الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire de Mila

Institut des Sciences et de la Technologie **Département de Mathématiques et Informatique**

Mémoire préparé En vue de l'obtention du diplôme de Master
En :
Filière : informatique général

Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Conception et réalisation d'une application de streaming peer to peer

Préparé par : Daas Abderrahmen
Belmehboul Fayçal

Soutenue devant le jury :

Président : Benchikhelhocin.M	MAA,	C.U.Mila.
Examinateur: kimouche.A	MAB,	C.U.Mila.
Promoteur : Ali.w	MAB,	C.U.Mila

Année universitaire : 2013/2014

Remerciements

اللهم صل وسلم على سيدنا محمد اشرف المرسلين وعلى اله وصحبه وسلم

الحمد لله الذي وفقنا فيما نحب و يرضاه و كتب لنا فيه الخير الكثير أسالك أن تنفعنا بهذا العلم وتجعله في ميزان حسناتنا

Si ce mémoire a pu voir le jour, c'est certainement grâce au soutien et l'aide de plusieurs personnes qui ont permis d'accomplir ce travail dans des conditions idéales. Nous profitons de cet espace pour les remercier tous.

Nous tiens à remercier notre encadreur ***Mme Ali Wided*** *à Centre universitaire de Mila pour son encadrement. Son expérience et ses qualités scientifiques ont toujours été sources d'enrichissement nous permettant de mener à bien ce travail.*

Et Nos vifs remerciements vont également aux membres du jury ***Mr Benchiekh elhoucine Madjid*** *Enseignant de l'Université Mila, et Mr* ***kimouche Abdelkader*** *Enseignant de l'Université Mila pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions. Aussi nos remercierons pour l'aide et l'assistance qu'ils nous ont apportés.*

Nous remercions les étudiants de la promotion 2013/2014 pour avoir été liés et unis tout au long de cette année et tous ceux qui ont collaborés de prés ou de loin à l'élaboration de ce travail. Qu'ils acceptent nos humbles remerciements.

Dédicace

Je tiens en tout premier lieu à remercier le dieu, je voudrais dédie ce modeste travail

À mes très chers parents à qui j`ai transmis mon stress et anxiété, pour leur affection, leur patience, leur soutien et leurs encouragements qui m`ont permis d'arriver au bout de ce travail.
À mes beaux-frères.
À mes chères sœurs.
À toute ma famille.
À mon binôme Abderrahmen que j`estime beaucoup.
À mes amis saber, faris, zaki, nassim, amine, redouane, achref, lyazid, moussa, dhayae
À Tous Mes Collègues d'étude et de travail
Je dédie ce modeste travail.

Fayçal

Dédicace

À mes très chers parents à qui j`ai transmis mon stress et anxiété, pour leur affection, leur patience, leur soutien et leurs encouragements qui m`ont permis d'arriver au bout de ce travail.

À mon frère et mes sœur que je les aime énormément.

À mon binôme Fayçal que j`estime beaucoup.

À toute ma famille, à tous mes amis surtout Saber, Faris, Ibrahim, Achref, Lyazid, Redoine, diaa elhaq, Amine, moussa, Nassim

À Tous Mes Collègues d'étude.

AbDeRrAhMeN

Table des matières

Chapitre I : les réseaux peer-to-peer

Chapitre II : le streaming peer-to-peer

Chapitre III : Conception

Chapitre IV : Techniques utilisées et implémentation

Liste des figures :

Introduction générale

Introduction générale

Un réseau est un ensemble de machines interconnectées qui permet l'échange d'informations et le partage des ressources physiques (unité centrale, mémoire de stockage, bande passante, imprimante, etc.), de fichiers, la diffusion d'informations et ainsi de suite. Ces réseaux ont été initialement fondés sur des architectures client-serveur.

Selon l'architecture client-serveur, il n'y a qu'une entité centrale plus puissante, le serveur, et plusieurs entités, la plupart du temps de puissances inférieures, les clients. Le serveur est l'unité d'enregistrement centrale aussi bien que le seul fournisseur de services aux clients. Un client ne demande que le contenu ou l'exécution des services, sans partager aucune de ses ressources propres. Avec ce modèle, le contrôle et la responsabilité de l'information diffusée dépendent du serveur.

Une telle architecture présente beaucoup d'inconvénients, l'existence d'un point central étant le plus évident. L'architecture P2P (peer-to-peer) se pose comme une solution de rechange à l'architecture client-serveur en offrant une répartition du trafic et de la charge, une résistance aux fautes et l'anonymat.

Les réseaux P2P offrent un moyen de décentraliser des services et des ressources autrefois centralisés sur un ou plusieurs serveurs. Par exemple, le calcul distribué (calculs complexes dans le domaine médical, spatial, cryptographique..), le partage de fichiers (données scientifiques, multimédia,..) et la collaboration (discussion, échange de données, édition simultanée de documents,..) sont le fruit du développement d'applications déployées sur des réseaux P2P visant à délocaliser les services.

D'un autre coté, dans le mode d'accès aux contenus partagés, la lecture d'un contenu n'est possible qu'après la fin de son téléchargement. Ce dernier implique un délai d'attente plus ou moins long en fonction du volume des données. Or, la consommation des contenus multimédias est marquée par le désir des utilisateurs d'y accéder instantanément. En effet, un temps d'attente important est plus acceptable si le contenu est un logiciel, mais il l'est moins d'un film ou d'une chanson. Toutefois, il existe un autre mode d'accès bien plus adapté à la transmission du contenu multimédia, à savoir la diffusion en continu, communément appelé streaming.

Le but de ce projet est la conception et la réalisation d'une application de streaming P2P. Une application qui permet de diffuser des média audio et vidéo entre les ordinateurs (dans notre application le streaming est représenté par l'appel vidéo) , L'application doit être en mesure de partager des fichiers entre plusieurs participants , l'échange de messages textuels ,…..etc.

Ce document est organisé en deux grandes parties théorique et pratique, chacune comporte deux chapitres :

Dans le premier chapitre nous présenterons les réseaux Peer to Peer en abordant leurs propriétés, leurs différentes topologies et leurs domaines d'applications,…etc. Nous terminerons par décrire le principe d'overlay (le réseau logique).

Le deuxième chapitre est consacré à l'étude du mode streaming audio/vidéo. Nous commencerons ce chapitre par une introduction avec des notions et définitions de ce mode. Nous aborderons également les différentes étapes du streaming, leurs champs d'application, les modes et les types de diffusion des contenus audiovisuels, ainsi que les Contraintes et les besoins pour le streaming.
Enfin de ce chapitre nous expliquerons le fonctionnement des protocoles de transmission en streaming.

Le troisième chapitre nous proposerons une conception et une description détaillée de notre prototype de l'application.

Le dernier chapitre est consacré à l'exposition des outils de développement, la réalisation et l'implémentation de l'application, de plus nous représenterons quelques interfaces graphiques de notre projet.

On finira par une conclusion sur le travail réalisé et les perspectives envisagées.

Chapitre 01 :

Les Réseaux Peer-to-Peer

1. Introduction :

Les réseaux peer-to-peer ont beaucoup évolué ces dernières années à tel point que les dernières générations de systèmes peer-à-peer fournissent à leurs usagers des applications de « télévision sur Internet » et de « Vidéo-à-la-demande ».

Que ce soit pour un usage grand public tel que les deux exemples cités précédemment ou bien pour des applications scientifiques afin de répartir des calculs ou stocker des données, aussi pour le partage des fichiers ,il est de nos jours de plus en plus courant d'avoir la nécessité de recourir à l'utilisation de réseaux peer-to-peer.

Dans ce chapitre, nous allons décrire les différents concepts des réseaux P2P (peer-to-peer). Nous y fournissons une définition, les propriétés de ces systèmes, Ensuite, nous allons détailler les différents niveaux de structuration de décentralisation et le degré de la structuration des réseaux P2P ainsi que leurs principales applications. Enfin, nous décrirons la notion d'overlay et ses différents types.

2. Définition :

Les réseaux peer-to-peer sont des systèmes distribués consistant de nœuds interconnectés, capables de s'auto-organiser au niveau de la topologie réseau afin de partager des ressources (contenu, cycles CPU, espace de stockage, bande passante), mais aussi capables de s'adapter aux pannes et de s'accommoder de la volatilité des nœuds, tout en maintenant une connectivité et des performances acceptables, sans avoir besoin de l'intermédiation ou du support d'un serveur centralisé global ou d'une quelconque autorité [1].

3. Notion de servent :

Contrairement au mode client/serveur classique où chaque entité à un rôle unique (i.e. client ou serveur), les peers d'un système P2P sont appelés des servent, un terme anglais spécifique créé à partir des mots SERVer et cliENT qui entend mettre en évidence le double rôle joué par les peers qui sont, suivant le sens de l'interaction, clients ou serveurs en même temps [2].

4. Propriétés

Les systèmes P2P ont des propriétés qui les rendent plus complexes que les systèmes classiques [3] :

- chaque peer participant joue tour à tour le rôle de client ou de serveur.
- pas de coordination centralisée.
- pas de base de données centralisée, l'information est disséminée parmi les nœuds (qui ne disposent que d'une vue locale/partielle du système).
- le comportement global émerge à partir des interactions locales.
- les peers sont autonomes.
- très disponibles (i.e., le système est disponible 7j/7, 24h/24).
- indépendants vis-à-vis de l'infrastructure physique.
- permettant de répartir la charge de manière efficace.
- robustes (i.e., résistants aux attaques et aux pannes).
- évolutifs (i.e., le nombre de participants peut augmenter sans diminuer les performances du système ; au contraire, plus il y a de participants, plus les systèmes P2P – du moins certaines architectures – sont efficaces).
- économiques car il n'y a pas nécessairement besoin d'investir dans de puissants serveurs. Ce sont les participants du système qui contribuent à l'amélioration des ressources du système.
- auto-organisés : si des nœuds tombent en panne, cela n'a pas d'impact significatif sur le système dans son ensemble, les systèmes P2P s'auto organisent et s'adaptent, c'est d'ailleurs une propriété indispensable pour des systèmes aussi dynamiques.

On peut aussi noter que les nœuds sont hétérogènes, en ce sens qu'ils n'ont pas tous nécessairement les mêmes caractéristiques. Par exemple, l'espace disque qu'ils peuvent partager peut être plus ou moins grand, de même que leur bande passante (montante et/ou descendante) . . .

5. Différents architectures P2P :

Comme nous l'avons indiqué dans la définition, les systèmes P2P sont principalement décentralisés, mais il existe différentes architectures possibles. Dans cette section, nous allons tenter de les passer en revue, de manière à illustrer les avantages et inconvénients de chaque approche.
La figure suivante illustre une taxonomie des systèmes informatiques présentée par B. Pourebrahimi [2].

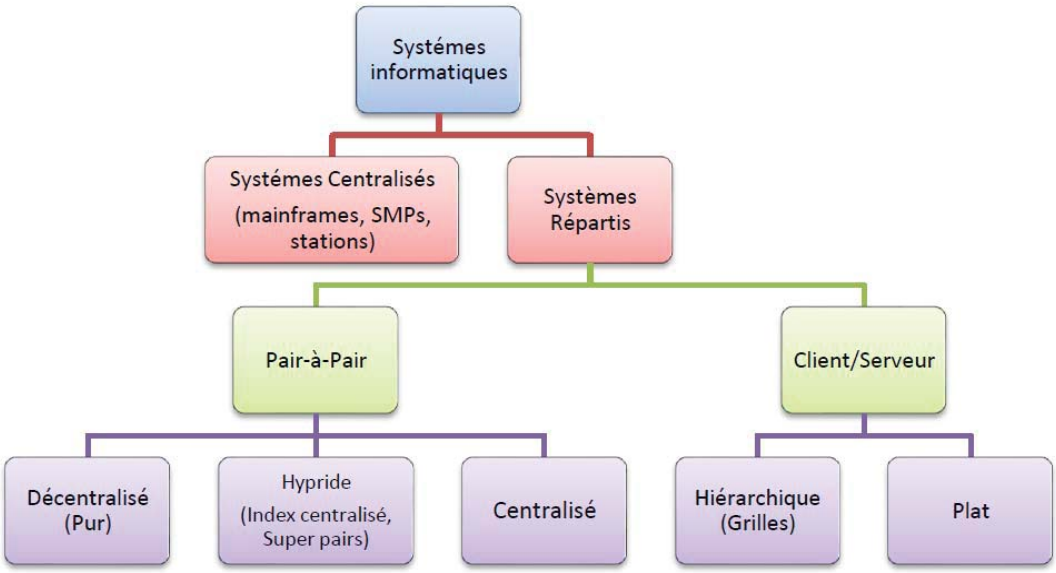

Figure I.1 : Taxonomie des systèmes informatiques

La classification des infrastructures P2P en fonction de leur degré de décentralisation scinde le modèle P2P en trois sous modèles, qui sont les modèles pur, hybride et centralisé. La Figure 1.2 représente un exemple de topologie de chacun de ces trois sous-modèles.

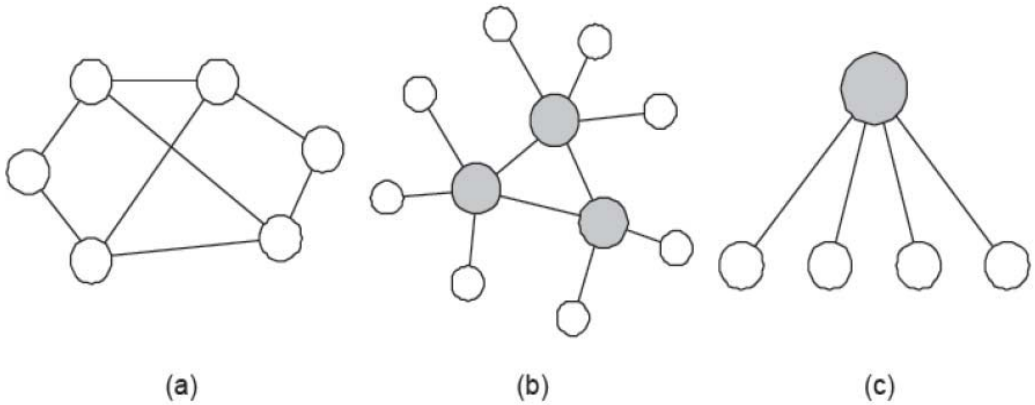

Figure I.2 : Exemple de topologies P2P qui présentent différents niveaux de décentralisation.

(a) Topologie construite selon le modèle pur.

(b) Topologie construite selon le modèle hybride.
(c) Topologie construite selon le modèle centralisé.

5.1. Le modèle centralisé :

Le modèle centralisé, dont un exemple de topologie est représenté sur la Figure I.2.c, est à la limite du modèle P2P car il repose sur un serveur dédié qui centralise et maintient une connaissance globale sur l'état du système, les ressources étant toujours hébergées sur les peers. Une fois les opérations de découverte et de localisation sont effectuées sur le serveur central, les peers peuvent interagir directement entre eux. Ainsi, cette interaction différencie le modèle P2P centralisé du modèle client-serveur.

Cependant, la seule limite de ce modèle étant la puissance de calcul ou la bande passante du serveur, bien qu'il n'intervient pas dans les transferts de fichier, le serveur est nécessaire lors de chaque insertion de nœud ou de contenu, de chaque recherche, ainsi que le téléchargement pour actualiser les sources. Son dimensionnement et sa disponibilité sont donc critiques pour le fonctionnement du système tout entier.

Du point de vue de la sécurité, la présence d'un serveur représentant une autorité peut résoudre beaucoup de problèmes. Chaque nœud peut facilement recevoir un certificat signé lui permettant de communiquer de manière sûre avec les autres peers. La sécurité du serveur peut être assurée par des mécanismes classiques d'authentification, de contrôle d'accès et de disponibilité. En revanche, il présente le problème du point central de panne ou SPoF (Single Point of Failure).

De plus, cette centralisation implique une autorité, qui peut être vue comme contre les principes mêmes du peer-to-peer. Ce modèle a été utilisé dans des applications telles que Napster. [4]

5.2.Le modèle hybride :

Le modèle hybride, dont un exemple de topologie est représenté sur la Figure I.2.b, ajoute un degré de hiérarchie au modèle pur.

Les peers particuliers utilisés dans le modèle hybride sont généralement appelés des super-nœuds. Le rôle qu'ils jouent n'est pas figé et varie d'une infrastructure à une autre. En général, ils sont utilisés pour assurer des fonctions relatives au routage, à la comptabilité ou à l'organisation fonctionnelle des peers. Par exemple Jxta l'utilise pour la découverte et la localisation de ressources. Un autre exemple est celui de Skype [5] qui offrent des solutions de communication et de voix sur IP (Internet Protocol). Les nœuds ordinaires sont des simples clients Skype, qui effectuent, des appels et envoient des messages texte. Tout nœud avec une adresse IP publique, ayant suffisamment de CPU, mémoire et de bande passante est un super nœud potentiel. Chaque nœud se connecte à un super-peer via un serveur d'authentification et seuls les supers-peers

communiquent par inondation. Ensuite, les communications sont réalisées directement entre les peers concernés.

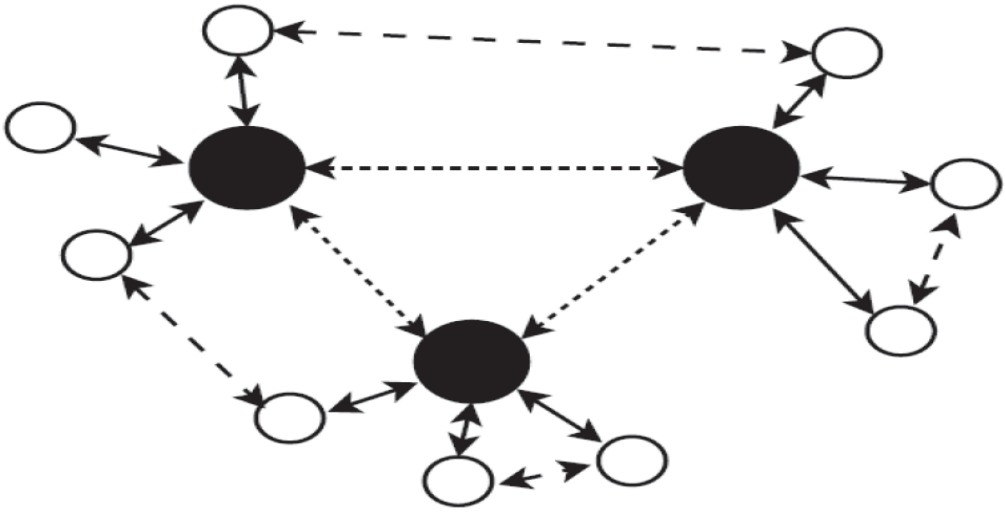

Figure I.3 : Topologie d'un réseau P2P hybride.

