Un système de sauvegarde P2P sécurisé s'appuyant sur une architecture AAA

Houssem Jarraya, Maryline Laurent

Un système de sauvegarde P2P sécurisé s'appuyant sur une architecture AAA

LOgiciels-**R**éseaux	**Houssem Jarraya** **Maryline Laurent-Maknavicius**	08-002 LOR 2008

A SECURE DISTRIBUTED P2P BACKUP SYSTEM BASED ON AAA ARCHITECTURE

---oOo---

Abstract

A peer-to-peer (P2P) network is composed of a set of entities (called pairs) which share a set of resources and play simultaneously the roles of server and client. This model has been very successful since the beginning and has been considered as a real alternative to the classical client-server model. The distributed backup system is one of the new usages of this model. Their objective is to enable users to keep a copy of their data in the disk space of other users so restoration of them is possible after a hard disk crash, error of handling … However, the security, the selfish behaviour of a few peers and the quality of service are still challenging topics in this type of network.

In this report, we solved these problems in the « DisPairSe » project [1]. DisPairSe is a one-year project financially supported by institut TELECOM and including the following partners: TELECOM Bretagne, TELECOM & Management SudParis, et l'institut Eurecom. We proposed solutions to ensure the integrity and confidentiality of data stored in the P2P network (album photos, payrolls …) while respecting the anonymity of users. We defined mechanisms in order to control the access to the DisPairSe system and to ensure accounting reports. An AAA architecture is used that helps to support control and to count the disk space consumed or made available by the pair to detect selfish behaviours. We also proposed solutions to control the behaviour of P2P nodes and to provide a high quality of service to users.

Key words: peer-to-peer, integrity of data, confidentiality of data, anonymity, selfish behaviour.

Résumé

Un réseau P2P (pair à pair) est composé d'un ensemble d'entités (appelées pairs), partageant un ensemble de ressources, et jouant à la fois le rôle de serveur et de client. Dès son apparition, ce modèle a connu un grand succès et a été considéré comme une vraie alternative au modèle classique client-serveur. Parmi les nouvelles utilisations de ce modèle, on trouve les systèmes de sauvegarde distribuée qui permettent à un utilisateur de garder une copie de ses données dans les espaces de stockage d'autres utilisateurs afin de les restaurer en cas de crash de disque dur, d'erreurs de manipulation… Cependant, la sécurité, le comportement égoïste de quelques pairs et la qualité de service offerte restent toujours des défis dans ce type de réseau.

Dans ce rapport, nous nous sommes intéressés à résoudre l'ensemble de ces problèmes dans le cadre du projet DisPairSe [1], projet d'un an financé en 2007 par l'Institut Telecom et auquel participent les partenaires suivants : TELECOM Bretagne, TELECOM & Management SudParis, et l'institut Eurecom. Nous avons proposé des solutions pour assurer l'intégrité et la confidentialité des données sauvegardées dans le réseau P2P (album photos, bulletins de salaire,…) tout en respectant l'anonymat des utilisateurs. Nous avons défini des mécanismes permettant de contrôler l'accès au système DisPairSe et de comptabiliser, pour chaque pair, l'espace disque consommé et l'espace disque fourni afin d'éviter qu'un nœud ne puisse adopter un comportement égoïste. Pour cela, nous nous sommes appuyés sur une architecture AAA. Nous avons aussi proposé des solutions permettant de contrôler le comportement des différents nœuds du réseau P2P et d'assurer une bonne qualité de service aux usagers.

Mots-clés : pair-à-pair, intégrité des données, confidentialité des données, anonymat, comportement égoïste.

--oOo---

Houssem Jarraya
Stagiaire
TELECOM & Management SudParis - Département LOR
9 rue Charles Fourier 91011 Evry cedex
E-mail: Houssem.Jarraya@int-edu.eu

Maryline Laurent-Maknavicius
Professeur
TELECOM & Management SudParis - Département LOR
9 rue Charles Fourier 91011 Evry cedex
E-mail: Maryline.Maknavicius@int-edu.eu

TABLE DES MATIERES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Les réseaux pair à pair (P2P, de l'anglais "peer-to-peer") sont composés d'un ensemble d'entités partageant un ensemble de ressources, et jouant à la fois le rôle de serveur et de client. Ce modèle connait un très grand succès depuis la fin des années 90, grâce à l'utilisation des applications de partage de fichiers tels que : Napster, eMule, FastTrack,…

Actuellement, la recherche et l'industrie voient en ce modèle une vraie alternative au modèle classique client-serveur et contribuent à de nombreux travaux dans ce domaine. Parmi les nouvelles utilisations de ce modèle, on trouve le calcul distribué qui propose d'utiliser les machines connectées à l'Internet pour effectuer de petites portions d'un calcul colossal, et les systèmes de sauvegarde distribuée qui permettent à un utilisateur de sauvegarder une copie de ses données dans les autres pairs du réseau P2P afin de les récupérer en cas de perte de ses données locales ou en cas de crash de son disque dur.

Ce rapport, qui s'intéresse principalement à l'étude de la sécurité et l'introduction des services AAA (**A**uthentication, **A**uthorization, **A**ccounting) dans le système de sauvegarde distribuée « DisPairSe » [1], est organisé comme suit : le premier chapitre introduit le modèle P2P et présente les caractéristiques et le principe de fonctionnement du protocole P2P qui a été adopté dans le projet DisPairSe. Dans le deuxième chapitre, nous présentons la structure d'un fichier sauvegardé dans le réseau P2P et les traitements de sécurité à appliquer sur chaque type de donnée afin d'assurer sa confidentialité et son intégrité. Le troisième chapitre décrit les processus suivis par un utilisateur pour sauvegarder, modifier, restaurer ou supprimer un fichier du réseau P2P. Dans le quatrième chapitre, nous décrivons l'architecture que nous avons proposée pour le système DisPairSe et les techniques permettant d'introduire les services AAA. Enfin, le dernier chapitre présente les services que nous avons ajoutés pour contrôler le comportement des différents nœuds et la méthode proposée pour obliger les pairs à bien respecter les règles de fonctionnement du système et éviter les comportements malveillants.

