of Singapore, 2008.

[18] PREMKUMAR Vadapalli, PRADIP K Srimani, *"A New Family of Cayley Graph Interconnection Networks of Constant Degree Four",* Rapport technique CS-95-107, Université de l'État du Colorado, 1996.

[19] YVES Colin de Verdière, *" spectres de graphes ",* livre, Institut Fourier et Institut Universitaire de France, 1991. Mathematics Subject Classification. 05C10, 05C15, 05C50,35J10, 58G25.

[20] SICARD Jérome, *" Contributions à la Modélisation, évaluation et Conception de Systèmes de Recherche d'Information en Pair-à-Pair ",* thèse doctorat, Université d'Évry Val d'Essonne, 2010.

Bibliographie

[21] OLGA Lopez ACEVEDO, *" Marches quantiques généralisées pour l'algorithmique Quantique ",* thèse doctorat, Université de Cergy-Pontoise, 2005.

[22] *" UML2 Analyse et Conception,"* Éditions DUNOD, 2008.

[23] Ken Arnold, James Gosling et David Holmes, *"The Java Programming Language,* Fourth Edition, Addison-Wesley Professional, 2005, ISBN 0-321-34980-6".

[24] http ://netbeans.org/community/articles/interviews/yarda-tulach.html, dernier accès : 15/05/2013.

[25] http ://www.telecharger.org/windows/conception-graphique/logiciel-illustration/edraw-max.html, dernier accès : 19/06/2013.

Glossaire

Ad Hoc Ad Hoc est un réseau sans fil spontané, auto-organisé, ne reposant sur aucune infrastructure.

BitTorent BitTorent est un protocole de transfert de données pair à pair à travers un réseau informatique, été conçu en avril 2001et mis en place à l'été 2002 par le programmeur Bram Cohen.

Broadcast Broadcast Désigne une méthode de transmission de données à l'ensemble des machines d'un réseau.

CAN CAN est un réseau structuré où les clés sont associées à des coordonnées dans un espacede dimension d, et chaque pair est responsable d'une zone de clés.

Chord Chord est un protocole de réseau P2P structuré basé sur des tables de hachage.

Cluster Cluster est une grappe de serveurs (ferme de calcul) constituée de deux serveurs au minimum(nœuds) et partageant une baie de disques communs, pour assurer une continuité de service et/ou répartir la charge de calcul et/ou la charge réseau.

Freenet Freenet est un réseau informatique anonyme et décentralisé datant de 1996. Il offre la plupart des services actuels d'Internet (email, téléchargement, web...).

FastTrack FastTrack est le réseau Peer-to-Peer le plus utilisé grâce aux clients KaZaA.

JXTA JXTA est une plate-forme de développement d'applications P2P. Défini par une sériede protocoles conçus pour gérer n'importe quelle application P2P.

Gnutella Gnutella est un protocole P2P qui permet le transfert de fichiers.

Hash Hash est un algorithme de protection utilisé par Internet pour les signatures numériques, MD5 est l'algorithme hash le plus utilisé.

Morpheus Client grand public le plus utilisé du réseau Peer to Peer FastTrack. Il est principalement utilisé pour l'échange de fichiers pirates comme les MP3.

KAZAA KAZAA est un logiciel de partage de fichiers.

Napster Napster est un logiciel qui fonctionne sur une architecture P2P et qui permet le partage de fichiers (musique, vidéo) sur Internet. souvent considéré comme le premier réseau P2P.

Overlay Overlay est un réseau sous-jacent, construit au-dessus d'un réseau physique réel.

OceanStore OceanStore est un système qui permet de stocker des données de manière dispersée sur tout le réseau Internet.

PAST PAST est un outil facile à utiliser logiciel d'analyse de données gratuit.

SETI@home SETI@home est une plate forme construite sur une architecture P2P.

BOOCH une des méthodes d'analyse et de conception orientées objet à l'origine d'UML.

Liste des abréviations

ADSL	**A**symmetric **D**igital **S**ubscriber **L**ine
CPU	**C**entral **P**rocessing **U**nit
DNS	**D**istributed **D**omain **N**ame **S**ystem
HTTP	**H**ypertext **T**ransfer **P**rotocol
IP	**I**nernet **P**rotocol
P2P	**P**air à **P**air
PC	**P**ersonal **C**omputer
RPC	**R**emote **P**rocedure **C**all
TTL	**T**ime **T**o **L**ive
UML	**U**nified **M**odeling **L**anguage
OMT	**O**bject **M**odeling **T**echnique
OOSE	**O**bject **O**riented **S**oftware **E**ngineering EDI
CDDL	**C**ommon **D**evelopment and **D**istribution **L**icense
GPLv2	**C**ommon **D**evelopment and **D**istribution**L**icense
IDE	**I**ntegrated **D**evelopment **E**nvironmen
SPARC	**S**calable **P**rocessor **A**RChitecture
JDK	**J**ava **D**evelopment **K**it

Résumé

De nos jours, les réseaux pair â pair sont devenus très populaires en raison de leurs utilisation dans de nombreux domaines et applications en particulier dans le partage des fichiers entre les utilisateurs d'une manière décentralisée.
Cette décentralisation permet d'effectuer non plus une relation de client/serveur, dans laquelle chaque entité joue un rôle bien distinct, mais une relation d'égal à égal, dans laquelle le rôle de chaque entité est équivalent.

La recherche et la localisation des services dans les réseaux P2P est l'un des problèmes rencontré dans ce type de réseaux (*trouver l'emplacement des peers qui stockent les profile des utilisateurs*) ce qui a mené à l'apparition de nombreux graphes pour permettre l'efficacité de la recherche des services dans ces derniers.
Dans ce mémoire nous avons présenter le concept général des réseaux P2P, puis aborder les graphes de Cayley, ensuite proposer une nouvelle méthode de recherche de services dans un tel réseau en se basant sur ce type de graphes.
Mots clés :P2P, graphe de Cayley, accélération and optimisation.

Abstract

Today, the peer-to-peer networks has become very popular due to their use in many areas and applications especially in files sharing among users with a decentralized manner.
This decentralization makes possible a relationship of (*client/server*) architecture, where each entity plays a distinct role, but a relationship of equals, in which the role of each entity is equivalent.

Lookup and localization services in P2P networks is one of the problems encountered in this type of network (*find the location of peers that store profiles of users*) which lead to the appearance of many graphs to allow effective research services in the last.

In this paper we present the general concept of P2P networks and address the Cayley graphs, then we propose a new method of lookup services in a network based on Cayley graphs.
Key words :P2P, Cayley graph, accélération and optimisation.