Cette organisation permet de réaliser des requêtes aussi complexes que dans Napster [4], tout en évitant les faiblesses de passage à l'échelle de la centralisation. Néanmoins, le choix des super-peers n'est pas facile, puisque pour un réseau fonctionnant de manière optimale ces super-peers doivent être assez puissants et bien connectés.

5.3.Le modèle pur :

Dans [7], Schollmeier propose une définition du modèle pur qui le caractérise par le fait que, dans une communauté de peers, la suppression de n'importe quel peer présent n'affecte pas le service offert.

Un exemple de topologie construite selon le modèle pur est représenté sur la Figure I.3.a où l'on voit qu'aucun élément ne joue de rôle particulier. Actuellement, de nombreuses applications P2P sont construites selon ce modèle. On trouve par exemple Gnutella, tel qu'il était déployé dans sa première version, et toutes les infrastructures à base de tables de hachages distribuées telles que Chord, CAN.

6. Champs d'application des réseaux P2P :

Les architectures peer to peer sont utilisées dans plusieurs catégories d'applications qui peuvent être classées en cinq grandes classes comme le montre la figure suivante :

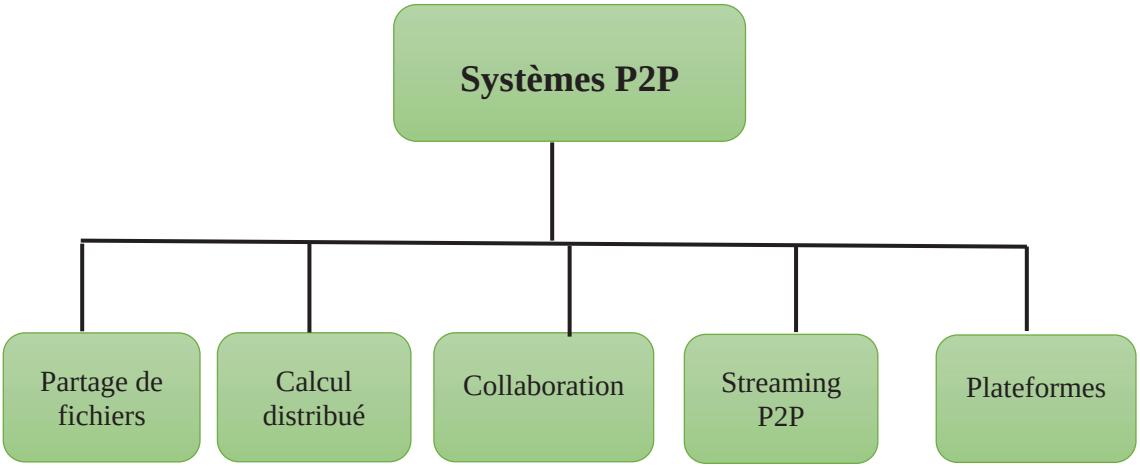

Figure I.4 : les applications de Peer to Peer.

6.1.Partage de fichiers :

Le partage de fichiers est l'application la plus courante et la plus populaire des systèmes P2P. En effet, ce sont chaque jour des millions d'internautes qui utilisent des applications P2P comme BitTorrent, eMule, Kazaa … L'idée est de mettre en commun la bande passante et l'espace disque des participants afin de permettre l'échange de fichiers.

Chaque peer met à disposition les fichiers qu'il désire et peut télécharger ceux partagés par les autres [2].

6.2. Le calcul distribué :

Cette catégorie inclut les systèmes qui ont pour but d'utiliser la puissance de calcul (cycles CPU) disponible sur un grand nombre d'ordinateurs. L'idée est d'utiliser les machines oisives d'un réseau pour effectuer des parties d'un calcul découpé en plusieurs unités indépendantes et ré-assemblables. Ces unités sont distribuées sur les différents peers qui en font leurs tâches et retournent les résultats. Un serveur central est souvent nécessaire pour découper et distribuer les tâches puis collecter les résultats des peers pour effectuer le calcul final [6].

6.3. Travail collaboratif

Ce type d'applications permet à un groupe d'utilisateurs de collaborer en temps réel sans utilisation d'un serveur central.

Dérivés des systèmes « groupware » utilisés au niveau des entreprises, ce type d'applications permet à un ensemble d'utilisateurs de travailler de manière commune sur un projet distribué [2].

6.4. Streaming P2P

Le streaming P2P est le fait de regarder en direct des flux produits et/ou relayes par d'autres peers 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 [7].

6.5. Les plateformes :

Une analyse des systèmes peer-to-peer existants permet de constater que les réseaux partagent un grand nombre de fonctionnalités communes. Malheureusement, ces fonctionnalités sont souvent volontairement implémentées par des mécanismes propriétaires et la plupart des logiciels peer to peer ont été développés de manière spécifique sans références à des standards propres au peer to peer. Les plateformes proposent une infrastructure générique pour développer des applications peer to peer en offrant les fonctions de base tel que la gestion de peer, l'attribution d'identifiants, la découverte des ressources, la communication entre peers, la sécurité… ainsi Sun propose sa plateforme JXTA basée sur la technologie Java et Microsoft intègre dans .NET des outils pour développer des applications peer-to-peer.

7. Overlays

Les peers participants à un système P2P forment souvent un réseau virtuel, appelé overlay [8] construit au-dessus de réseau physique. Un exemple d'overlay est représenté sur la Figure I.5. Généralement, un tel réseau virtuel présente une propriété de transparence qui s'applique à plusieurs points. Tout d'abord, il permet de faire abstraction des différences de nature des peers. Un réseau peut être composée de peers avec des caractéristiques diffèrent sur les plans :

a. matériel, avec par exemple des stations de travail, des téléphones mobiles ou un cluster de machines.

b. logiciel, au niveau des systèmes d'exploitation et langages de programmation.

c. communication, avec l'utilisation de technologies différentes.

La transparence s'applique aussi sur le routage effectué au niveau sous-jacent : deux voisins de la topologie virtuelle ne le sont pas forcément physiquement : ils peuvent être situés dans des espaces physiques et sur des réseaux différents (exemple, les peers A et B présentés dans la Figure I.5). L'overlay rend ainsi transparent le routage effectué au niveau physique.

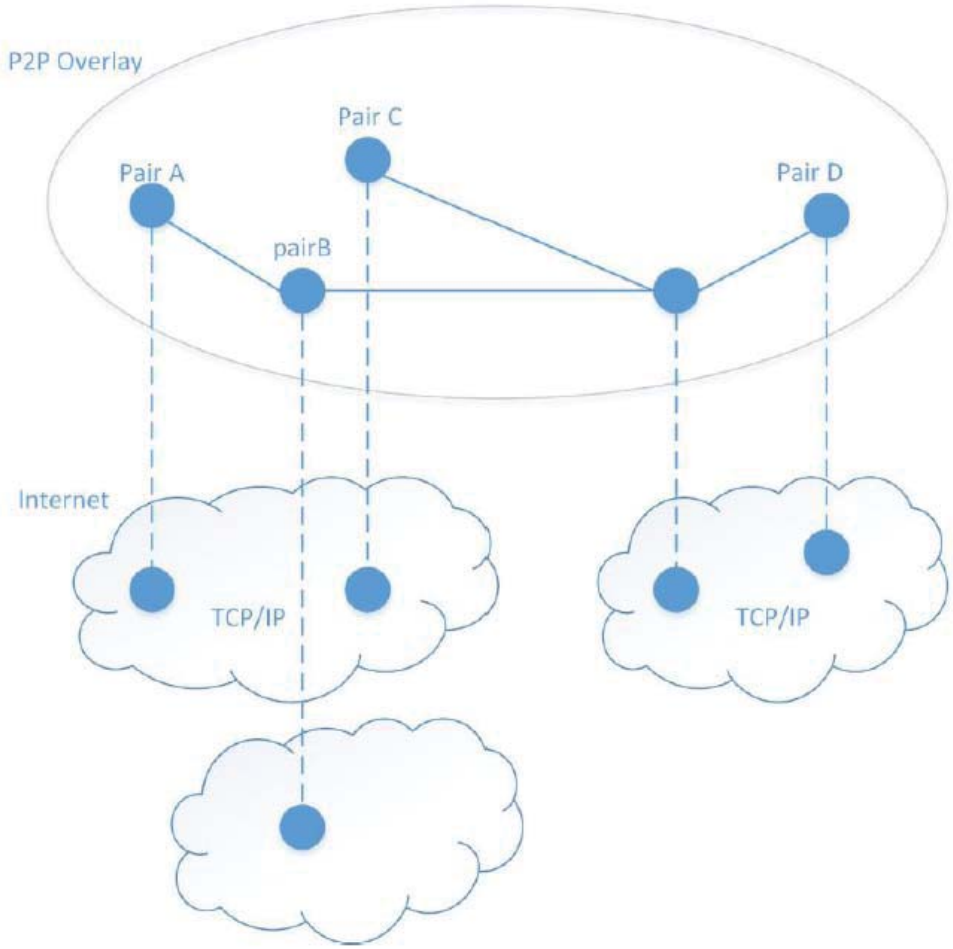

Figure I.5 : P2P overlay au-dessus d'une infrastructure d'Internet

La topologie de l'overlay (la manière dont les nœuds sont liés les uns aux autres dans l'overlay) est différente selon que l'overlay est :

– non structuré

– structuré

Il n'y a pas de « meilleur » type d'overlay, chaque approche a ses avantages et ses inconvénients. Le choix dépend de plusieurs facteurs, dont le type d'application envisagée, la taille du réseau, son dynamisme, . . .

7.1. Non structurés

Dans les réseaux Peer-à-Peer non structurés des liens aléatoires sont générés entre les nœuds. Pour s'insérer dans le réseau, chaque nouveau nœud doit connaître un nœud appartenant à ce réseau, qui lui sert d'amorçage ou de bootstrap. Les requêtes se passent ensuite sous la forme d'inondation (demande à tous les voisins qui demandent à tous leurs voisins. . .) ou de marche aléatoire (demande à un voisin qui demande à un voisin. . .).

Dans ce type de systèmes P2P la Clé est une chaîne de caractères, appartenant à un espace de noms de taille arbitraire. D'habitude, les clés contiennent la description de la ressource qu'elles identifient, exemple : dans des applications

de partage des fichiers comme Gnutella, chaque clé représente le nom d'un fichier.

Lorsqu'un peer cherche une ressource, il envoie une requête qui contient des informations qui doivent correspondre à la clé identifiant de la ressource. La requête est envoyée à un certain nombre de voisins (peut-être à tous), lesquels cherchent la clé correspondante parmi les ressources qu'ils stockent. Ces peers renvoient à leur tour la requête originelle à ses propres voisins, pour que la recherche atteigne tous les peers. Les clés correspondantes sont renvoyées au peer à l'origine de la requête comme résultat de la recherche.

Les paramètres importants pour ce type de réseau sont les degrés entrant et sortant (nombre de connexions) de chaque nœud, car ils conditionnent directement la connectivité et la robustesse du réseau. L'autre paramètre important est le paramètre de requête. Dans le cas d'une inondation, une valeur trop forte génèrera un fort trafic inutile alors qu'une valeur trop faible ne retournera pas forcement les réponses souhaitées.

Dans le cas d'une démarche aléatoire, il s'agit de déterminer les voisins qui recevront la requête : si le voisin est mal choisi, alors le résultat de la requête ne sera pas intéressant. Gnutella est le premier réseau non structuré proposé. A partir de ce dernier, Gia est proposé pour améliorer le passage à l'échelle de Gnutella.

7.2.Structurés :

Les réseaux structurés apportent un élément supplémentaire, ils établissent un lien entre les ressources partagées (e.g. fichiers) et l'endroit où les trouver (i.e., les nœuds qui les partagent). Grâce à cet élément supplémentaire, les overlays structurés apportent une solution au problème de l'indexation et de la recherche dans les systèmes P2P décentralisés. Partielle seulement, car ils ne permettent pas de faire des recherches sur base de mots clés, mais uniquement de retrouver des ressources sur base de leur identifiant.

Néanmoins et c'est un avantage non négligeable, cette localisation est déterministe, si la ressource est partagée, on est sûrs de la trouver en utilisant un nombre borné de messages. Le nombre de messages utilisés pour trouver une ressource donnée dans un overlay structuré est de l'ordre de O(log n), n est le nombre de nœuds présents dans l'overlay [9] (ce nombre dépend de l'organisation de l'overlay).

Pour arriver à ce résultat, les overlays structurés doivent maintenir une certaine structure au niveau de l'organisation des nœuds dans le réseau (i.e., une certaine topologie). Un des problèmes est qu'il est difficile/coûteux de maintenir cette structure quand le système évolue rapidement (i.e., quand la composition du réseau évolue rapidement à cause d'arrivées/départs rapides des nœuds et/ou des ressources [1]). Le protocole Chord est un exemple de tel système. Il repose sur une topologie en anneau.

8. Conclusion :

Dans ce chapitre, nous avons parlé sur les réseaux P2P. Nous avons commencé par présenter la définition des réseaux p2p et leurs propriétés. Ensuite nous nous, sommes attardés sur les différentes architectures existantes et quelques exemples de champs d'applications. Finalement, nous avons tenté de présenter la notion d'overlay et ses différents types.

Le but était donc dans ce chapitre de fournir une idée claire sur les réseaux P2P. Maintenant nous allons pouvoir aborder le chapitre qui concerne le streaming et spécialement le P2P streaming.

Chapitre 02 :

Streaming Peer to Peer

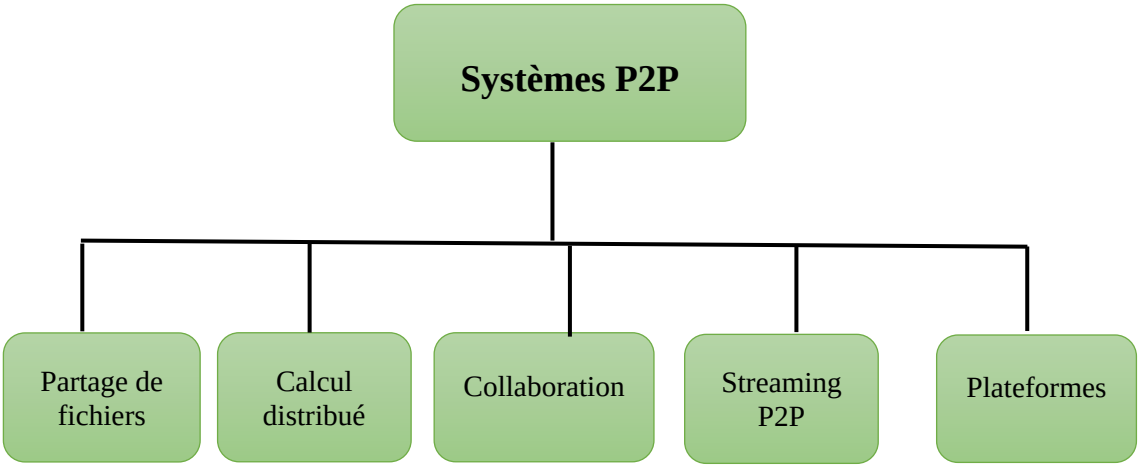

Figure I.4 : domaines d'applications de Peer to Peer.

6.1.Partage de fichiers :

Le partage de fichiers est l'application la plus courante et la plus populaire des systèmes P2P. En effet, ce sont chaque jour des millions d'internautes qui utilisent des applications P2P comme BitTorrent, eMule, Kazaa … L'idée est de mettre en commun la bande passante et l'espace disque des participants afin de permettre l'échange de fichiers.

Chaque peer met à disposition les fichiers qu'il désire et peut télécharger ceux partagés par les autres [2].

6.2. Le calcul distribué :

Cette catégorie inclut les systèmes qui ont pour but d'utiliser la puissance de calcul (cycles CPU) disponible sur un grand nombre d'ordinateurs. L'idée est d'utiliser les machines inactifs d'un réseau pour effectuer des parties d'un calcul découpé en plusieurs unités indépendantes et ré-assemblables. Ces unités sont distribuées sur les différents peers qui en font leurs tâches et retournent les résultats. Un serveur central est souvent nécessaire pour découper et distribuer les tâches puis collecter les résultats des peers pour effectuer le calcul final [6].

6.3. Travail collaboratif

Ce type d'applications permet à un groupe d'utilisateurs de collaborer en temps réel sans utilisation d'un serveur central.

Dérivés des systèmes « groupware » utilisés au niveau des entreprises, ce type d'applications permet à un ensemble d'utilisateurs de travailler de manière commune sur un projet distribué [2].

Pour pouvoir réaliser cette opération, le serveur envoie le fichier vidéo ou audio par paquets de données qui seront traités par l'ordinateur de l'utilisateur au fur et à mesure de leurs arrivées. A cause des fluctuations réseaux les paquets n'arrivent pas toujours dans le bon ordre. On utilise donc une mémoire tampon pour regrouper les paquets dans le bon ordre. Cette mémoire tampon ou buffer est créé par le lecteur média de l'ordinateur de l'utilisateur. Au bout de quelques secondes, une fois que le buffer de réception possède assez d'informations, la lecture du flux commence et les images ou le son sont retransmis. La mémoire tampon a donc pour rôle de fluidifier le flux. Si la connexion réseau est mauvaise l'arrivé des paquets sera ralentie. Lorsque le buffer de réception est vide, la lecture s'arrête et reprendra lorsqu'elle possèdera assez de données pour continuer. L'image est alors figée.