I INTRODUCTION AUX RÉSEAUX P2P

1 Introduction

Les réseaux P2P sont caractérisés par le comportement dynamique de ses pairs, il est donc difficile d'obtenir une vue globale de l'ensemble des ressources disponibles et surtout de déterminer la localisation d'une ressource particulière. Pour répondre à ce problème, plusieurs modèles ont été envisagés.

Dans ce chapitre, nous présentons les différents niveaux de décentralisation des réseaux P2P. Puis, nous détaillons le modèle pur utilisant une table de hachage distribuée et nous décrivons particulièrement le principe de fonctionnement du protocole de routage « Pastry » qui a été adopté dans le projet DisPairSe. À la fin de ce chapitre, nous présentons quelques domaines d'application des réseaux P2P.

2 Différents niveaux de décentralisation

Les infrastructures P2P sont classifiées en fonction de leur degré de décentralisation, en deux [2] ou trois [3] sous-modèles. Nous considérons que le modèle P2P peut se diviser en trois sous-modèles qui sont les modèles pur, hybride et centralisé.

2.1 Le modèle centralisé

Ce modèle repose sur un serveur central auquel se connectent les pairs. Ce serveur maintient la liste des ressources disponibles et les paramètres réseaux (adresse IP et le numéro de port) des nœuds qui les hébergent. Par exemple, dans la figure 1 les ressources R1, R2, R3 et R4 sont hébergées respectivement par les postes P2, P4, P1 et P3, et toutes ces informations sont maintenues par le serveur central « SC ».

Quand un utilisateur désire accéder à une ressource, il interroge depuis le nœud-client le serveur central qui lui indique alors la liste des nœuds-serveurs sur lesquels il est susceptible de la trouver. Une fois les opérations de découverte et de localisation effectuées, une communication directe est en place entre le nœud-serveur et le nœud-client. C'est cette interaction directe entre les pairs qui différencie le modèle P2P centralisé du modèle client-serveur classique.

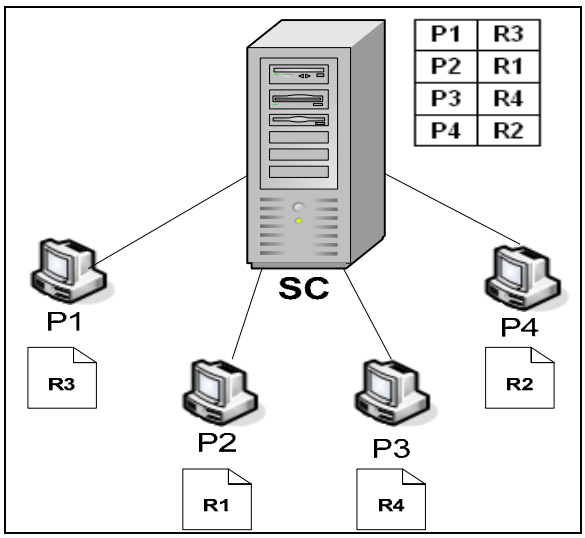

Figure 1. Le modèle centralisé

Le modèle centralisé est peu couteux puisque l'accès à une ressource particulière ne fait intervenir que deux machines : le serveur central pour la phase de découverte et le nœud-serveur pour l'accès à la ressource. Par contre, son principal inconvénient est sa fragilité puisqu'il suffit que le serveur central soit inaccessible (panne, surcharge,…) pour que tout le service P2P soit indisponible.
Ce modèle est utilisé dans des applications telles que : Napster, Skype, ….

2.2 Le modèle hybride

Le modèle P2P hybride représente un modèle P2P dans lequel certains pairs, appelés "super pair" ou "super nœud", jouent un rôle supplémentaire. Ces pairs particuliers sont, généralement les nœuds possédant une forte puissance de calcul et une importante bande passante, et ils se chargent de gérer un groupe de postes. Par exemple, dans la figure 2 le "super pair" P1 est responsable des postes P2, P3 et P4.

Le rôle d'un "super pair" dans son groupe est équivalent à celui du serveur central dans le modèle centralisé. En effet, il maintient la liste des ressources hébergées par les membres de son groupe et il se charge, en plus, de leur résoudre toutes les requêtes de recherche. Dans la figure 2, le "super pair" P1 ne maintient que la liste des ressources hébergées par les postes P1, P2, P3 et P4, cette liste est constituée de R1 et R2.

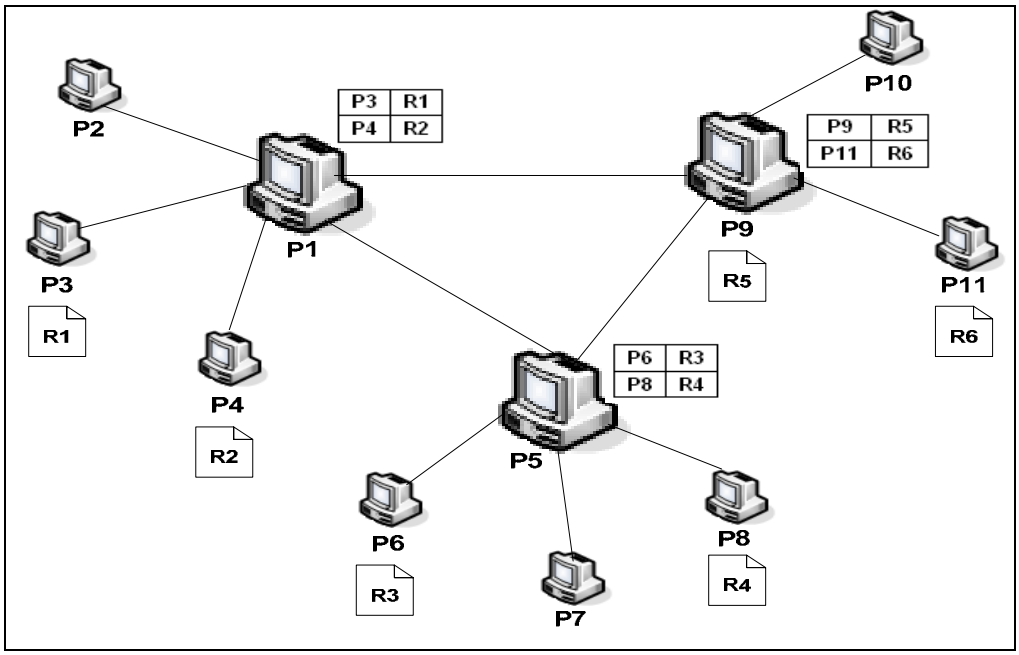

Figure 2. Le modèle hybride

Pour accéder au réseau P2P, un nœud doit se connecter à un "super pair" auprès duquel il déclare la liste des ressources qu'il veut partager. Une fois connecté, un pair peut accéder à toutes les ressources du système en envoyant, tout simplement, ses requêtes à son "super pair". Ce dernier se chargera de déterminer la localisation des ressources demandées, soit en accédant à ces informations locales (la liste des ressources hébergées par les postes qu'il gère), soit en diffusant la requête aux autres "super pair".
L'avantage de ce modèle est qu'il ne se base pas sur un nœud central et donc il est plus robuste aux attaques de déni de service. En contrepartie, il est plus complexe à mettre en place du fait de la hiérarchie qui existe entre les nœuds.

Ce modèle est utilisé dans des applications telles que Kazaa, …

Remarque : Certaines classifications des architectures P2P considèrent les modèles hybrides et centralisés comme identiques : le modèle centralisé n'étant qu'un modèle hybride contenant un seul super-pair.

2.3 Le modèle pur

Dans ce modèle, les pairs sont strictement équivalents et donc la suppression de n'importe lequel des nœuds présents n'affecte pas le service offert (cf. Figure 3). Par contre, l'absence d'un nœud particulier (serveur central ou "super nœud") ayant une vue globale de l'ensemble des ressources hébergées dans le réseau P2P crée le problème suivant : Comment un nœud peut-il déterminer la localisation d'une telle ressource ?
Pour répondre à ce problème, deux solutions ont été proposées : la première utilise une méthode d'inondation et la deuxième se base sur les tables de hachage distribuées (DHT).

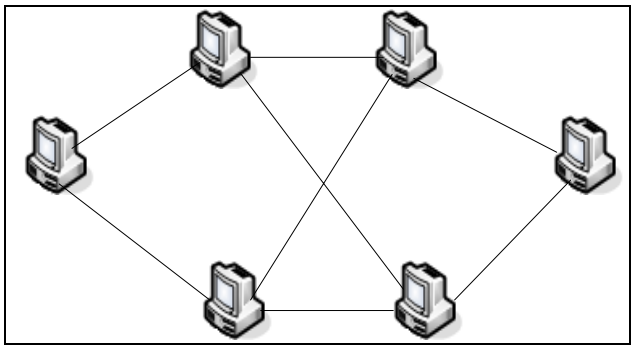

Figure 3. Le modèle pur

- **Recherche par inondation**

Dans cette solution, chaque pair est responsable de l'indexation de ses propres ressources. Ainsi, lors d'une phase de recherche, une requête sera transmise de proche en proche jusqu'à atteindre l'hébergeur de la ressource demandée. Afin que ces requêtes ne circulent pas indéfiniment sur le réseau, le système leur associe une durée de vie limitée exprimée en termes de nombre maximal d'ordinateurs à atteindre (*TTL* pour *Time To Live*).
Ce fonctionnement totalement décentralisé du système P2P possède des inconvénients majeurs. Tout d'abord, les recherches s'avèrent peu efficaces puisque tous les nœuds du réseau ne peuvent être interrogés, ce qui peut aboutir à l'échec d'une recherche bien que la ressource demandée soit disponible dans le réseau P2P. Ensuite il faut relever le nombre de messages envoyés dans le réseau qui est extrêmement important.
Cette méthode est utilisée dans le système Gnutella.

- **Les tables de hachages distribuées (DHT)**

Une table de hachages distribuée est une structure de données fournissant un service de recherche puissant aux utilisateurs. Son idée de base est de répartir l'ensemble des informations nécessaires pour la localisation des ressources sur tous les nœuds de telle façon que la déconnexion d'un pair n'entraîne pas de pertes importantes pour l'ensemble du réseau.
Dans le paragraphe suivant, nous détaillons les caractéristiques générales d'un système P2P utilisant une DHT et par la suite, nous nous concentrons sur le protocole « Pastry » qui a été adopté dans le projet DisPairSe.

3 Les tables de hachage distribuées

Dans cette section, nous présentons les principes généraux, applicables à l'ensemble des solutions basées sur une DHT, ainsi que leurs propriétés.