Pour que le streaming soit rendu possible il faut effectuer un traitement sur le fichier avant de le transmettre sur le réseau. En effet la taille d'une vidéo étant en général assez importante, il faut la compresser afin de réduire le nombre de paquets à envoyer.

Il ne faut donc pas attendre du streaming la même qualité que le fichier d'origine. En plus de la compression, on peut appliquer au fichier certains filtres ou encore le convertir dans un autre format.

3. Téléchargement versus streaming:

Les données dynamiques peuvent être affichées à partir d'une machine locale ou d'une machine distante. Dans le premier cas, les données sont entièrement disponibles localement, stockées sur un disque dur ou disponible sur un DVD par exemple. Lorsqu'un spectateur demande la restitution d'un contenu audiovisuel, les données sont récupérées depuis l'espace de stockage local, décodées, puis passées à la carte graphique et/ou carte son qui se charge(nt) de les lui restituer.

Lorsqu'un spectateur demande la restitution d'un contenu audiovisuel se trouvant sur une machine distante, les données sont récupérées depuis l'espace de stockage de cette machine, puis sont transmises à la machine de l'utilisateur par l'intermédiaire d'un réseau. Trois modèles de transmission sont possibles ; téléchargement simple, streaming et téléchargement progressif.

Dans le modèle de téléchargement simple, les données sont entièrement téléchargées d'une machine distante vers un espace de stockage local avant de commencer leur décodage puis leur restitution (voir figure II.1). Le transfert de données peut se faire à l'aide d'un serveur http ou FTP standard [12].

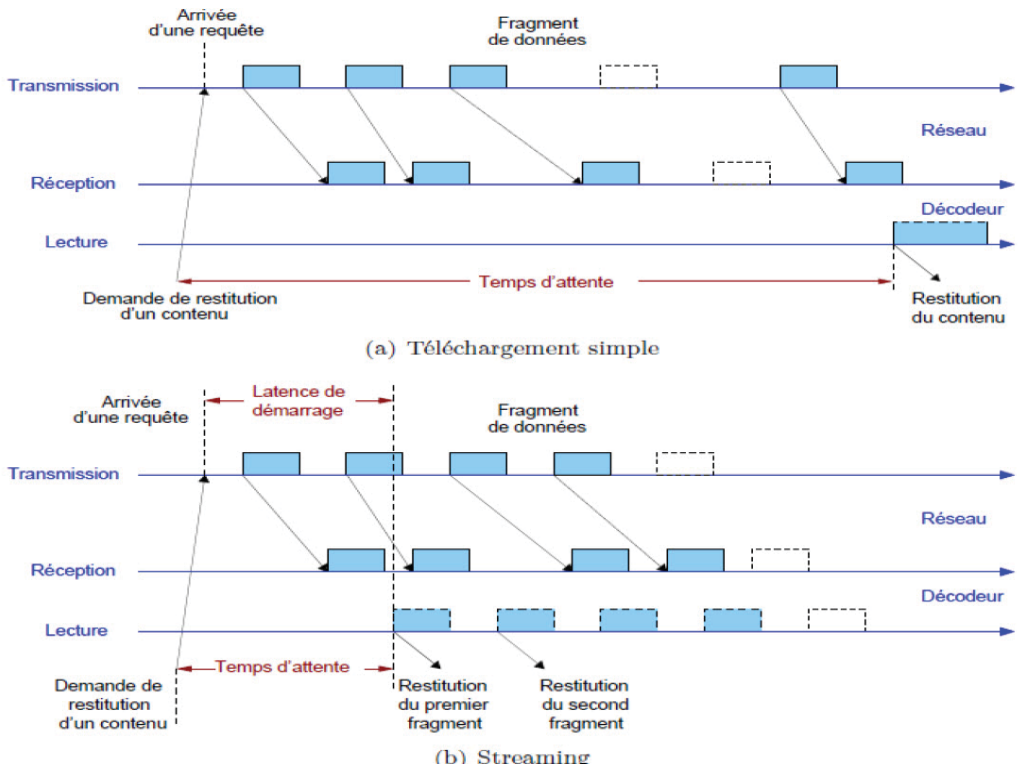

Figure II.1 Téléchargement versus streaming

❖ Téléchargement progressif [11]

Téléchargement progressif ou pseudo streaming, permet à l'utilisateur de lire un média alors que le fichier est encore en cours de téléchargement. Il se différencie du téléchargement " Normal" pour lequel l'utilisateur doit attendre d'avoir complètement téléchargés les fichiers pour pouvoir ensuite le lire depuis son disque dur. Avec le pseudo streaming, une copie du fichier est conservée sur le disque de l'utilisateur, ce qui lui permettra de relire le media ultérieurement. Le téléchargement progressif est également appelé streaming HTTP car les logiciels serveur Web utilisant des protocoles standard (serveurs HTTP) peuvent transmettre des fichiers à téléchargement progressif.

4. Les phases de streaming :

Le streaming est le traitement appliqué à un flux de données en temps réel transitant sur le serveur ou à un montage vidéo ou à un fichier audio installés sur le serveur. Il procède en plusieurs étapes :

4.1. Encodage :

Afin de réduire le nombre de paquets de données à transmettre (et donc économiser la bande passante nécessaire) et permettre leur lecture en temps réel, les fichiers multimédias doivent être compressés dans un format de streaming du serveur : c'est l'encodage.

Cela suppose deux opérations : la compression à l'aide d'un codec et le multiplexage dans un conteneur (muxeur).

- **Compression avec un codec :**

Un codec (pour "**CO**deur/**DEC**odeur") est un algorithme de compression utilisé pour réduire la taille d'un flux ou d'un fichier (audio ou vidéo). Il génère un format de compression spécifique. Il y a des codecs audio et des codecs vidéo.

- **Multiplexage dans un conteneur :**

Le multiplexage consiste à encapsuler (empaqueter) ensemble les différents flux requis dans un même fichier (conteneur) avant que celui-ci ne soit diffusé sur le réseau.

Un conteneur (ou muxeur) contient donc un ou plusieurs flux déjà encodés à l'aide de codecs, mais aussi d'autres informations qui définissent comment lire les données en indiquant le nom du codec nécessaire au décodage des flux audio et vidéo.

- ✓ **WAV, AIFF**,...: sont des conteneurs audio (flux audio seulement).
- ✓ **AVI, OGG, QuickTime, ASF, MP4**,...: sont des conteneurs vidéo (flux audio et vidéo).

4.2. Diffusion des données sur le réseau (Buffering) :

Le fichier audio ou le conteneur vidéo est ensuite placé sur le serveur qui, à chaque requête d'un internaute, duplique le fichier demandé et le délivre sous la forme d'un flux continu de données (petits paquets de données marqués temporellement afin d'être réordonnancés de manière cohérente par le client).

4.3.Lecture du média :

Les différentes opérations permettant la lecture du média sont assurées par le lecteur multimédia de l'utilisateur. Tout lecteur est libre (gratuit) et multiplateforme. Il dispose de plusieurs codecs à sa disposition.

- **Démultiplexage :**

Le conteneur est tout d'abord démuxé (c'est le démultiplexage) : les différents flux audio et vidéo sont séparés et sauvegardés dans des fichiers différents.

- **Décompression :**

Chacun de ces flux sont ensuite décodés (décompressés) en temps réel avec les mêmes codecs que ceux utilisés pour compresser ces flux. Le lecteur multimédia peut être amené à devoir télécharger le codec nécessaire si celui-ci n'est pas présent dans l'environnement et s'il est autorisé à le faire. Ces flux sont alors restitués avec le maximum de qualité possible à l'utilisateur. Dans la plupart des cas, la compression est asymétrique : la compression (en fonction de la qualité que l'on souhaite avoir) sera plus longue et la décompression assez rapide pour permettre une lecture presque instantanée du flux.

5. Types de communication en streaming :

5.1. Broadcast (one to all) :

Type de communication caractérisé par le nombre important de récepteurs. Le retour (feedback) d'un récepteur vers l'émetteur est généralement impossible, ceci limite la capacité du système à s'adapter. Il est utilisé dans le domaine de l'audio-visuel.

5.2. Unicast (point à point ou one to one):

Cette technique consiste, pour le serveur, à associer et à envoyer un (ou plusieurs) flux à chaque utilisateur (connexion point à point). L'inconvénient de cette technique est qu'elle exige beaucoup de bande passante côté serveur, et peut engendrer une saturation si le nombre de clients est trop important. En contre partie, cette méthode autorise plus de souplesse au client : il peut choisir son débit, changé facilement de position dans la lecture du fichier, etc.

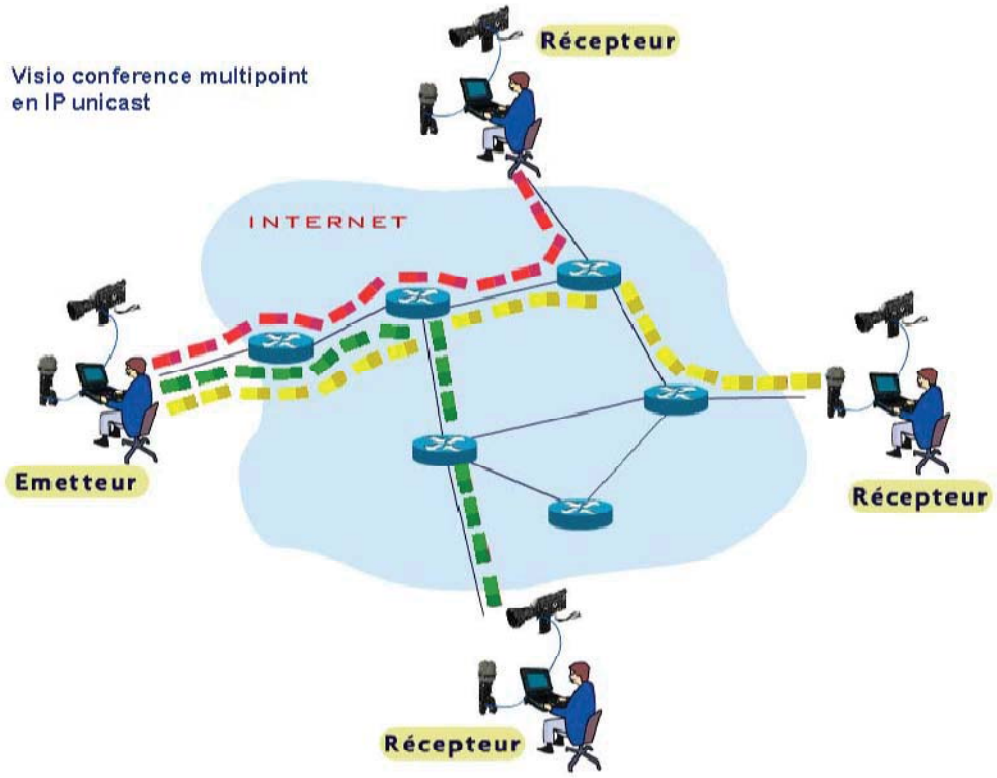

Figure II.2 : Diffusion Unicast

5.3. Multicast (point à multipoint ou one to many):

C'est une technique point à multipoint entre serveur et clients. Le serveur envoie un flux unique pour un groupe d'utilisateurs, flux qui est ensuite distribué via des routeurs multicast (sans duplication inutile). Cela permet de réduire le trafic au niveau du serveur de streaming, mais nécessite d'avoir des routeurs sur le réseau qui gèrent le multicast, ce qui n'est pas toujours le cas sur Internet.

Cette technique est plus contraignante pour l'utilisateur car il ne peut pas contrôler le flux qu'il reçoit (il peut juste s'inscrire/se désinscrire du groupe multicast). Le multicast est souvent utilisé pour la diffusion en direct ou la multi-conférence.

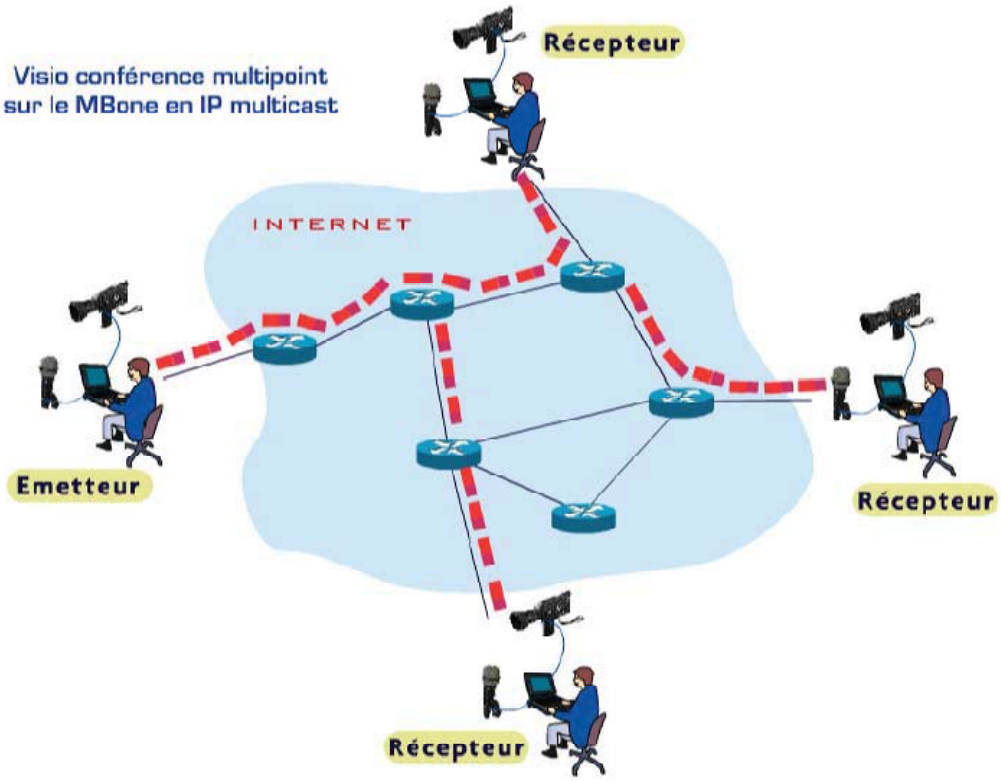

Figure II.3 : Diffusion Multicast

6. Les domaines d'applications de streaming :

Partout où la communication électronique est employée, les demandes de streaming sont sans fin. Streaming peut être délivré comme paquet complet de vidéo de la programmation linéaire, comme service d'abonnement, ou comme PPV (Pay-Per-View). Il peut faire partie d'un site Web interactif.
Parmi ces applications on trouve:

6.1. Streaming vidéo à la demande :

Le streaming vidéo à la demande (VoD : Vidéo on Demand) permet la distribution des streams (flux) audio et vidéo à travers des LANs, intranet et internet après une demande du client. Le client entre en contact avec le serveur vidéo et demande le Stream approprié. Beaucoup d'applications de VoD sont disponibles comme Microsoft Netshow, RealVideo pour les réseaux de transmission en temps réel.

6.2.La vidéoconférence :

La vidéoconférence est une application qui utilise la vidéo pour faire des réunions interactives entre groupe d'individus distants. La vidéo est fournie à tous les participants en temps réel. La vidéoconférence exige des installations chères et des coûts de transmission élevés en termes de la consommation de la bande passante.

6.3.Streaming peer-to-peer vidéo :

Le concept principal du calcul peer-to-peer est que chaque peer est un client et serveur en même temps. Dans ce contexte, le contenu multimédia joué par l'utilisateur est mis en commun entre des peers. Le partage peer emploie le mode ouvrir après le téléchargement, mais dans le cas du streaming peer-to-peer vidéo; on lit les données pendent le téléchargement. Un des avantages de streaming peer-to-peer de vidéo est que les peers ont la liaison directe à d'autres peers évitant la communication par l'intermédiaire des serveurs de médiation.

7. Contraintes et besoins pour le streaming audiovisuel [14]

Offrir un service de streaming sur l'Internet présente plusieurs défis: respecter des contraintes de délai et de qualité imposées par la nature des données audiovisuelles, en utilisant un réseau comme Internet qui assure un service dit « best effort », c'est-à-dire assurant à l'utilisateur un acheminement de paquets sans garantie de fiabilité ou de délai. Les contraintes les plus connues seront détaillées dans la suite de cette section.

7.1. Contraintes de débit

En règle générale, la qualité des flux multimédia s'améliore avec l'augmentation du débit du flux. Cependant, au-dessus d'une certaine limite de débit Rmax, l'augmentation de qualité est négligeable. De la même façon, réduire la bande passante en dessous d'une certaine limite Rmin rend les données inutilisables par les récepteurs.

7.2. Contraintes de transmission

La transmission de flux multimédia est régie par différentes contraintes de délais et de robustesse. Ainsi, pour des raisons de délais, les flux audio seront transmis aussitôt après la génération de la trame, provoquant la transmission de paquets de petites tailles. De la même façon, il est nécessaire de mettre en œuvre des politiques intelligentes de paquettisation lors de la communication vidéo afin de rendre le flux plus robuste aux pertes de paquets. Ces règles de paquettisation conduisent à l'utilisation de paquets de tailles variables dépendantes à la fois du débit de la source mais également de la nature de la vidéo transmis.

7.3. Contraintes d'adaptation

Suivant le schéma de compression et le type de données utilisés, l'adaptation du débit de la source aux contraintes du réseau ne peut se faire qu'avec une certaine granularité. Ainsi, pour une source vidéo classique la granularité est fonction de la stratégie de réduction de fréquence temporelle (i.e. nombre d'images par seconde). On parle également de granularité d'adaptation temporelle.

7.4. Contraintes de délais

Les flux multimédia ont des contraintes de délais que l'on peut qualifier de temps-réel. Pour des applications interactives telles que la visiophonie, il est impératif que le délai entre la capture de l'image et l'instant où le récepteur visionne l'image soit le plus court possible. Il est impératif également, et ceci est aussi vrai pour les applications de streaming, d'assurer une continuité dans la présentation des médias au récepteur. Cependant, le débit disponible de bout-en-bout pour le fonctionnement du service peut varier considérablement et à tout moment. Pour régulariser et absorber les variations du débit, la stratégie employée consiste à utiliser des mémoires tampons. Néanmoins, si un certain seuil est dépassé (i.e. si le délai de bout-en-bout dépasse 300ms), ceci peut rendre les applications de streaming non conversationnel.