3.1 Principe général

Une table de hachage distribuée est une structure de données qui associe une clé à chaque ressource. Chaque clé de la table est le résultat d'une fonction de hachage appliquée à un élément de la ressource, comme par exemples son nom ou son contenu. L'utilisation d'une bonne fonction de hachage garantit que, pour deux ressources différentes, les clés générées le seront aussi. Cette unicité de clé permet d'identifier et de retrouver de manière fiable la ressource à laquelle elle est associée. Les fonctions de hachage fréquemment utilisées sont : SHA-1 [4] et MD5 [5].

L'innovation apportée par les tables de hachage distribuées est la manière de distribuer les différentes données de localisation sur l'ensemble des nœuds du système, car si dans un modèle centralisé, il est simple de parcourir une table pour rechercher une donnée, dans le cas d'un modèle P2P pur, le problème de recherche d'une ressource se pose. Cette innovation consiste à attribuer pour chaque pair un identifiant qui est encore le résultat d'une fonction de hachage appliquée, par exemple, sur son adresse IP ou son nom de machine. Ensuite, la table est distribuée de manière à ce que les clés des ressources soient maintenues par les pairs dont l'identifiant est le plus proche, d'après une fonction de distance donnée.

Pour bien illustrer ce principe, prenons l'exemple présenté dans la figure 4. On peut remarquer que le pair P1 héberge la partie de la table correspondant à la ressource R1 qui possède le même identifiant numérique, à savoir 1. De même, P2 héberge l'entrée de la table pour R2, et ainsi de suite. On constate aussi, que chaque pair hébergeant une partie de la table de hachage, ne possède qu'une référence vers une ressource et non la ressource elle-même. Par exemple, le pair P3 héberge physiquement la ressource R4 et possède un pointeur vers P1 qui héberge R3, la ressource qui possède le même identifiant. Ainsi, découvrir une ressource particulière revient à découvrir le pair d'identifiant le plus proche de celle-ci. Ce dernier contient une référence sur le nœud hébergeant réellement la ressource.

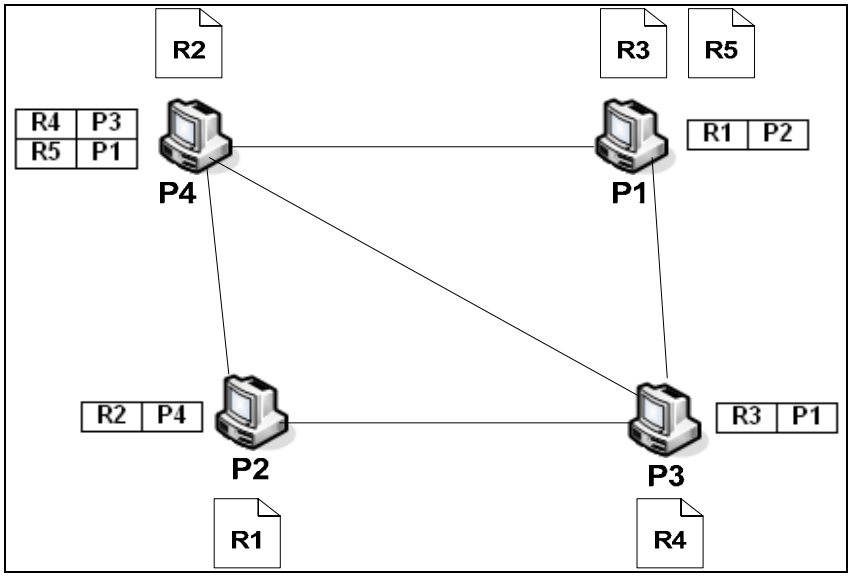

Figure 4. La découverte de ressources en utilisant une table de hachage distribuée

3.2 Propriétés

Les DHTs présentent de bonnes propriétés qui sont liées à l'utilisation d'un modèle P2P pur : aucun pair ne présente de rôle particulier et chacun agit de manière strictement équivalente.

Parmi ces propriétés [6], on peut citer :

✓ **Les performances :** Pour la plupart des DHTs, le nombre de sauts nécessaires à l'acheminement d'une requête est O (logB(N)), où N est le nombre de nœuds et B est la base des identifiants pour les pairs et les ressources. Ceci montre que les réseaux P2P utilisant une table de hachage distribuée sont très performants. Par exemple, dans un réseau de 10^6 pairs utilisant une base d'identifiants hexadécimale, la longueur moyenne d'une requête est de 5 sauts.

✓ **Le passage à l'échelle :** Même dans le cas d'un réseau comptant un grand nombre de nœuds, une solution basée sur une DHT reste encore performante. En effet, le nombre de sauts nécessaires au routage des requêtes reste toujours petit. En plus la distribution des informations de routage sur l'ensemble des pairs garantit que la portion de la table de hachage maintenue par chaque nœud est toujours de taille raisonnable.

✓ **L'équilibre de la charge et du trafic :** L'utilisation de fonctions de hachage pseudo-aléatoires (MD5, SHA-1) pour calculer les identifiants de pairs et les clés de ressources permet de créer un réseau P2P équilibré, c'est-à-dire où les pairs ont statistiquement en charge une part égale de ressources à référencer et donc de trafic.

✓ **La tolérance au départ de nœuds :** L'absence d'un serveur central ou d'un "super nœud" donne aux réseaux P2P utilisant une DHT, une très bonne tolérance face à des suppressions aléatoires de nœuds. De plus, la plupart de ces réseaux introduisent des mécanismes de redondance permettant de dupliquer les informations contenues dans la table de hachage sur plusieurs nœuds et ceci afin d'éviter l'inaccessibilité d'une ressource disponible dans le réseau.

Chaque réseau P2P utilisant une table de hachage distribuée, doit définir en plus un protocole de routage assurant l'acheminement des requêtes de recherche aux bonnes destinations. Parmi les protocoles qui ont été définis, nous citons : Chord [7], Viceroy [8], Pastry [9]. Dans le paragraphe suivant nous détaillons le fonctionnement du protocole Pastry qui a été adopté dans le projet DisPairSe.

4 Protocole Pastry

Pastry est un protocole de routage et de localisation dans un réseau P2P basé sur une table de hachage distribuée. Chaque nœud membre du réseau est muni d'un identifiant unique de 128 bits qui est le résultat d'une fonction de hachage appliquée par exemple à son adresse IP. Les deux opérations de base définies dans Pastry, comme dans les autres protocoles P2P, sont :

- **store (key, value)** : permet de sauvegarder la ressource « value » dans le pair dont l'identifiant est le plus proche numériquement de sa clé « key ».

- **lookup (key)** : permet de récupérer la ressource associée à la clé « key ». Cette requête sera routée vers le pair dont l'identifiant est le plus proche numériquement de la clé « Key ». Soit ce dernier sauvegarde réellement la ressource demandée qui sera donc transmise directement au nœud-client, soit il contient les paramètres de localisation (l'adresse IP et le numéro de port) de son hébergeur.

4.1 La table d'état

Chaque pair dans Pastry maintient une table d'état contant trois catégories d'entrées.

Dans la première catégorie, on trouve l'ensemble des voisins virtuels du pair considéré en termes de distance numérique entre les identifiants des nœuds. Cet ensemble est subdivisé en deux sous-parties : l'une contenant les identifiants numériquement plus petits, l'autre, les identifiants numériquement plus grands.

La seconde catégorie est la table de routage du pair. Cette table est un tableau bidimensionnel dont les colonnes représentent un des éléments de la base choisie (de 0 à F pour l'hexadécimale) et les lignes, la taille du préfixe commun entre l'identifiant courant et celui considéré. Plus formellement, la cellule définie par (l=i, c= j) contient l'identifiant d'un pair possédant un préfixe commun de i digits exactement avec l'identifiant de référence et dont le (i+1)ème digit est égal au (j)ème digit de l'alphabet utilisé. Dans le cas où il n'y a pas un identifiant accomplissant ces deux conditions, la cellule considérée reste vide.

La troisième catégorie de la table d'état contient l'ensemble des voisins réels du nœud. La métrique choisie pour évaluer la distance réelle entre pairs est le nombre de sauts IP.

La figure 5 présente un exemple simplifié de table d'état d'un nœud Pastry. Dans cet exemple, la longueur d'un identifiant est de 16 bits représentés en base 4 (les digits permis sont 0, 1, 2 et 3). Par exemple, la deuxième colonne de la cinquième ligne (soit l = 4) possède une entrée pour un pair possédant un préfixe commun de 4 digits, et un cinquième digit égal à 1.

Pair 10233102

Voisins virtuels

Les identifiants numériquement plus petits que 10233102

10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Les identifiants numériquement plus grands que 10233102

Table de routage

02212102	1	22301203	31203203
0	11301233	12230203	13021022
10031203	10132102	2	10323302
10200230	10211302	10222302	3
10230322	10231000	10232121	3
10233001	1	10233232	
0		10233120	
		2	

Voisins réels

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Figure 5. La table d'état d'un nœud Pastry d'identifiant 10233102 (base 4) [9]

4.2 Le routage

Le principe du routage de Pastry est le suivant : lorsqu'une requête munie d'une clé K arrive dans un pair n, celui-ci vérifie tout d'abord si la clé est comprise dans l'ensemble de ses voisins virtuels. Si c'est le cas, la requête est routée vers son voisin virtuel qui a l'identifiant numérique le plus proche de la clé K. Sinon, le pair n détermine m, le nombre de chiffres communs à son

identifiant et à K, consulte sa table de routage pour trouver un pair qui présente m+1 chiffres en commun et lui fait suivre la requête. Si aucun pair ne correspond, le pair n cherche, dans toute sa table d'états, un autre pair ayant m chiffres en commun et dont l'identifiant est le plus proche de K.
Le message est arrivé à destination lorsqu'aucune entrée de la table d'état ne permet de se rapprocher plus de la clé K.

4.3 Entrée d'un nœud dans le réseau

Lorsqu'un nœud désire entrer dans le réseau, il envoie, à un pair d'entrée, une requête d'entrée ayant comme destination son propre identifiant. Le pair d'entrée transmet à l'émetteur de la requête sa table d'état, si nécessaire met à jour sa propre table et route ensuite la requête normalement.

Chaque pair qui reçoit par la suite la requête d'entrée fait de même. Finalement, le nœud voulant entrer dans le réseau P2P crée sa table d'état en fonction des réponses qu'il a reçues et la distribue à tous les nœuds contenus dans sa table d'état. Cette distribution finale de l'état d'un nœud entrant permet aux nœuds déjà membres du réseau de mettre à jour leur table d'état sans besoin d'émettre une requête [10].

5 Domaine d'applications des réseaux P2P

Pour le grand public, le terme « peer-to-peer » peut se confondre avec les systèmes de partage de fichiers tels que : eMule, Bittorrent, … mais, en réalité, les réseaux P2P sont utilisés dans plusieurs autres domaines comme :

- **Le calcul distribué ou Grid Computing**

Un réseau P2P peut contenir des milliers d'ordinateurs disposant chacun de ressources limitées. Mais en "additionnant" les ressources de chacun, on obtient de grandes performances qui peuvent être exploitées pour effectuer de gros calculs tels que l'inversion d'une matrice ou la multiplication de deux matrices.