7.5. Contraintes de Qualité de Service stable

L'adaptation fréquente du débit de la source à des fins de réactivité face aux phénomènes de congestion peut mener à de fortes variations de débit pour la source. Ces fortes variations ne sont évidemment pas compatibles avec les besoins de QoS des applications multimédia. Lors d'une transmission vidéo, par exemple, les clients sont très sensibles aux brusques différences de qualité. Afin, de permettre aux récepteurs d'avoir une qualité plus stable, il est nécessaire de fixer des seuils de variations de qualité acceptables et de s'y conformer.

7.6. Contraintes liées aux pertes de paquets

Au cours d’une session de streaming vidéo, l’impact visuel engendré par la perte d’un paquet doit être limité pour fournir une qualité de service acceptable. En pratique, les règles de mise en paquets doivent permettre à l’encodeur de faire face aux pertes de paquets. En considérant que le protocole de transport est assez robuste pour maintenir la synchronisation au décodeur. Toutefois la notion de tolérance aux pertes est très dépendante de l'application et des utilisateurs de l'application. Ainsi, le streaming d'un morceau de musique sera très peu tolérant alors qu'une tolérance supérieure est acceptée pour des applications de visiophonie.

7.7. Contraintes de volume des données

A l'inverse des données textuelles, les données audiovisuelles digitalisées présentent des volumes très importants. Des techniques de compression vidéo (H.264 AVC [15], SVC [16]) ont donc été développées afin de réduire ce volume. Néanmoins, même après compression, les données vidéo demeurent volumineuses.

8. Les protocoles de transmission en streaming :

Les flux de données audio ou vidéo sont envoyés à partir d'un serveur de streamer média à un client qui utilise souvent un protocole de transmission en temps réel appelé RTP (Real time Transport Protocol).

8.1. Le protocole RTP

Le protocole RTP (Real-Time Transfert Protocol) standardisé en 1996, il fonctionne avec UDP ou TCP, spécialisé dans le transport de données possédant des contraintes temps réel. Il permet la diffusion de manière synchrone des flux temps réel transportés mais n'inclut pas le contrôle de la qualité de la communication. RTP reconstitue l'ordre des paquets, synchronise les médias et détecte la perte de paquets. Il est utilisé notamment pour la diffusion de flux vidéo en direct (ou pour la téléphonie sur Internet). Il ajoute en fait un en-tête aux paquets UDP pour donner des informations sur le média qui est transporté, le séquencement, afin que le récepteur puisse détecter les paquets perdus sur le réseau ou incorrectement reçus, et puisse éventuellement reconstituer un flux continu **[17]**.

- **But et fonctionnement**

Le but de RTP est de fournir un moyen uniforme de transmettre des données soumises à des contraintes de temps réel. Son rôle principal est de mettre en œuvre des numéros de séquence de paquets IP pour reconstituer les informations de voix ou vidéo même si le réseau sous-jacent change l'ordre des paquets **[18]**.
La responsabilité du contrôle de la transmission est entièrement laissée aux équipements d'extrémités. RTP leur apporte les informations qui leur manquaient pour s'en acquitter, sans pour autant interférer dans la transmission temps réel. En définitive RTP est un protocole de bout en bout pour les applications temps réel sur des services réseaux multicast ou unicast (conférence audio/vidéo interactive, diffusion vidéo/audio…). Il ne réserve pas de ressources dans le réseau, il n'apporte aucune fiabilité, et ne garantit pas de délais de livraison **[11]**.

RTP offre des moyens aux applications pour :

- Identifier le type de l'information transportée,
- Ajouter des marqueurs temporels et des numéros de séquence à l'information,
- Contrôler l'arrivée à destination des paquets **[18]**.

❖ **Remarque :**

RTP peut en théorie être utilisé seul mais il est généralement accompagné de **RTCP** (Real Time Control Protocol). Ce second protocole assure le trafic de contrôle **[19]**.

8.2. Le protocole RTCP

Le protocole du transport en temps réel (Real Transport Control Protocol) est basé sur la transmission périodique de paquets de contrôle à tous les participants d'une session. Ce protocole de contrôle des flux RTP permet de véhiculer des informations basiques sur les participants d'une session, et sur la qualité de service (QoS : Quality of Service).

- **Fonctionnement de RTP/RTCP**

RTP/RTCP est au-dessus du transport UDP/TCP, mais pratiquement au-dessus d'UDP. RTP et RTCP se contentent d'assister l'émetteur et le destinataire dans leur activité d'échange de données et c'est à l'application de s'adapter à la situation. Les paquets de données traversent des routeurs avant d'arriver à leur destination, or lors de leurs passages par le routeur ils sont disposés selon une file FIFO et sans aucune priorité. Afin de pouvoir réserver des ressources en matière de bande passante on introduit une priorité dans le traitement des paquets et ceci en faisant appel au protocole RSVP (Ressource reservation protocol) **[11]**.

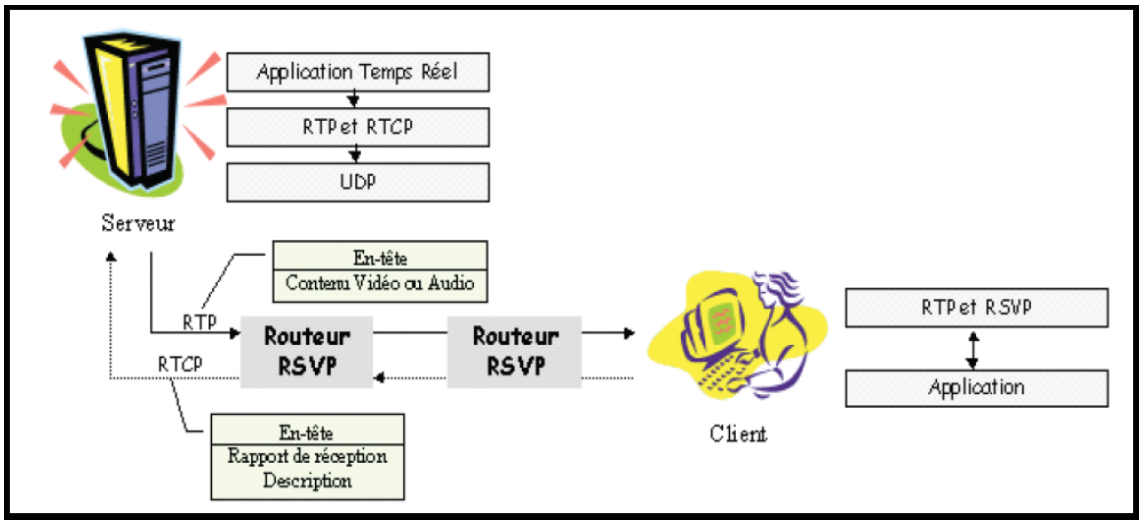

Figure II.4: Fonctionnement de RTP/RTCP

8.3. Le protocole RSVP

RSVP (Ressource reSerVation Protocol), souvent associé au protocole RTP, peut fournir fiabilité et qualité de service (QoS) qui font défaut à ce dernier (RTP est un protocole de bout en bout, il ne gère pas les paramètres liés au réseau). RSVP est un protocole de contrôle de réseau qui permet au destinataire des données de demander une certaine qualité de service (par exemple le délai

ou la bande passante) à travers le réseau. Il est utilisé par les applications « temps réel » afin de réserver les ressources nécessaires au niveau des routeurs.
Les routeurs communiquent via RSVP pour initialiser et gérer les ressources réservées aux sessions (une Session est un flot de données avec une destination particulière et un protocole de transport).
Ce protocole a pour objectifs :

- Etablissement et maintien d'un chemin unique pour la transmission de données.
- Elaboration d'un système d'ordonnancement des paquets.
- Création d'un module de contrôle pour les ressources des différents nœuds du réseau **[19]**.

8.4. Le protocole RTSP

RTSP (Real Time Streaming Protocol) a été développé par Real Networks, Netscape et l'Université de Columbia au. RTSP utilise RTP pour former les paquets contenant les données multimédia. Il est conçu pour diffuser efficacement des données audio-visuelles pour un grand nombre de personnes.
Contrairement au protocole RTP, qui ne permet qu'un streaming unidirectionnel (la transmission du contenu multimédia du serveur vers le client), le protocole RTSP est un protocole bidirectionnel qui utilise généralement TCP pour communiquer. Ce protocole de niveau applicatif est prévu pour fonctionner sur des protocoles tels que RTP/RTCP et RSVP **[11]**.

9. Le streaming vidéo sur les réseaux P2P

9.1. Définition :

Les systèmes de streaming basés sur la technologie P2P sont appelés systèmes de streaming P2P. Ce genre de système a émergé notamment grâce à la multiplication des accès haut-débit à Internet via des machines ayant de plus en plus de puissance de traitement et de stockage.
Dans les systèmes de streaming P2P, composés uniquement de peers, ces derniers prennent en charge les tâches du serveur. Les peers dans tel systèmes effectuent donc, en plus de la restitution de contenus multimédias, le streaming et la retransmission de ces contenus. Les systèmes de streaming P2P permettent aux utilisateurs de partager leurs contenus multimédias en mode streaming, combinant ainsi les avantages de la technologie P2P et du streaming.
Le streaming vidéo est de plus en plus utilisé par les entreprises et établissements universitaires. Cela permet notamment de réaliser des conférences, assemblées générales ou causeries qui peuvent dans de nombreux cas être couplées à d'autres applications :

- ✓ Affichage de slides ou d'images pour suivre facilement la conférence,
- ✓ Affichage de tableaux communs où chacun peut écrire à sa guise pour partager des idées avec les autres abonnés au service.

L'utilisation du streaming est un formidable vecteur de communication. Son impact est plus important que les médias traditionnels (Papier, mails, conférences).

Le retour sur investissement est évident et est engendré par les réductions de coûts (peu de ressources humaines et de matériel monopolisés). C'est un outil idéal de communication et de diffusion rapide (pour les services Marketing en particulier).

Il permet d'éviter la multiplication de la localisation des documents audio et vidéo dans l'entreprise. En effet, tout le contenu est centralisé sur un serveur de streaming. Il n'y a pas de téléchargement sur le poste local lors de la diffusion et donc le streaming garantit le respect et le suivi de la gestion des droits d'un média.

9.2. Types de systèmes de streaming P2P

On distingue deux familles de systèmes de streaming P2P : peer-to-peer (pseudo ou pur) et hybrides.

9.2.1. Architectures hybrides

Dans les architectures hybrides, il existe toujours un serveur qui prend en charge le streaming des contenus multimédias aux différents peers. Les peers, en prenant part à la diffusion globale de contenus, contribuent à alléger considérablement la charge du serveur.

La participation des peers dans le streaming est mise en œuvre par un mécanisme de multicast applicatif. Un serveur diffuse en continu vers un client (peer), qui à son tour, fait suivre le flux vers un autre peer qui en fait de même et ainsi de suit. En théorie, le serveur pourrait desservir des millions de personnes avec un seul flux de diffusion. Toutefois, dans la pratique, une telle approche n’est pas envisageable lorsque la chaine des peers acheminant successivement un contenu diffusé en continu est très longue. Ceci est dû aux contraintes temps réel inhérentes aux données audiovisuelles.

9.2.2. Architecture P2P

Dans la deuxième famille de systèmes de streaming P2P, les peers remplacent totalement le serveur de streaming. Le service de streaming est assuré par les peers pour les peers. Les fonctionnalités de recherche de ressources peuvent également être prises en charge par les peers (P2P pur), ou alors par un annuaire dédié mis en place à cette fin (pseudo P2P).

9.3. Les problèmes de streaming P2P

En plus de tous les problèmes hérités du protocole IP : bande passante non garantie, présence de perte dans le réseau, et délai de transmission et gigue non

bornés, le streaming des flux vidéo sur les réseaux P2P présente des problèmes spécifiques, comparé au streaming en mode client/serveur. En effet, les problèmes sont liés à l'organisation du réseau et aux comportements non déterministes des nœuds, nous énumérons ici les principaux défis techniques auxquels sont confortés les systèmes de streaming P2P :

- **Adaptation aux ressources :** Le streaming dans les réseaux P2P doit prendre en compte quelques contraintes liées à la présence de plusieurs nœuds dans le réseau, disposant du contenu désiré. La disponibilité de plusieurs copies du flux sur différents nœuds pose le problème de sélection des meilleurs nœuds pouvant assurer la qualité de service au récepteur, en particulier, pour maximiser la bande passante reçue.

- **Équité dans le service :** Un problème spécifique aux réseaux P2P, est lié au comportement dynamique des nœuds participants au réseau. Le plus important est le comportement connu sous le nom de « free-riding » qui consiste à quitter le réseau une fois le téléchargement terminé. Dans un réseau P2P, un nœud décide à lui seul quand rejoindre ou quitter un réseau P2P, il peut décider de ne plus partager la ressource demandée, ou aussi de limiter la bande passante « Upload » utilisée pour le streaming. Ce comportement imprévisible des nœuds pose un réel problème pour maintenir la qualité de service. Certains mécanismes incitent et motivent les utilisateurs à rester connectés au réseau P2P pour servir d'autres utilisateurs. Les mécanismes d'incitation peuvent être implémentés en conservant l'historique des contributions et en tenant compte de la quantité des ressources partagées. Néanmoins, le comportement dynamique et imprévisible des utilisateurs doit être géré et contrôlé afin de servir efficacement les clients. Des solutions d'adaptations doivent être utilisées pour, d'une part, adapter le flux vidéo aux ressources disponibles dans le réseau, mais aussi alléger le comportement dynamique des nœuds.

- **Passage à l'échelle :** Le principal défi des applications de streaming P2P est de mettre en place des techniques de distribution de données qui garantissent une distribution continue et permettre d'éviter une accumulation excessive de retard indépendamment du nombre d'utilisateurs du système.

- **Adaptation à la topologie du réseau :** La majorité des applications Peer-à-Peer, se basent sur la notion de voisinage (proximité) réseau pour construire leurs réseaux Overlay. Le fait de ne pas prendre en considération la topologie du réseau sous-jacent peut affecter considérablement la performance du système, comme les retards de réception des flux et l'utilisation inutile des ressources du réseau.

- **Robustesse :** Le fonctionnement des applications de streaming P2P de grande échelle repose entièrement sur les utilisateurs (nœuds), ce qui augmente la probabilité que le service soit perturbé par le comportement aléatoire des

utilisateurs. En effet, un nœud peut, à tout moment, entrer ou quitter le système ou tomber en panne: pour fournir un service sans interruption, le système doit donc être capable de s'adapter à cette dynamique, de limiter le plus possible les perturbations causées par le comportement aléatoire des nœuds.

10. Conclusion :

En conclusion le streaming est un système très intéressant pour les utilisateurs mais qui est parfois fort couteux à mettre en place du coté serveur.
Le fait de pouvoir regarder tout et n'importe quoi, en direct ou non, sans avoir à le garder sur votre disque dur, sans une longue attente et souvent gratuitement voir avec un prix très faible, est un réel plaisir pour l'utilisateur.
Il est interdit de mettre à disposition du contenu protégé par des droits d'auteur en streaming tel que des films sortis en salle de cinéma/dvd qui n'ont pas le droit d'être mis gratuitement à la disposition des utilisateurs, ce qui est donc illégal de la part du serveur
Dans ce chapitre, on a essayé de présenter certaines notions et définitions sur la technologie de streaming, les problèmes de la mise en œuvre de cette technologie ainsi que les solutions existantes, ensuite on a présenté les protocoles régissant ce type de transport de données. Enfin, nous avons énuméré les principaux défis techniques auxquels sont confrontés les systèmes de streaming sur les réseaux P2P.

Chapitre 03 :

Conception

1. Introduction :

Ce chapitre est consacré à la conception de notre travail, nous utilisons le langage de modélisation UML (Unified Modeling Language) représenté par un diagramme de cas d'utilisation en plus en une description de ces cas d'utilisation, nous aussi représentons l'architecture P2P utilisé parmi les connues.

- **Présentation du langage UML :**

UML est un langage de modélisation adopté et standardisé par Object Management Group (OMG). Il se définit comme un langage de modélisation graphique et textuel, destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML est aujourd'hui un outil de communication incontournable, utilisé sur des centaines de projets de par le monde. Etant un langage de modélisation, UML permet de faire une représentation abstraite d'un système réel ou non réel.

Parmi les diagrammes utilisés en ce langage, nous juste modélisons le diagramme de cas d'utilisation et une description de chacun de ses cas d'utilisation [20].

2. Objectif:

L'objectif de ce travail est la conception et la réalisation d'une application P2P. L'application comporte les objets fondamentaux :

- Partage de fichier.
- Téléchargement des fichiers après faire une recherche selon un mot clé.
- L'envoie des fichiers vers un autre peer.
- L'échange des messages textuels entre les pairs.
- Appel vidéo.

3. Architecture p2p streaming utilisée :

L'architecture centralisée d'un réseau peer-to-peer est connue pour ne pas garantir une bonne résistance aux défaillances et pour ses goulots d'étranglements. En effet, si le serveur central connait une panne, le réseau cesse de fonctionner. Ce type d'architecture ne garantit pas un bon passage à l'échelle, car le nombre de clients connectés au serveur central est limité par la puissance de ce dernier ainsi que par les performances du réseau en termes de latence et de bande passante.

Une architecture purement décentralisée fait face aux problèmes posés par une architecture centralisée. Cependant, elle présente une grande complexité de gestion, et se caractérise par des performances réduites dues aux inondations causées par les broadcasts et les multicasts, surtout si le TTL (Time To live) caractérisant un tel réseau est relativement élevé, de plus la recherche dans un réseau de ce type d'architecture n'est pas déterministe. En effet, les objets situés à une distance importante relativement au TTL ne sont pas atteints au cours de la recherche. Ceci représente, dans notre cas, un manque de précision lors du calcul de la disponibilité des fichiers.

Malgré ses inconvénient nous choisissons la première pour des raisons tels que la facilité d'implémentation, aussi nous utilisons le réseau local qui n'exige pas des grandes performances de plus la complexité du deuxième architecture.

Nous utilisons deux versions pour notre application, les deux sont centralisés la différence est que nous limitons les fonctionnalités du serveur dans la deuxième versions qui plus développée que la première comme nous allons vu prochainement.

4. Diagramme de cas d'utilisation :

Les cas d'utilisation sont une technique de description du système étudié privilégiant le point de vue de l'utilisateur. Un cas d'utilisation est une façon spécifique d'utiliser le système. Il est composé d'une séquence d'action déclenchée par un acteur externe et qui produit un résultat identifiable.

Dans notre cas nous modélisons un seul diagramme d'activité pour les deux versions. Nous expliquons les différences entre eux après en la description de ces cas d'utilisation.

4.1. Les acteurs :

L'acteur est une entité externe qui agit sur le système. Dans notre application les acteurs qui participent au déroulement de l'application sont :

- **L'utilisateur (peer ou client)** : par l'interface graphique il peut partager, télécharger des fichiers, télécharger des vidéos en temps réel et communiquer avec d'autre peer.

- **Le serveur** : pour gérer le réseau, gérer la liste de fichiers partagés et se charger à la transmission des messages.

4.2. Les cas d'utilisation :

Sont des objectifs du système motivés par un besoin d'un ou plusieurs acteurs :

- **Cas d'utilisation d'utilisateur :** on cite
  - Se connecter
  - Partager fichier
  - Rechercher fichier
  - Télécharger fichier
  - Chat (l'échange des messages textuels)
  - Appel vidéo

- **Cas d'utilisation de serveur :**
  - Gérer le réseau
  - Afficher la liste des fichiers

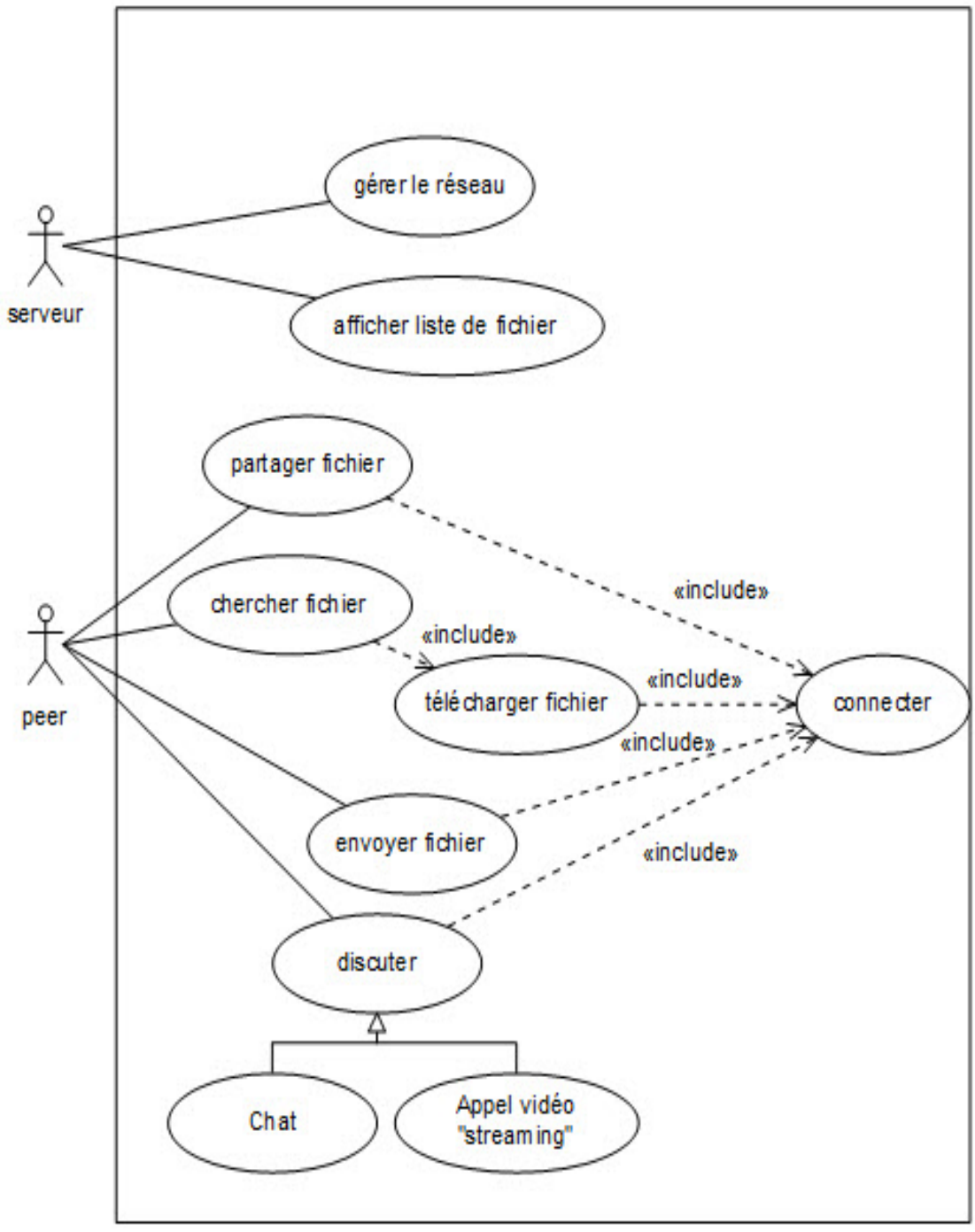

Figure III.1 : Diagramme de cas d'utilisation

5. Description des cas d'utilisation :

Dans cette description, nous expliquons chaque cas d'utilisation en détails pour les deux versions pour représenter la différence entre eux, est cela pour les deux acteurs serveur et peer, nous commençons par le premier (serveur) puis nous passons vers le deuxième (peer).

5.1. L'acteur « serveur » :

5.1.1. Cas d'utilisation « gérer le réseau » :

Ce tableau explique la différence entre les fonctionnalités des deux serveurs :

Les fonctionnalités	Version 1	Version 2
Gérer la connexion (ajouter un peer)	✓	✓
Gérer la déconnexion (supprimer un peer)	✓	✓
Diffuser la liste des peer connectés	✓	✓
diffuser les requêtes aux peers	✓	✗
Recevoir les requêtes de peers (rechercher fichier ,…etc.)	✓	✗

Tableau III.1 : la différence entre les deux serveurs

5.1.2. Cas d'utilisation « affiche la liste des fichiers » :

Le serveur affiche dans un tableau la liste des fichiers partagés, ce tableau contient les informations suivants : le nom de fichier, la taille de fichier et le nom de peer.

Le serveur mettre à jour le tableau en cas de connexion d'un nouveau peer au réseau (ajouter ses fichiers) ou de déconnexion d'un peer (supprimer ses fichiers)

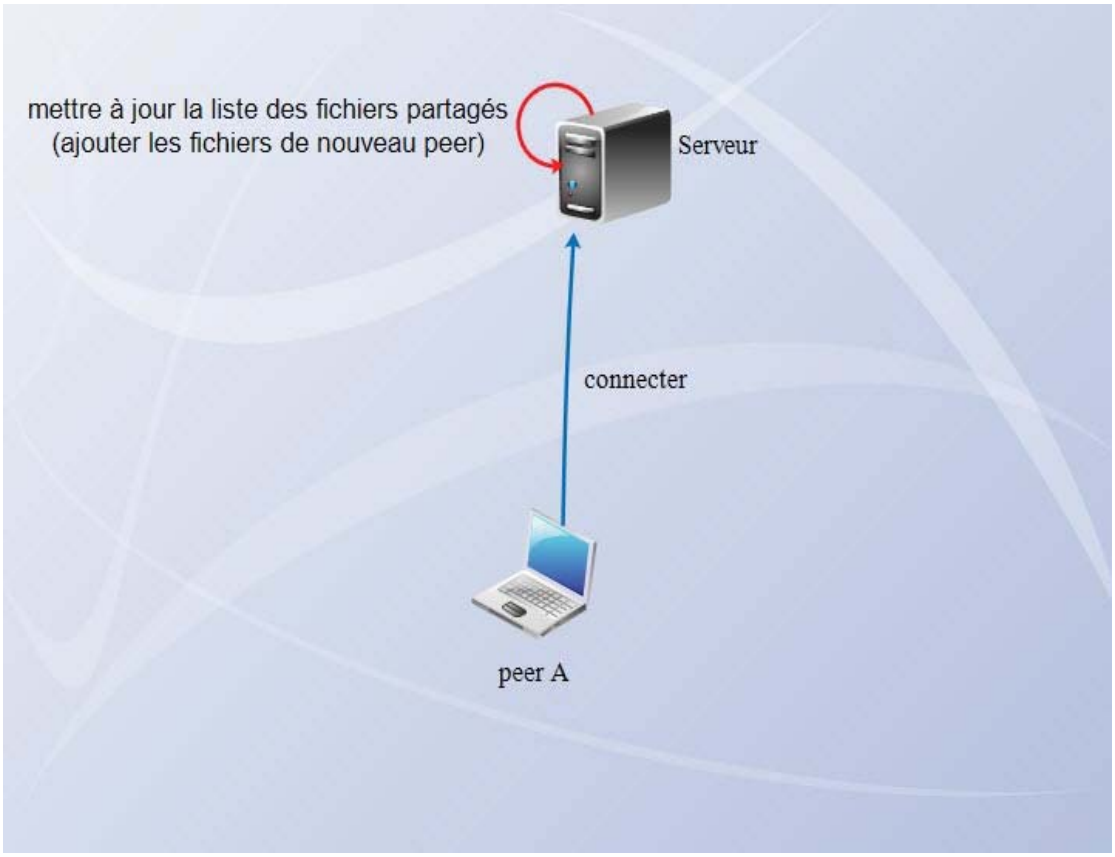

Figure III.2 : ajouter les fichiers de nouveau peer à la liste des fichiers partagés

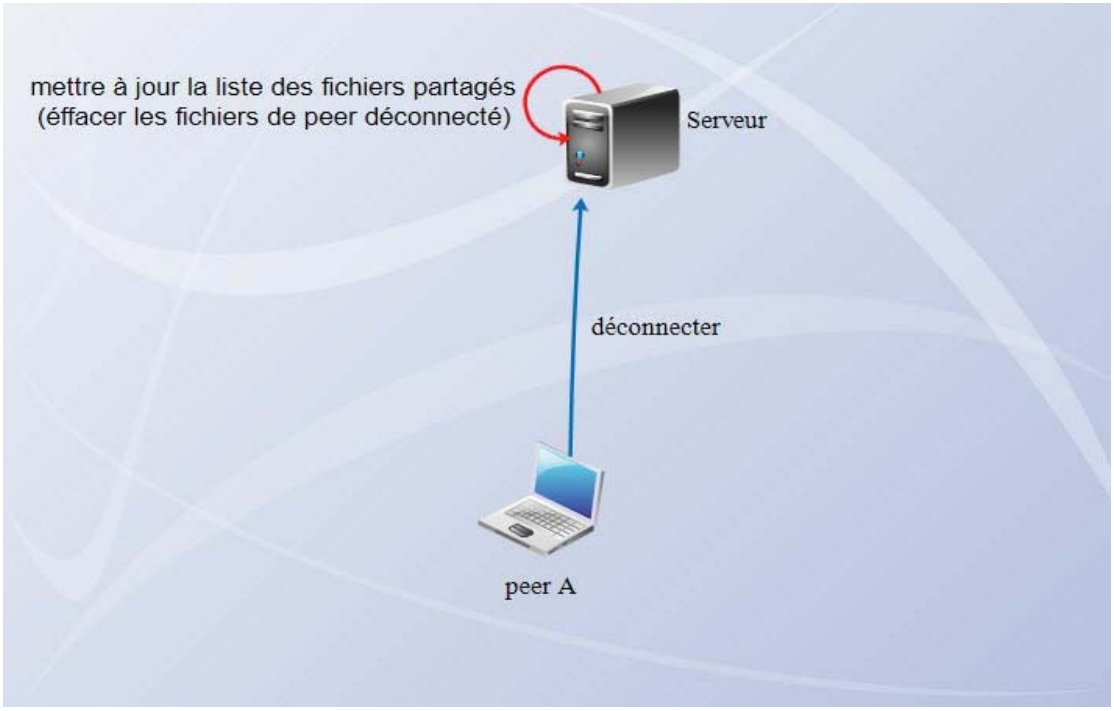

Figure III.3 : effacer les fichiers de peer déconnectés

5.2.L'acteur « peer» :

5.2.1. Cas d'utilisation « partager fichier » :

Le peer peut partager des fichiers dans sa propre base de données, le serveur insérer le nouveau fichier dans la liste des fichiers partagés, le peer aussi peut supprimer un ou plusieurs fichiers dans la base de données

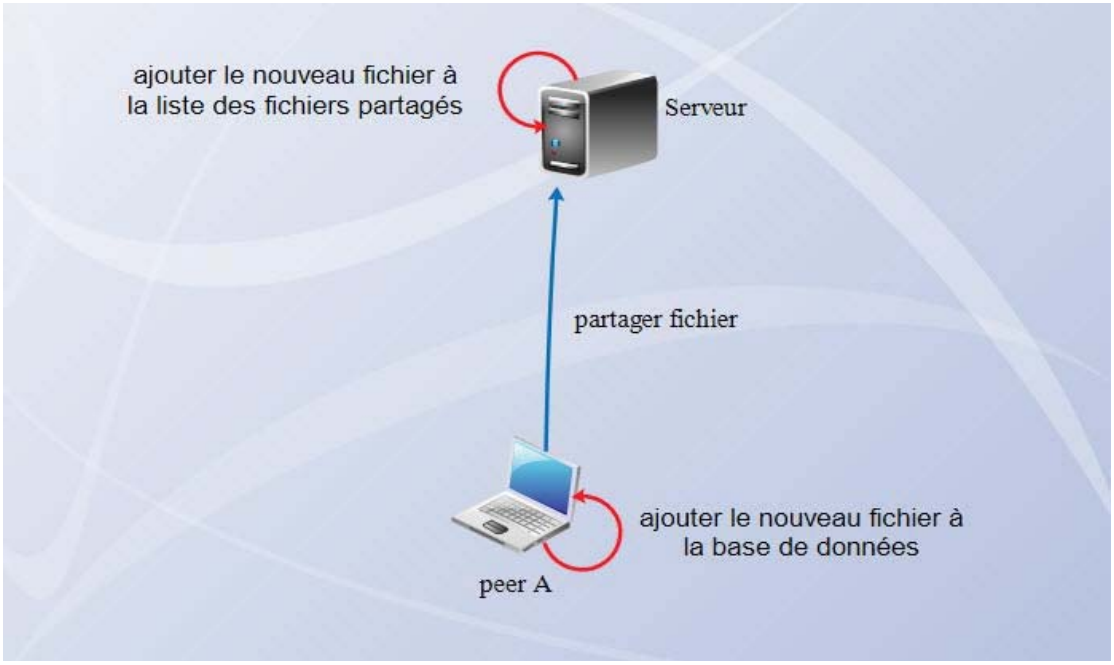

Figure III.4 : partager un fichier

5.2.2. Cas d'utilisation « chercher fichier » :

5.2.2.1. Version 1 :

Dans cette versions le peer envoie une requete de recherche de fichier au serveur, puis le serveur effectue une recherche dans la liste des fichiers partagés , ensuite il retourne le résultat au peer.

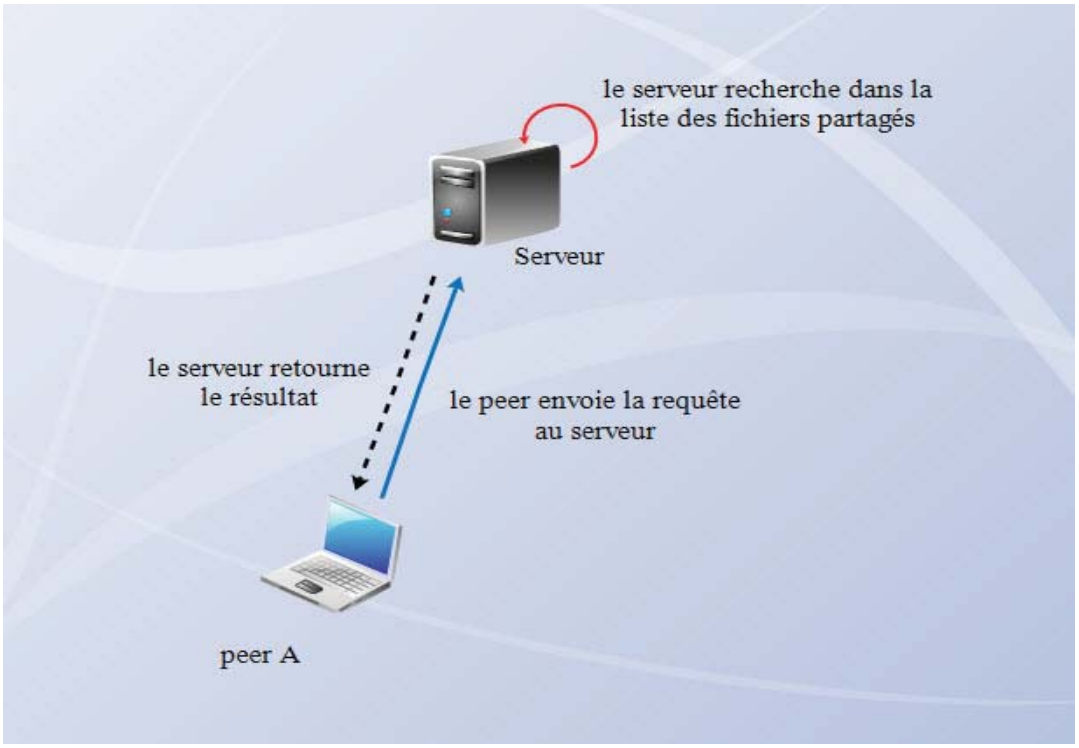

Figure III.5 : rechercher un fichier version une.