- **La diffusion des données**

Actuellement, pour diffuser une donnée (telle qu'une vidéo) à plusieurs utilisateurs, il faut non seulement un serveur très performant, mais également une très grande bande passante : plus le nombre d'utilisateurs est important, plus la bande passante doit être importante. Par contre, si on utilise un réseau P2P où chaque pair peut jouer le rôle d'un serveur et diffuser donc la donnée aux autres pairs, il est possible de toucher beaucoup plus d'utilisateurs et sans aucune contrainte sur les ressources disponibles.

- **Le travail collaboratif**

Ce type d'applications permet à des utilisateurs de bénéficier d'un service sans utiliser de serveur central.
Le P2P dans DisPairSe fait partie de ce type de réseau P2P puisqu'il permet à un pair de sauvegarder une copie de ses données dans l'espace disque des autre pairs afin qu'il les récupère en cas de perte de ses données locales. Il s'agit donc de la création d'un système de restauration distribué.

6 Conclusion

Les réseaux P2P se divisent, selon leurs niveaux de décentralisation, en trois sous-modèles qui sont : les modèles centralisé, hybride et pur. La solution adoptée dans le projet DisPairSe se

base sur un modèle pur utilisant une table de hachage distribuée et plus précisément sur l'utilisation du protocole de routage « Pastry ».
Après la présentation des caractéristiques du protocole Pastry et la description de son principe de fonctionnement, nous décrivons dans le chapitre suivant les solutions que nous avons proposées pour assurer l'intégrité et la confidentialité des données sauvegardées dans le réseau P2P.

II SECURISATION DU P2P DANS LE PROJET DISPAIRSE

1 Introduction

Contrairement à un modèle client/serveur ou à un réseau local, un réseau P2P est constitué d'un ensemble de nœuds n'ayant aucune relation de confiance entre eux et dont les uns peuvent avoir des comportements malveillants.
Dans ce chapitre, nous présentons les propriétés de sécurité à garantir dans un système de sauvegarde distribuée, puis nous détaillons nos propositions pour conserver ces propriétés dans le système DisPairSe. La plupart de ces solutions sont basées sur les caractéristiques d'un réseau basé sur une table de hachage distribuée.

2 Propriétés de sécurité

Il est important qu'un système de sauvegarde distribuée garantisse les propriétés suivantes :

- les données d'un utilisateur ne doivent être compréhensibles que des utilisateurs autorisés (confidentialité).
- toute corruption de données sauvegardées, qu'elle soit intentionnelle ou non (faute logicielle ou matérielle sur la machine stockant les données), doit pouvoir être détectée par son propriétaire lors de la restauration (intégrité).
- les données d'un utilisateur doivent être sauvegardées dans les autres pairs du réseau P2P sans aucun renseignement sur son identité (anonymat).

Ne permettre la vérification de l'intégrité d'un fichier que par son propriétaire et lors de la phase de restauration ne permet pas de garantir une bonne qualité de service du système P2P dans DisPairSe. En effet, il est possible qu'un fichier en cours de sauvegarde dans le P2P soit modifié, lors de la phase de routage, par un nœud malicieux. Ainsi, sa copie de secours sauvegardée dans le réseau P2P sera incohérente et donc inutile. Par exemple, dans la figure 6, la donnée (X = 4) a été modifiée par le nœud (N3), et malgré cela, elle a été sauvegardée dans le nœud (N5). Ce type de comportement malveillant affecte la fiabilité du système et induit la perte de ressources du système P2P (espace disque, bande passante, …).

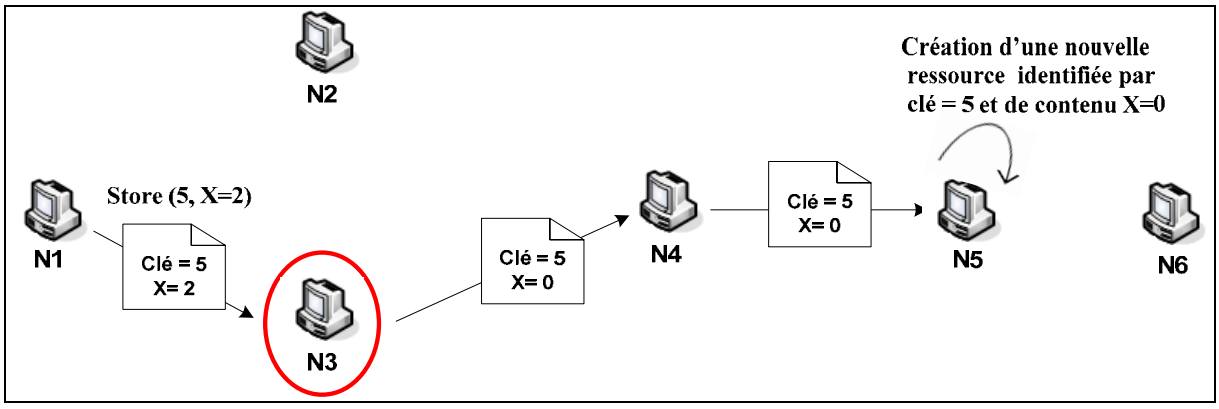

Figure 6. Exemple de comportement malveillant : modification du contenu d'une ressource

Afin de résoudre ce problème, nous devons permettre aux nœuds de sauvegarde de vérifier l'intégrité des données des autres utilisateurs et sans aucune connaissance particulière sur leur identité. De cette façon, nous évitons la sauvegarde de données incohérentes dans le réseau P2P et

nous pouvons, par la suite, diminuer le niveau de fiabilité de chaque nœud ayant sauvegardé une donnée incohérente du fait de la non vérification de son intégrité.

3 Structure d'un fichier sauvegardé dans le réseau P2P

Avant de présenter les mécanismes de sécurité que nous avons proposés pour protéger les données d'un utilisateur, nous définissons dans ce paragraphe la structure d'un fichier sauvegardé dans le réseau P2P.

Tout d'abord, chaque fichier sauvegardé dans le réseau est fragmenté en fragments. Cette fragmentation a pour but de réduire la duplication inutile des données dans l'espace (c'est-à-dire entre des fichiers appartenant à différents utilisateurs) et dans le temps (entre les différentes versions d'un même fichier). Elle permet, aussi, d'assurer une répartition homogène des charges entre les différents pairs du réseau P2P. En effet, la création de fragments de même longueur et l'utilisation d'une bonne fonction de hachage pour le calcul des clés DHT garantissent une distribution uniforme des données entre les différents pairs.

Chaque fichier est formé de deux types de fragments (comme dans le système pStore [11]) : un ensemble de fragments FB (File Block) contenant les informations réelles du fichier et un fragment de description FBL (File Block List). Ce dernier fragment, contient la liste des différents fragments FB de chaque version du fichier (cf. Figure 7). Pour chaque fragment FB, nous précisons sa clé DHT, sa clé de chiffrement, sa longueur et son offset par rapport au début du fichier. Ainsi, à partir du fragment FBL, l'utilisateur peut récupérer, lors de la phase de restauration, n'importe quelle version de son fichier.

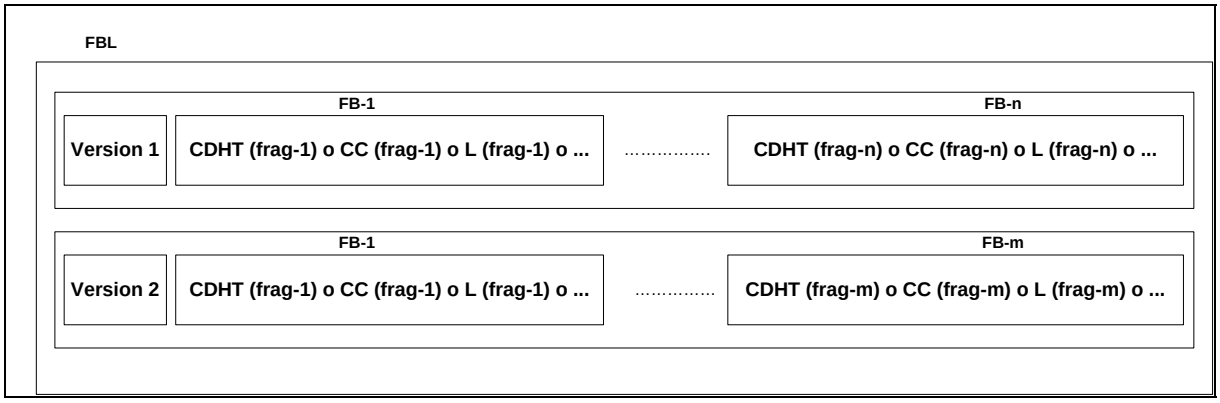

Figure 7. Structure d'un fragment FBL

CDHT : la clé DHT du fragment
CC : la clé de chiffrement du fragment
L : la longueur du fragment
o : opération de concaténation

Le fragment FBL peut contenir des informations supplémentaires sur le fichier telles que : son nom, son chemin, sa date de création,…

4 Protection des fragments FB

Dans ce paragraphe, nous présentons nos solutions pour assurer l'intégrité et la confidentialité des fragments FB.

4.1 Confidentialité

Pour assurer la confidentialité des fragments FB, nous pouvons utiliser la technique du chiffrement convergent *« Convergent Encryption »* [12]. Cette technique consiste à chiffrer un bloc de données par le condensé de son contenu (cf. Figure 8) et elle est utilisée par plusieurs systèmes de sauvegarde distribués tels que : pStore [11], ABS [13],… .

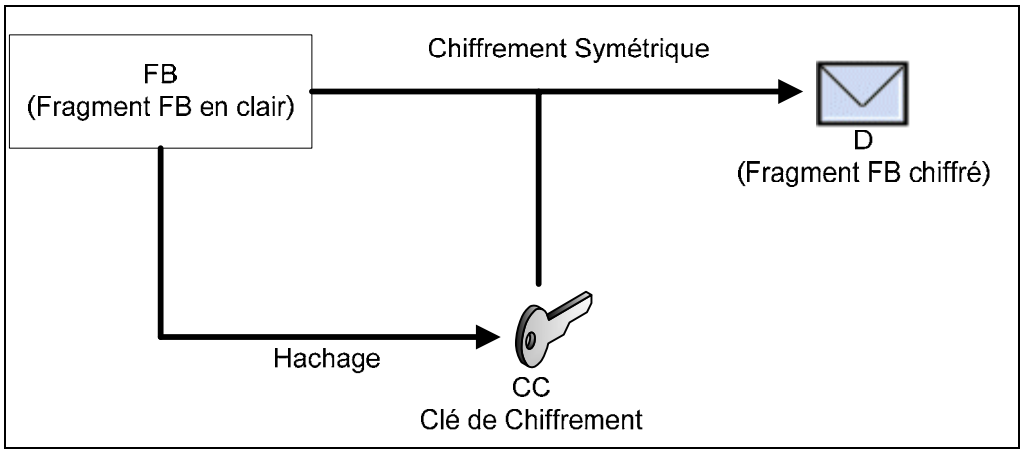

Figure 8. Confidentialité des fragments FB par la technique du chiffrement convergent

FB : fragment FB en clair
D : fragment chiffré (les données à sauvegarder dans le système)

L'application de la technique *« Convergent Encryption »* sur les fragments de données offre les propriétés suivantes :

✓ conservation d'identité : une identité au niveau des fragments en clair sera conservée au niveau des fragments chiffrés à cause de l'utilisation de la même clé de chiffrement (cf. Figure 9). Ainsi, nous associons pour un ensemble de fragments (FB) identiques, appartenant à différents utilisateurs et différents fichiers, un seul fragment chiffré (une seule nouvelle ressource). La sauvegarde de cette nouvelle ressource une seule fois dans le réseau P2P, permet d'optimiser l'utilisation des ressources du système (l'ensemble des espaces disques fournis par les différents pairs).