5.2.2.2. Version 2

La recherche se fait directement sur les bases de données des autres peers, chacun retourne son résultat au peer.

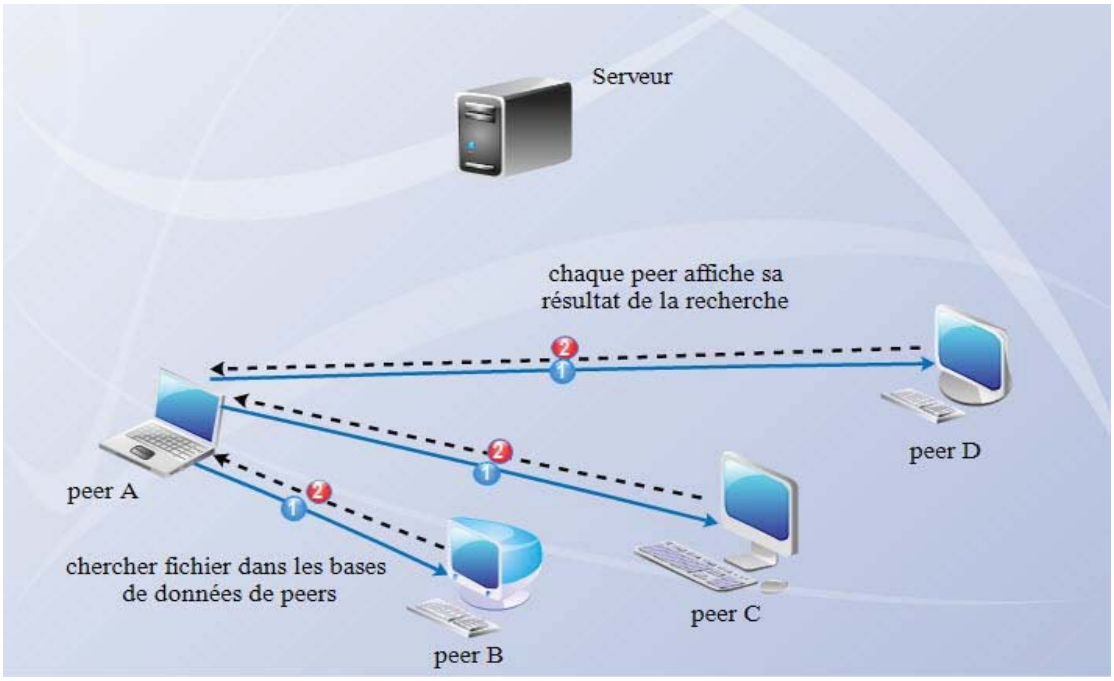

Figure III.6 : rechercher un fichier version 2.

5.2.3. Cas d'utilisation « télécharger fichier » :

Après faire la recherche le peer peut télécharger un fichier après le sélection.

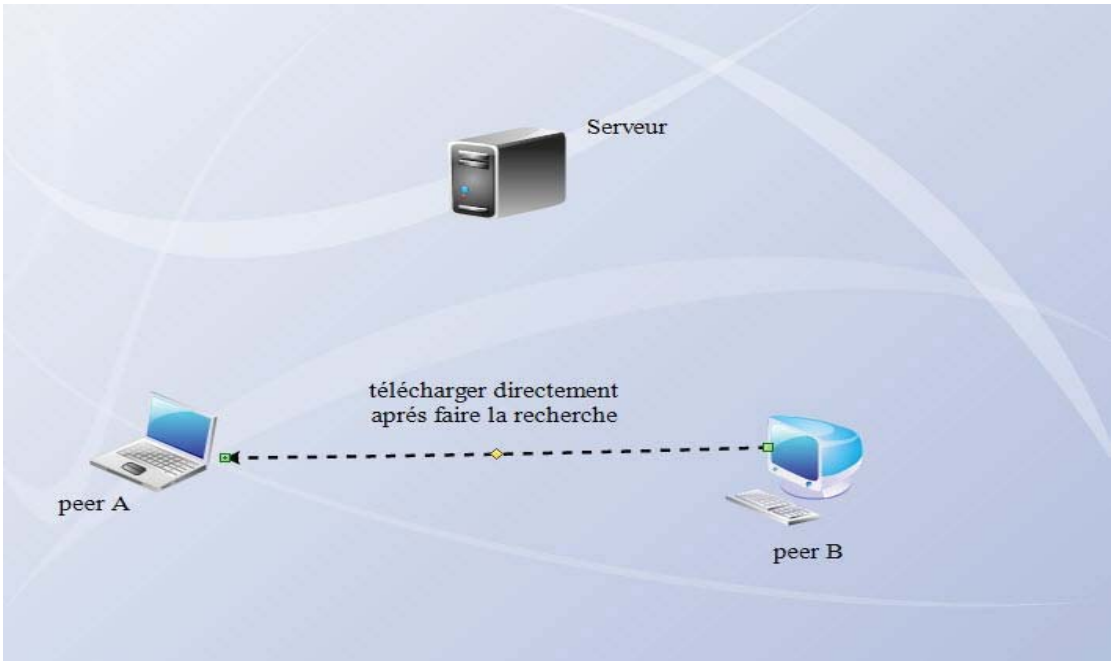

Figure III.7 : télécharger un fichier.

5.2.4. Cas d'utilisation « envoyer fichier » :

Le peer peut sélectionner un fichier puis le envoyer vers un peer destinataire après la confirmation de ce dernier.

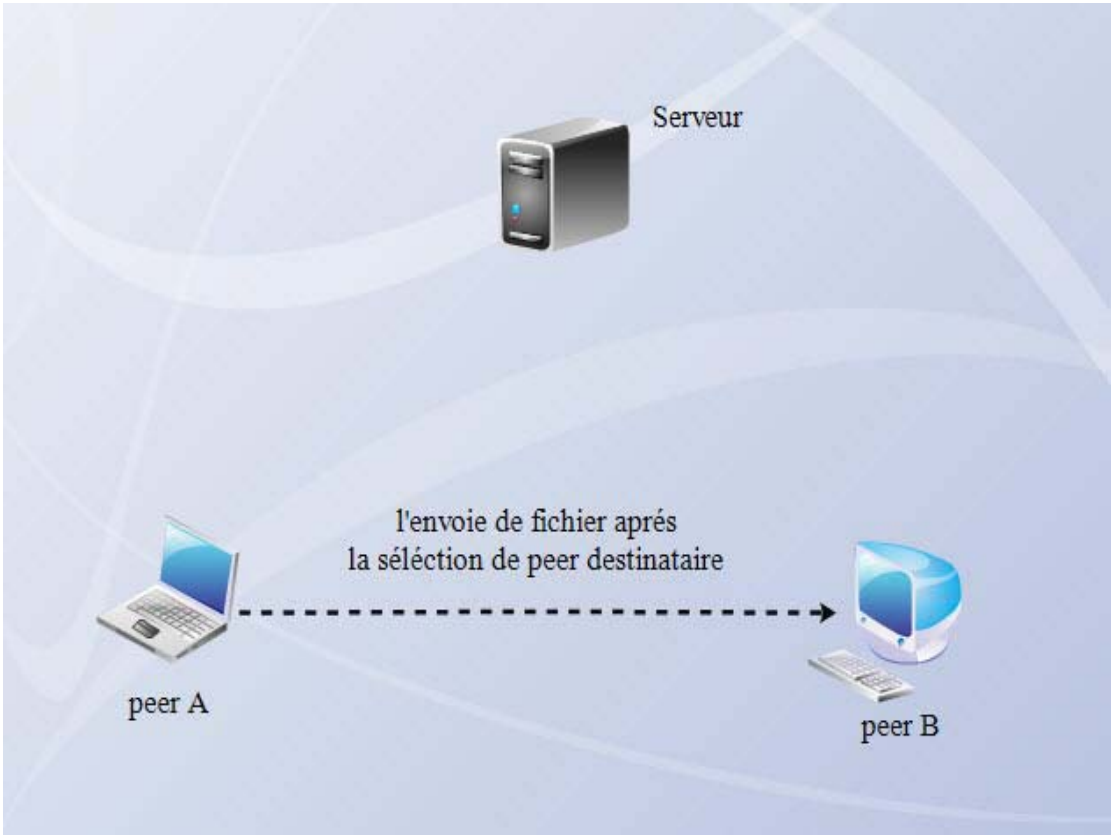

Figure III.8 : envoyer un fichier.

5.2.5. Cas d'utilisation « discuter : chat » :

Le peer peut echanger les messages avec les autres peers connectés.

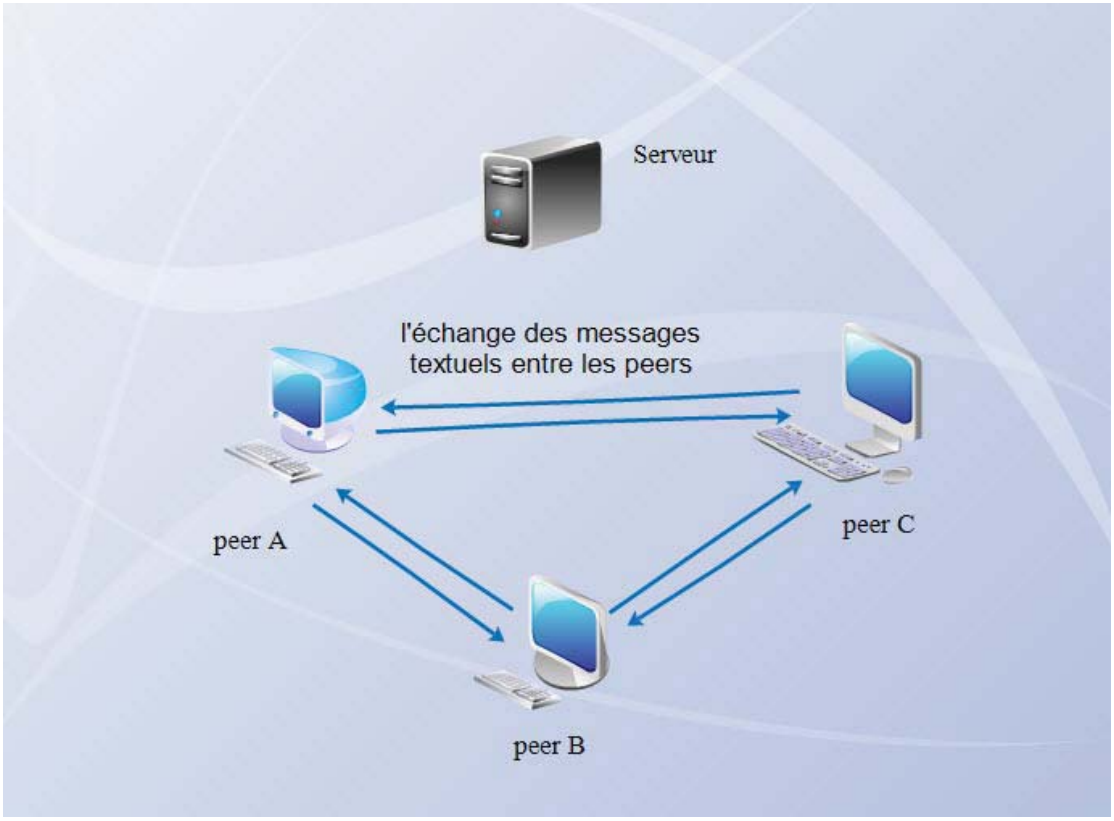

Figure III.9 : discuter « chat ».

5.3. Cas d'utilisation « discuter : appel vidéo (streaming) » :

Utilise la Cam le peer peut faire des appels vidéos avec d'autre, le peer au même temps émet et reçoit des flux audio et vidéo

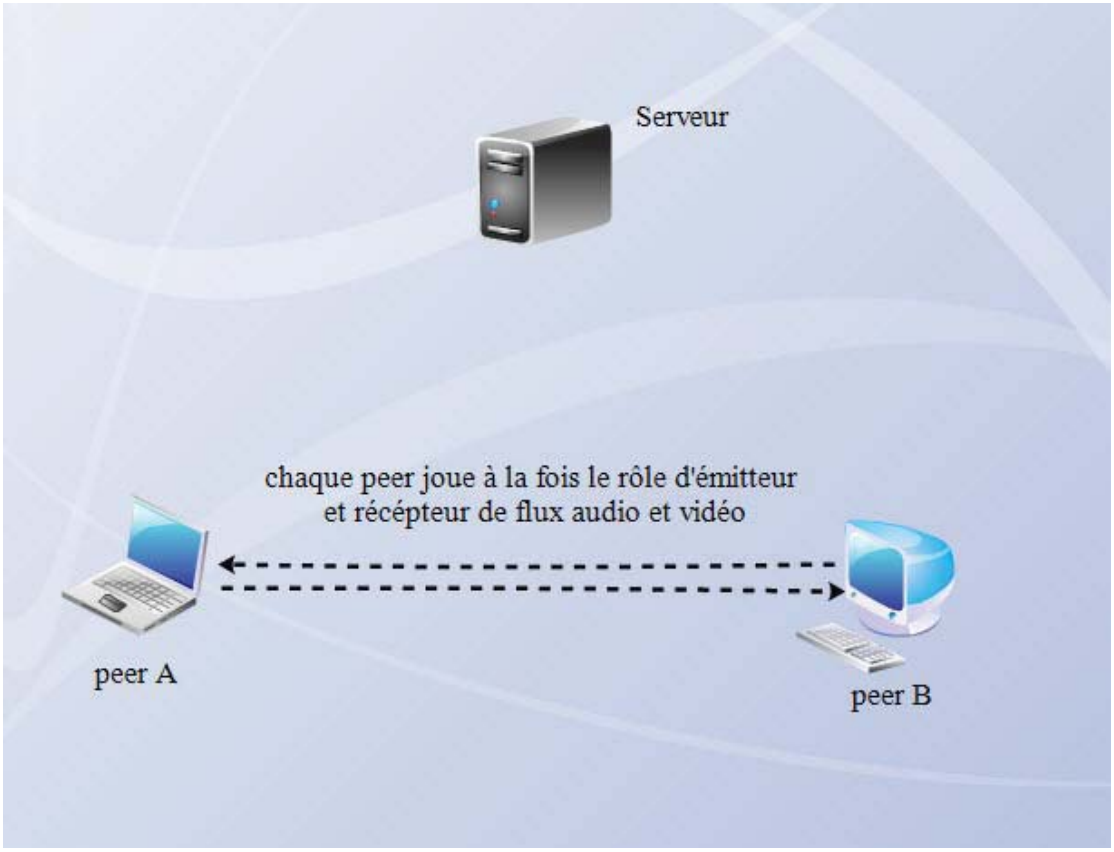

Figure III.10 : discuter « appel vidéo ».

6. Conclusion

Ce chapitre a décrit l'étape la plus importante du cycle de vie du logiciel. A travers les outils de la méthodologie UML, on a pu décrire l'architecture P2P utilisé dans un premier temps, Dans un deuxième temps on a décrit les principaux acteurs du système et ses fonctionnalités, et enfin on a détaillé quelques cas d'utilisation. Cette étape a facilité certes l'étape suivante qu'est l'implémentation.

.

Chapitre 04 :

Outils de développement Implémentation

1. Introduction :

Après la réalisation de l'étape d'analyse, il faut passer à la dernière phase de la démarche utilisé à savoir la phase d'implémentation.

Dans cette partie de projet nous présentons d'abord les outils utilisés pour le développement de l'application java NetBeans (l'environnement de développement) et Access (comme un SGBD). Ensuite, nous allons préparer les données nécessaires pour l'application, en convertissant les classes entités en une base de données. Enfin, nous décrivons quelques interfaces du logiciel réalisé.

2. Environnement de développement :

2.1.Plateforme de développement :

Java :

Pour développer notre application, nous avons opté pour le langage Java. Java est à la fois un langage de programmation informatique orienté objet et un environnement d'exécution informatique portable. La diversité des environnements de développement comme JBuilder, NetBeans, Eclipse, …font que java soit un outil puissant pour la conception des différentes applications.de plus Java permet l'utilisation des sockets.

- **Sockets** :
  Mécanisme universel de bas niveau, utilisable depuis tout langage (exemple : C, Java) qui permettent l'utilisation de l''interface de transport (TCP-UDP).

✓ Principe de fonctionnement des sockets : 3 phases

a. Le serveur crée une "socket serveur" (associée à un port) et se met en attente

b. Le client se connecte à la socket serveur, deux sockets sont alors créées : une "socket client", côté client, et une "socket service client" côté serveur. Ces sockets sont connectées entre elles.

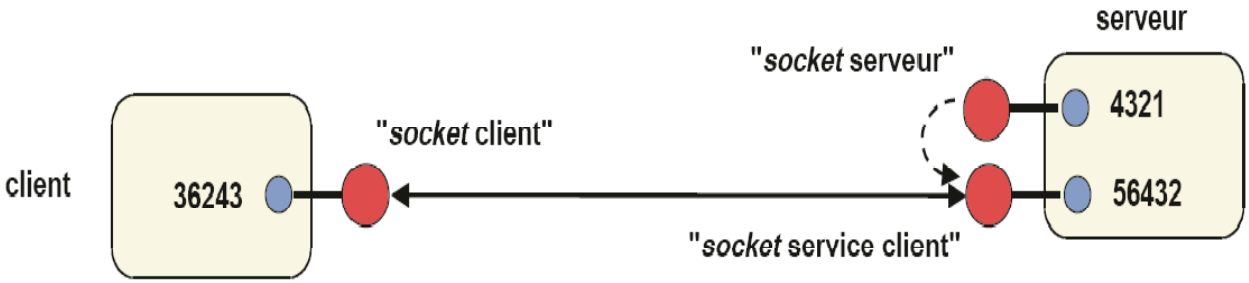

Figure VI.1 : principe des sockets

c. Le client et le serveur communiquent par les sockets, l'interface est celle des fichiers (read, write). La socket serveur peut accepter de nouvelles connexions.

✓ **Deux réalisations possibles :**

a. Mode connecté (protocole TCP) :

- Ouverture d'une liaison, suite d'échanges, fermeture de la liaison.
- Le serveur préserve son état entre deux requêtes.
- Garanties de TCP : ordre, contrôle de flux, fiabilité.
- Adapté aux échanges ayant une certaine durée (plusieurs messages).

b. Mode non connecté (protocole UDP)

- Les requêtes successives sont indépendantes.
- Pas de préservation de l'état entre les requêtes.
- Le client doit indiquer son adresse à chaque requête (pas de liaison permanente).
- Pas de garanties particulières.
- Adapté aux échanges brefs (réponse en 1 message).

✓ **Points communs**

- Le client à l'initiative de la communication ;
- Le serveur doit être à l'écoute.
- Le client doit connaître la référence du serveur [adresse IP, n° de port]
- Le serveur peut servir plusieurs clients (1 thread par client).

✓ **Les API Réseau de Java :** les packages java.net, javax.net

✓ **Principales classes :**

- Adresses IP : « InetAddress »
- Socket TCP : « Socket, ServerSocket, JSSE (Java Secure Socket Layer) »
- Sockets UDP : « DatagramSocket, DatagramPacket »
- Sockets MultiCast : « MulticastSocket, DatagramPacket »
- Classes réseaux au niveau application : « URL, URI, URLConnection, HttpURLConnection, JarURLConnection »

JDK :

JDK Java Développent Kit est nécessaire pour développer des applications Java, Il comprend JVM, les bibliothèques de base et d'autres composants supplémentaires pour exécuter des applications et des applets écrites en Java.

NetBeans :

NetBeans est un environnement de développement intégré (EDI), placé en opensource par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Développent 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, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible Windows, linux, Solaris, MAC OS, ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).Un environnement Java Développements Kit JDK est requis pour les développements en Java

 **Accès** :

MS Access est un logiciel utilisant des fichiers au format Access (extension de fichier *.mdb* pour Microsoft DataBase (extension *.accdb depuis la version 2007)). Il est compatible avec les requêtes SQL (Structured Query Language) et dispose d'une interface graphique pour saisir les requêtes (QBE - Query by Example - « Requête par l'exemple »). Il permet aussi de configurer, avec des assistants ou librement, des formulaires et sous-formulaires de saisie, des états imprimables (avec regroupements de données selon divers critères et des totalisations, sous-totalisations, conditionnelles ou non), …etc.

2.2.Exploitation du streaming :

2.2.1. Le concept JMF :

JMF Java Media Framework est une large et transparente API utilisée pour les applications multimédias (audio/vidéo), elle permet d'exploiter le streaming avec le langage java.

Ses possibilités sont nombreuses, on peut faire un ensemble important d'applications autour de multimédia avec JMF comme :

- ✓ Lecture de différents formats multimédia dans une applet java. Les formats supports sont AAU, AVI, MIDI, MPEG, QUICKTIME et WAV.
- ✓ Capture de données multimédias.
- ✓ Développement des applications java utilisant le streaming ou les vidéo conférences.
- ✓ L'accès à un large type de données.
- ✓ Offrir un support pour le protocole RTP.
- ✓ Lecture d'un flux de données (média) depuis internet.

- ✓ Capture du son et de l'image depuis un microphone et/ou une caméra et enregistre les données dans un format supporté.

Le format MP3 est pris en charge uniquement sur la plate-forme Windows, pour cela il faut installer le plugin mp3.
Bien que JMF pourrait soutenir un certain type de médias, elle peut ne pas supporter la compression spécifique (CODEC) utilisée pour les données à l'intérieur du fichier ressource.
Dans d'autre cas, on voit la vidéo s'afficher sans entendre le son. Cela est dû au fait que JMF ne prend pas en charge le format de compression de la piste audio. D'ailleurs, le format .AVI n'est pas totalement supporté par JMF et on risque de rencontrer des soucis avec des .AVI récents.

2.2.2. Fonctionnalité de JMF :

Autre la possibilité d'incorporer de l'audio et de la vidéo dans vos applications, JMF permet l'envoi et la réception de médias à partir du réseaugrâce à JMF RTP.

L'API permet en outre :

- ✓ la capture audio et vidéo avec un microphone et une caméra, puis l'enregistrement de ces données dans un format donné.
- ✓ Le traitement des fichiers vidéo (multiplexage, segmentation, compression).
- ✓ transmission et la réception de l'audio et de la vidéo en temps réel sur Internet (streaming vidéo) ou sur le réseau comme déjà mentionné.
- ✓ le transcodage des données, c'est-à-dire le passage d'un format à un autre avec de nouveaux codecs tout en admettant une architecture de plugin qui permet d'intégrer facilement de nouveaux codecs.

2.2.3. Architecture de JMF :

Comme le montre la figure ci-dessous, des appareils tels que le lecteur de cassettes et le magnétoscope fournissent un modèle familier pour l'enregistrement, le traitement et la présentation des médias à base de temps. En effet, lorsque vous regardez un film à l'aide d'un magnétoscope, vous devez fournir le flux média au magnétoscope en insérant une cassette vidéo. Le magnétoscope lit et interprète les données sur la bande et envoie les signaux appropriés à votre téléviseur et aux haut-parleurs pour la diffusion vidéo.

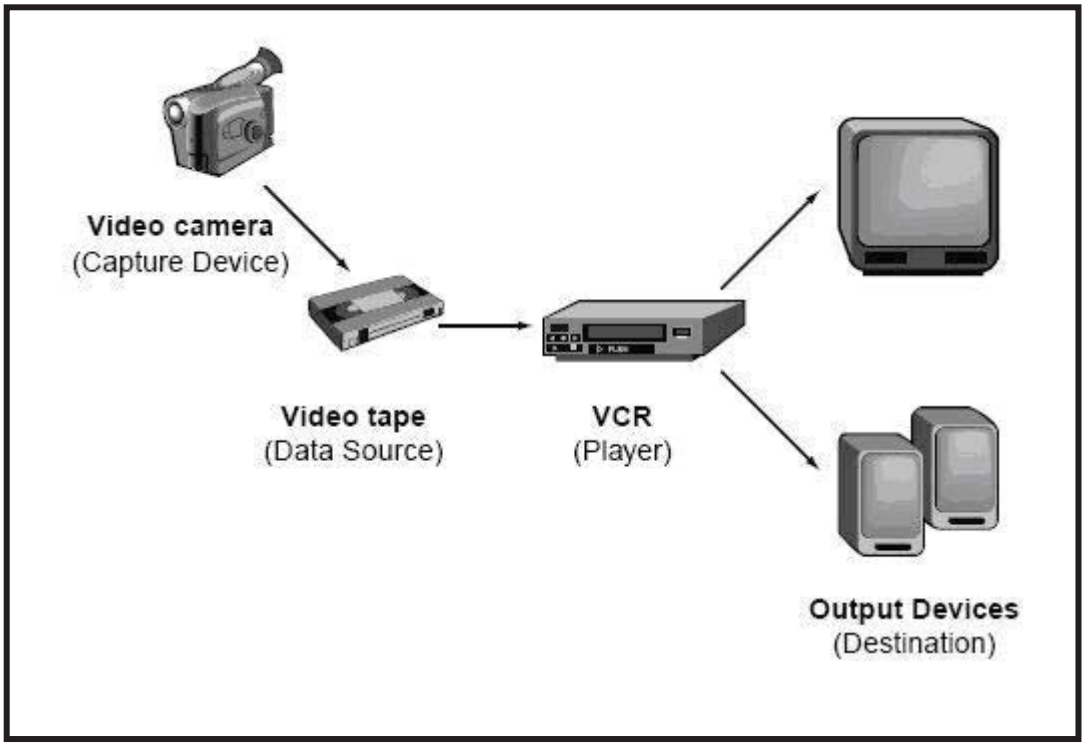

Figure IV.2 : architecture JMF

La chaine de traitement en JMF peut être assimilée à ce processus de diffusion en admettant quatre nœuds principaux pour le traitement :

- ✓ **Capture Device** : est le périphérique de capture ;
- ✓ **Data Source** : c'est la source de données qui modélise les sources de données audio et vidéo ;
- ✓ **Processor (Player)** : qui assemble les sources de données pour construire un format vidéo pouvant être affiché à l'utilisateur ;
- ✓ **OutPutDevice** : est la sortie du player récupérée par le DataSink qui peut être un fichier, un écran, etc.
- ✓ **Datasink** : une référence de l'interface DataSink récupère le média du player et le redirige vers une destination. Un DataSink permet par exemple d'enregistrer un média dans un fichier ou de l'afficher directement sur l'écran.

JMF utilise un modèle objet semblable à celui d'un système stéréo qui englobe les modules suivants :

a. Data source :

Représente le média audio, vidéo ou la combinaison entre les deux, elle peut être un fichier où provenir d'un flux internet. Le principe fondamental de cette classe est qu'une fois la provenance et le protocole définis, ces

derniers sont encapsulés automatiquement pour délivrer le média. Une fois créée, la DataSource peut être couplée à un player afin d'être jouée. De ce fait, celui-ci n'a pas à se soucier d'où vient la source ni de sa forme initiale. On peut récupérer les données d'une variété de sources comme depuis un fichier local, un réseau ou encore « un broadcast internet en live ». La DataSource est classée en deux catégories suivant le type d'initialisation du transfert de données :

- ✓ Pull Data Source : le client initialise le transfert de données et contrôle le flux de données depuis la source. (HTTP et Fichier serveur sont de bons exemples pour ce type d'initialisation).
- ✓ Push Data Source : le serveur initialise le transfert de données et contrôle le flux de données depuis une source spécifique (broadcast principalement).

b. Capture device :

Cet objet est un périphérique matériel utilisé comme une source de données. Cela peut être un microphone, un appareil photo ou une caméra. La source de données peut être connectée à un Player permettant ainsi un affichage à l'écran ou un traitement afin d'être convertie dans un format spécifique ou encore sauvegardée pour une utilisation ultérieure. Ces périphériques peuvent être également classés en « push » ou « pull » périphériques. Avec une source de type « pull », l'utilisateur contrôle les séquences de captures (par exemple un appareil photo). A l'inverse, un microphone est de type « push » car il fournit en continu un flux audio.

c. Le Player :

C'est un objet dont le rôle est la lecture des données audio ou vidéo à partir de la source de données et de leur renvoi, après traitement et à un moment précis à la carte son ou à la carte graphique pour affichage. C'est similaire à un lecteur CD qui lit un CD audio et renvoie le son au haut-parleur. Le player est l'acteur capital de l'API JMF. Il contrôle le chargement, l'acquisition des ressources et l'exécution (démarrage, arrêt, vitesse d'exécution...) des données multimédia. Avant d'émettre du son à la carte son ou des informations vidéo à la carte graphique, le player doit passer par six états, quatre d'entre eux sont fondamentaux et les autres sont considérés comme intermédiaires. Bref, un état intermédiaire se situe entre deux états fondamentaux.

Vu que le player effectue un traitement ou attend la disponibilité d'une ressource, on peut en fait considérer les états intermédiaires comme des états d'attente pour passer à un état fondamental.

Un « Player » est un objet avec état, il passe par un ensemble d'états dans un ordre bien définit

- ✓ **Unrealized :**
  Le player est instancié comme un nouveau né qui ne connait rien de son environnement et des ressources qu'il doit réaliser. On peut obtenir cet objet à partir du gestionnaire de document multimédia (la classe Manager).
- ✓ **Realizing**
  C'est un état intermédiaire dans lequel le Player détermine ses ressources qu'il pourrait partager avec d'autre Player. Ce sont la plupart du temps des ressources réseau. Du fait qu'elles peuvent être partagées entre plusieurs players, on peut les appeler des ressources non exclusives.
  Le passage à cet état est fait suite à un appel à la méthode **realize().**
- ✓ **Realized**
  Le Player connait à cet état le type de document multimédia qu'il doit traiter et sait quelles ressources il doit acquérir. De ce fait, il peut préparer le composant visuel pour l'affichage de la vidéo ainsi que le panneau des boutons de commandes et de lecture.
- ✓ **Prefetching**
  lorsque l'on appelle la méthode prefetch (), le Player se positionne à l'état Prefetching. Cet état déclare que le Player est prêt à présenter le média. Durant cette période, le Player pré-charge les données de son média, obtient les ressources exclusives et fait l'ensemble des traitements utiles à la lecture du média.
- ✓ **Prefetched**
  À cette étape, le palyer aura reçu toutes ses ressources exclusives pour traiter convenablement le document multimédia. Il est prêt ainsi à le présenter correctement.
- ✓ **Started**
  L'appel à la méthode **Start ()** mène le Player à l'état **started** entrainant ainsi l'affichage de la vidéo à l'écran.
  Un appel à la méthode **stop ()** permet de retourner à l'état **Prefetched**.

d. Processor :

Le processor est un type de Player qui se charge du contrôle et modification des données en entrée et/ou en sortie.

En plus des états de Player le processor en possède deux autres :

- ✓ **Configuring** : un Processor entre dans cet état à partir de l'état unrealized et lorsque la méthode configure () est appelée. Un Processor est également dans cet état lorsqu'il se connecte à la DataSource,

démultiplexe le flux d'entrée, et accède aux informations liées au format des données d'entrée.

- ✓ **Configured** : un Processor entre dans cet état lorsqu'il est connecté à la DataSource et lorsque le format est déterminé.

e. DataSink :

L'objet DataSink est l'interface de base pour lire un média à partir d'une DataSource ou envoyer le résultat vers une destination. DataSink peut être utilisé pour écrire un média dans un fichier.

f. Format :

Un objet Format représente un format de média précis. Il décrit le nom de l'encodage et le type de données que le format accepte. Il existe deux classes principales hesitant de Format :

- ✓ **AudioFormat** : format audio
- ✓ **VideoFormat** : format vidéo qui se décompose suivant 6 classes :
  1. H261 Format
  2. H263 Format
  3. IndexedColor Format
  4. JPEG Format
  5. RGB Format
  6. YUV Format

g. Manager :

Le Manager est un objet intermédiaire entre deux machines, physiquement, il n'a pas de représentation. C'est une classe qui permet de connecter deux objets différents. Comme exemple, créer un Player à partir d'une DataSource.

JMF fournit 4 types de Manager :

- ✓ **Manager** : il est utilisé pour créer des Player, Processor, DataSource, DataSink.
- ✓ **Package Manager** : ce manager gère un registre de package contenant les classes JMF, telles que les Player, Processor, DataSource, DataSink personnalisés.
- ✓ **CaptureDeviceManager** : ce manager contient le registre des périphériques disponibles.
- ✓ **Plugin Manager** : ce manager maintient un registre du plug-in JMF (composants de traitement) disponibles.

3. L'API JDBC " Java DataBase Connectivity":

Il existe des nombreuses bases de données sur le marché. Afin d'uniformiser les accès aux bases de données sous Windows, Microsoft a développé une interface appelée ODBC (Open DataBase Connectivity). Cette couche cache les particularités de chaque base de données sous une interface standard. Il existe sous Windows de nombreux pilotes ODBC facilitant l'accès aux bases de données sous une interface standard. Il existe sous Ms Windows de nombreux pilotes ODBC facilitant l'accès aux bases de données. Voici par exemple, une liste de pilotes ODBC installés sur une machine Windows par défaut :

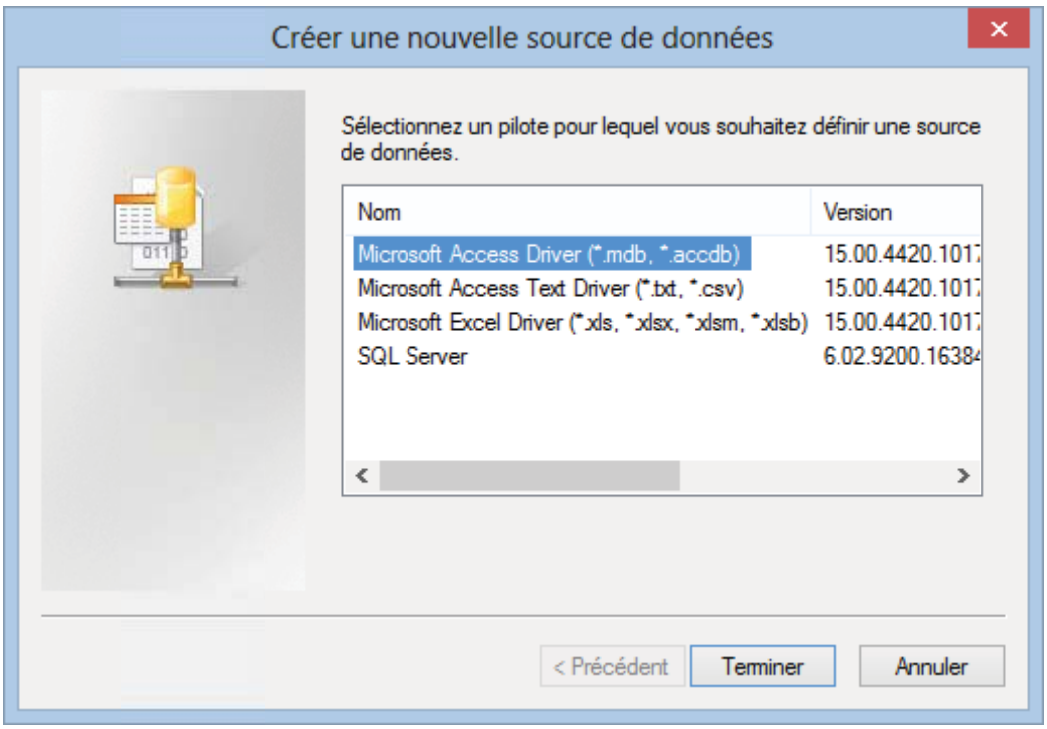

Figure IV.3 : Source de données utilisateur.

❖ **Configuration une base de données sur un pilote ODBC**

Pour permettre à une application Java d'accéder à une base de données (nommée par exemple Utilisateur), il faut la définir comme une source de données utilisateur dans le gestionnaire des bases ODBC :

a. Initialement il faut vérifier que le pilote ODBC adéquat à votre base de données est disponible dans la listes des pilotes ODBC fournit par votre système d'exploitation, dans notre cas la base de données est une base de données Access.
b. Aller : Panneau de configuration / outil d'administration, puis choisir Sources de données(ODBC), une fenêtre est apparue permet de sélectionner le pilote qui sera utilisé par la source de données.

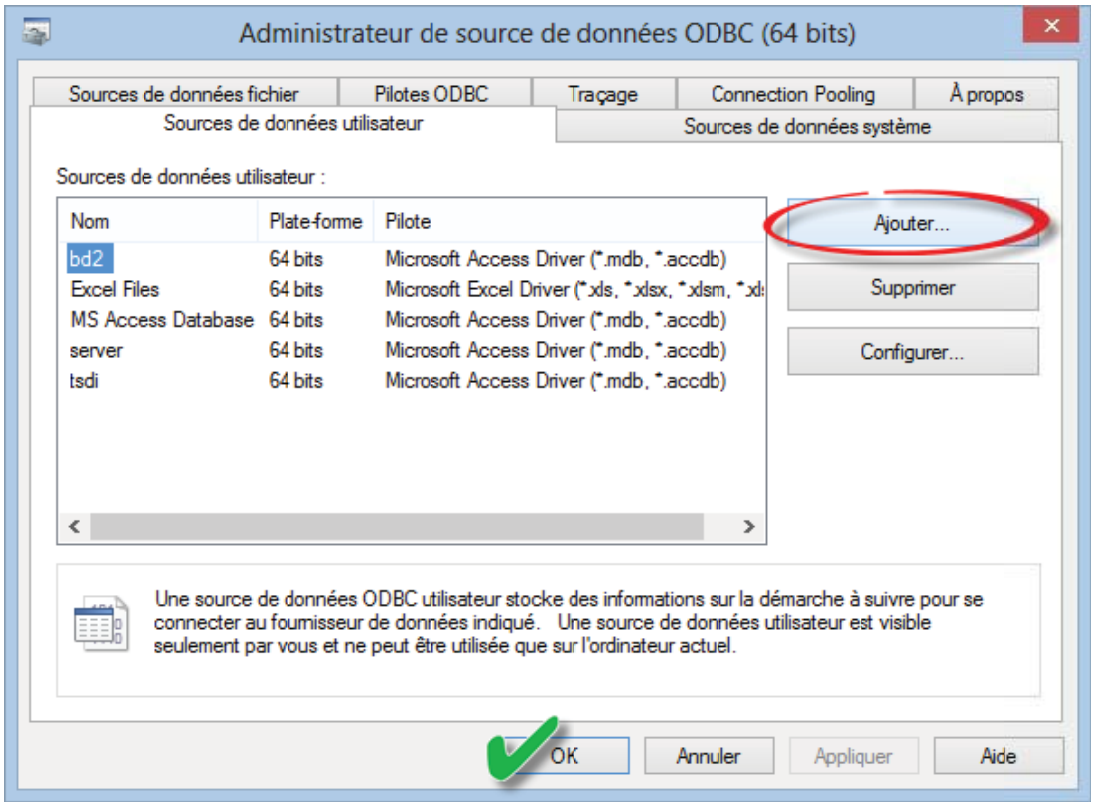

Figure IV.4 : Pilotes utilisées par la source de données.

c. Choisir l'onglet **"Sources de données utilisateur"**,Il suffit de sélectionner le pilote et de cliquer sur "Terminer". Dans notre application, le pilote sélectionné concerne une base Microsoft Access.

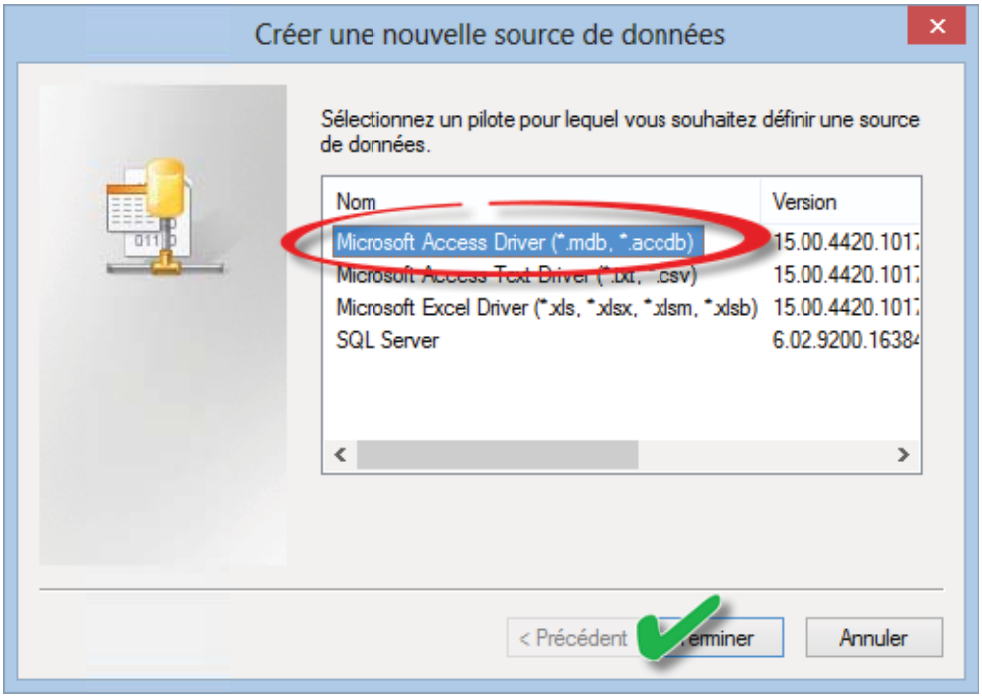

Figure VI.5 : Pilote de base Microsoft Access

a. Il suffit de saisir les informations nécessaire notamment le nom de la source de données et de sélectionner la base. Un clic sur "Ok" créé la source de données qui pourra alors être utilisée.

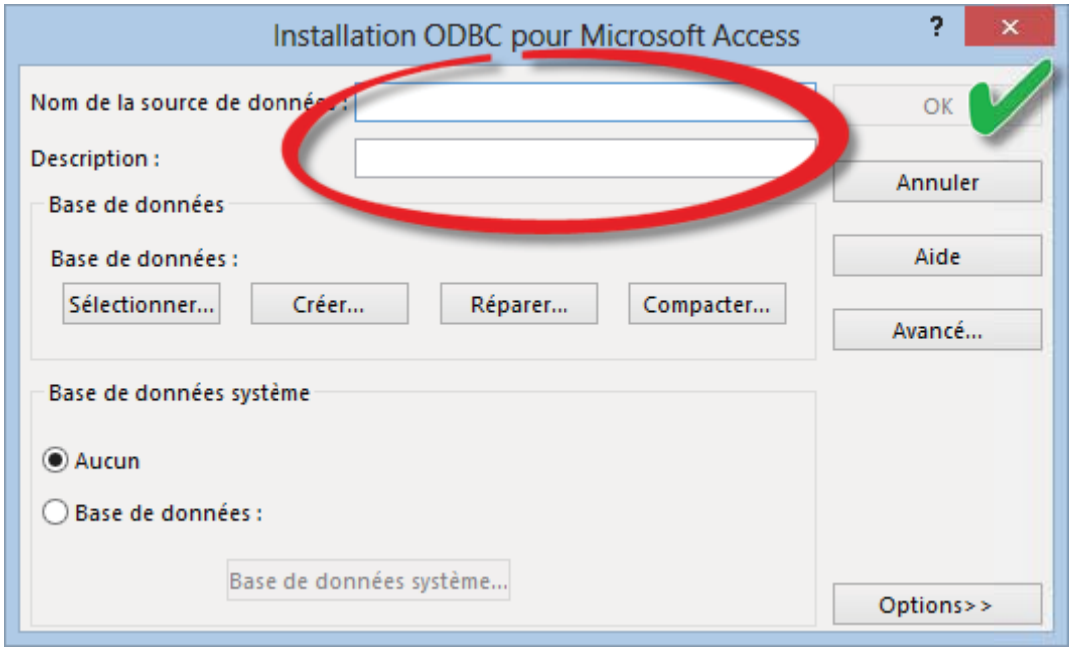

Figure VI.6 : informations nécessaire de la source de données et De la base.

4. Interfaces de l'application

Dans cette section nous présentons quelque fenêtre de notre application :

4.1. Interface Serveur :

Permet à l'utilisateur d'ajouter, supprimer ou connecter un serveur.

Figure IV.7 : interface Serveurs.

1. Bouton serveurs pour accéder à l'interface de configuration des serveurs.
2. Bouton Ajouter pour ajouter un nouveau serveur.
3. Bouton Connecter pour connecter à un serveur sélectionné dans la liste des serveurs.
4. Table contient la liste des serveurs.
5. Bouton Supprimer pour supprimer un serveur sélectionné.
6. Pour supprimer tout la liste des serveurs.

4.2. Interface Rechercher :

Permet de rechercher un fichier dans la base de données des pairs connecte ou télécharger.

Figure IV.8 : Interface Rechercher.

1. Bouton Rechercher pour accéder à l'interface Rechercher.

2. mot clé de la recherche.

3. Bouton pour lancer la recherche.

4. table contient la liste des fichiers trouve

5. Bouton Télécharger pour lancer le téléchargement d'1 fichier sélectionne.

4.3. Interface Télécharger :

Cette fenêtre permet à l'utilisateur de sélectionner un dossier pour enregistrer le fichier sélectionné pour télécharger.

Figure IV.9 : interface Télécharger.

4.4. Interface Partager :

Permet de partager un ou plusieurs fichiers sur le réseau.

Figure IV.10 : interface Partager.

1. Bouton Partager pour accéder à la fenêtre partager.
2. Bouton Parcourir pour accéder à l'emplacement de fichier partagé.
3. Bouton Partager pour ajouter le fichier sélectionner a la liste des fichiers partagé.
4. Table contient la liste des fichiers partagés.
5. Bouton rechercher pour rechercher dans la base de donnes des fichiers partagés.
6. Bouton supprimer pour supprimer un fichier sélectionné dans la base de données.
7. Bouton supprimer tous pour supprimer tout la liste des fichiers partagé.

4.5. Interface Chat :

Permet à l'utilisateur d'envoyer des messages, des appels vidéos ou joindre des fichiers avec les autres pairs connecte.

Figure IV.11 : interface Chat.

1. Bouton Chat pour lancer la fenêtre de chat.
2. Le pair en cour de chat.
3. La liste des amis connecte.
4. Le Bouton send message pour envoyer le message.
5. Bouton pour parcourir l'emplacement de fichier (vidéo, audio,…) pour l'envoyer.
6. Bouton pour lancer la Cam.
7. La liste des messages envoyés en chat.

5. Conclusion

Dans cette partie nous avons présenté l'environnement de développement de notre application. Ainsi qu'un ensemble d'interfaces (Imprime écran).

Conclusion générale

Conclusion générale

Dans ce mémoire de fin d'étude, nous sommes intéressés à la conception et à la réalisation d'une application de streaming peer- to- peer.

Au départ, une étude bibliographique des systèmes peer to peer nous a montré leur utilité et leur avantages qui ont accéléré leur adoptions. Ils ont été le résultat d'une évolution naturelle des architectures, conçues pour atteindre des objectifs spécifiques en les utilisant dans différents types d'applications et plus précisément dans le mode streaming pour les données audio/vidéo.

Un passage par l'étude de streaming nous a précisé leur avantage par rapport au téléchargement, il permet donc d'éviter l'attente due à la récupération d'un contenu avant de pouvoir le visionner.

L'élaboration de ce projet de fin d'études a permis d'avoir une idée claire sur la conception d'un prototype permettant d'effectuer le partage de données multimédias entre les différents peer.

Ce projet nous a donc permis d'appréhender les différents aspects que nous avons acquis au cours de notre formation. Tout d'abord le langage Java, la programmation orienté objet (POO) d'une façon générale ainsi que la manipulation d'une base de données Access. Effectivement nous avons réalisé une application réseaux de streaming peer to peer qui nous a permis de constater les nombreuses facettes de Java, les réseaux et leurs architectures.

D'un niveau plus que moyen au commencement de ce projet, nous avons maintenant acquis les concepts principaux du développement réseau. Nous avons amélioré notre maîtrise du système d'exploitation de part les nombreux problèmes logiciels rencontrés.

Bibliographie

[1] : Stephanos Androutsellis-Theotokis and Diomidis Spinellis, A survey of peer-to-peer content distribution technologies. ACM Computing Surveys (CSUR), December 2004.

[2] : B. Pourebrahimi, K. Bertels and S. Vassiliadis, A Survey of Peer-to-Peer Networks, Computer Engineering Laboratory, Netherlands. 2004.

[3] : Thomas CERQUEUS, Contributions au problème d'hétérogénéité sémantique dans les systèmes pair-à-pair, Thèse de Doctorat Université de Nantes, 2012.

[4] : C.Shirky, Peer-to-peer : "Harnessing the Power of Disruptive Technologies", chapter Listening to Napster, pp. 21–37, O'Reilly & Associates, 2001.

[5] : Skype, "http://www.skype.com/", 2014.

[6] : Gabrielle Feltin, Guillaume Doyen et Olivier Festor, Les protocoles Peer-to-Peer, leur utilisation et leur détection, LORIA - Campus Scientifique, Octobre 2003.

[7] : Houda HAFI, « Protocole pour la sécurité des réseaux sans fil peer to peer », Mémoire Présenté pour l'obtention du diplôme de : Magister en Informatique, Université Kasdi Merbah – Ouargla.

[8] : Mounir Tlili. Infrastructure P2P pour la Réplication et la Réconciliation des Données. Thèse de Doctorat Université de Nantes, 2003.

[9] : Ion Stoica, et al. Chord : A scalable peer-to-peer lookup service for internet applications. 2001 conference on Applications, ACM Press, 2001.

[10] : Bertrand Mathieu, A Network and QoE Aware P2P Video Streaming Solution, Orange Labs, 2008.

[11] : MOSTEFAOUI Soumia, « peer to peer streaming », Mémoire de fin d'études Pour l'obtention du diplôme d'ingénieur d'état en informatique, Ecole national Supérieure d'Informatique (ESI) Oued-Smar, Alger, 2008/2009.

[12] : Husam ALUSTWANI, « Interactivité et disponibilité des données dans les systèmes Multimédias Distribués », Thèse pour obtenir le garde de Docteur de l'université de Franche-Comté, spécialité : informatique, 2009.

[13] : Amine CHAOUCHE, « Gestion de la qualité de service pour le streaming de contenu audiovisuel sur les réseaux Pair-à-Pair », Mémoire Présenté pour l'obtention du diplôme de Magister, Filière : Informatique, École Nationale Supérieure d'Informatique, 2010.

[14] : T.Ahmed, “Transport Adaptatif et Contrôle de la Qualité des Services Vidéo sur les Réseaux IP Filaires, Sans-fil et sur les Architectures P2P”, HDR de l’université de BORDEAUX, novembre 2008.

[15] : IUT Rec. H.264/ISO IEC 14996-10 AVC, Advanced video coding for generic audiovisual services, 2003.

[16] : J.Reichel, H.Schwarz, T.Wiegand, G.Sullivan and M.Wien, Joint draft 9 of svc amendment, in Joint Video Team (JVT), JVT-V.201,Marrakech, Marocco, January 2007.

[17]: http://calculator.art.free.fr

[18]: http://www.commentcamarche.net

[19]: http://wapiti.telecom-lille1.eu.

Glossaire

P2P : Peer to Peer

IP : Internet Protocol

CPU : Central Processing Unit

SPoF : Single Point of Failure

VoD : Video On Demand

http : Hypertext Transfer Protocol

FTP : File Transfer Protocol

PPV : pay-per-view

QoS : quality of service

RTP : Real-Time Transfert Protocol

RTCP : Real Time Control Protocol

RSVP : Ressource reSerVation protocol

RTSP : Real Time Streaming Protocol

UDP : User Datagram Protocol

TCP : Transmission Control Protocol

UML : Unified Modeling Language

FIFO : First In First Out

OMG : Object Management Group

JDK : Java Développent Kit

SQL : Structured Query Language

QBE : Query by Example

JDK : Java Development Kit

JVM : Java Virtual Machine

XML : eXtensible Markup Language

SGBD : Système de Gestion de Base de Données

API : Application Programming Interface

Résumé

Actuellement, de nouveaux services de distribution de contenus audiovisuels et multimédia font leurs apparitions sur les réseaux. La Vidéo à la Demande (VoD) est considérée aujourd'hui comme étant un service de grande importance sur Internet. Après le partage de fichiers et la téléphonie IP, les applications de streaming p2p reçoivent un grand intérêt à la fois académique et industriel. Le but principal de cette technologie est de permettre aux nœuds de coopérer et d'échanger les contenus entre eux. Un grand nombre d'applications de streaming p2p sont actuellement utilisées, sur Internet, à la fois pour la vidéo à la demande et en temps réel comme les services de télévision IP, tels que PPStream, UUSee, SopCas, TvAnts et Joost. L'objectif de notre projet consiste au développement d'une application de streaming P2P permettant de diffuser des média audio et vidéo depuis un ordinateur vers un ou plusieurs, de partager des fichiers et permet à plusieurs personnes de discuter ensemble par l'envoie des messages.

Mot clés : Vidéo, streaming p2p, vidéo à la demande, RTP, JMF.

Abstract

Currently, new services of distribution of contents audio-visual and multimedia make their appearances on the networks. The Video on Demand (VoD) is considered among important services on Internet. After file sharing and IP telephony, p2p streaming applications are getting a great concern from both academia and industry. The main goal of this technology is to allow nodes to cooperate and exchange contents among themselves. A lot of p2p streaming applications are currently used, over the Internet, for both video on demand and real-time streaming services like IPTV such as PPStream, UUSee, SopCas, TVants and Joost. The objective of our project consists with the development of an application of streaming P2P making it possible to diffuse audio media and video since a computer towards one or more, to divide files and allows several people to discuss together by sends messages.

Key word: Video, streaming p2p, Video on Demand, RTP, JMF.