✓ utilisation des fonctions de chiffrement "légères" (symétriques) telles que 3DES ou AES, en vue de réduire la charge d'un nœud en termes de nombre d'opérations à exécuter. Cette propriété est indispensable pour les pairs de faible puissance (PDA, …) et elle permet en plus, d'accélérer les processus de sauvegarde et de restauration d'un fichier.

✓ utilisation d'une clef de chiffrement dépendant du contenu et donc ne pouvant être connue que des utilisateurs qui ont déjà une copie du fragment en clair. Ainsi, il n'y a aucun moyen, pour les nœuds malicieux, de déterminer la clé de chiffrement d'un fragment FB appartenant aux autres nœuds.

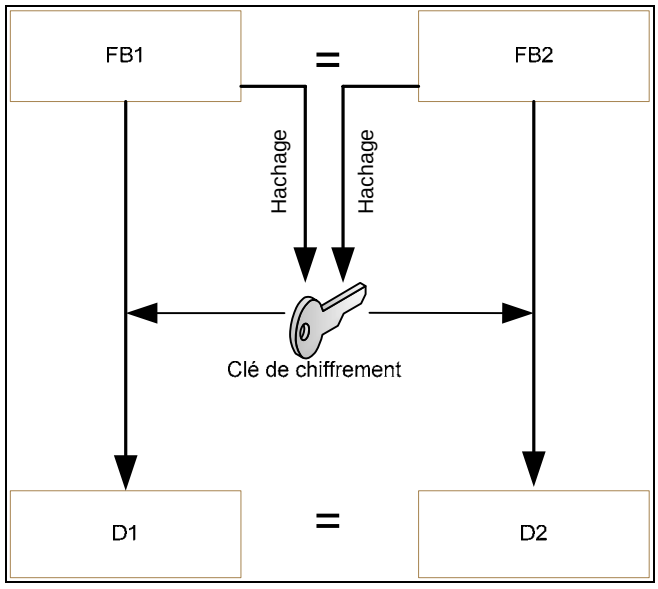

Figure 9. Conservation d'identité

L'utilisateur doit garder la clé de chiffrement (CC) dans un endroit sécurisé et sûr pour pouvoir par la suite la récupérer pour déchiffrer le fragment.

4.2 Intégrité

La technique classique utilisée pour assurer l'intégrité d'un message est la signature électronique dont le principe est le suivant : appliquer une fonction de hachage sur le message puis chiffrer son condensé par une clé (généralement la clé privée de l'émetteur). Cette technique exige un calcul énorme, à cause de l'utilisation d'une fonction de hachage et d'une fonction de chiffrement (généralement asymétrique), et donc elle ne peut pas être adoptée dans un réseau P2P où les nœuds sont de types variés et de performances inégales.

Les réseaux P2P de type DHT tels que Pastry associent à chaque ressource une clé qui permet de déterminer le nœud dans lequel la ressource sera sauvegardée. Cette clé DHT (CDHT) peut aussi être exploitée pour assurer l'intégrité du fragment FB en adoptant le principe suivant : la clé DHT d'un fragment FB est le condensé de son contenu (cf. Figure 10).

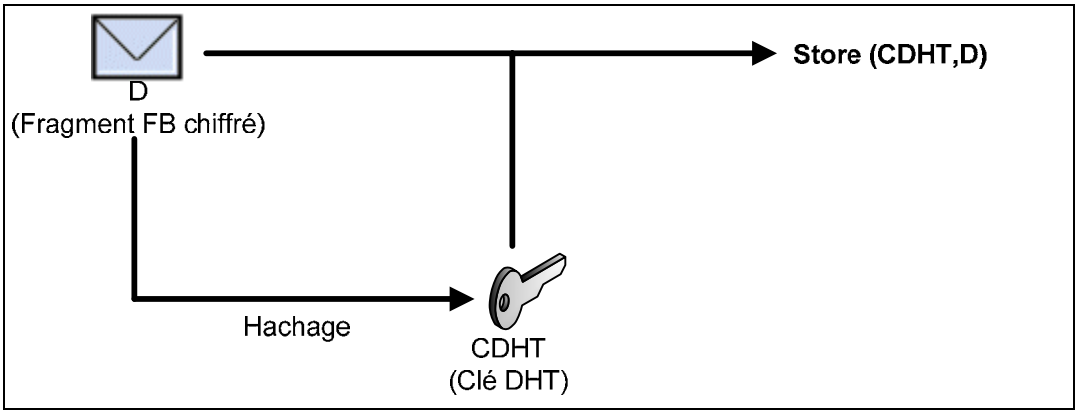

Figure 10. Intégrité des fragments FB

Ainsi, pour vérifier l'intégrité d'un fragment FB, il suffit de vérifier que son condensé est équivalent à sa clé DHT. Les avantages de cette méthode sont les suivants :

o l'intégrité du fragment est garantie sans besoin d'utiliser de clés supplémentaires (par exemple associer un couple de clés : clé publique et clé privée pour chaque nœud).

o tout nœud peut vérifier l'intégrité d'un fragment FB, ce qui évite la sauvegarde inutile des fragments incohérents dans le réseau P2P. En effet, lors de l'acquisition d'un nouveau fragment FB, le nœud de sauvegarde vérifie, tout d'abord, si cette nouvelle ressource est cohérente. Si c'est le cas, le fragment sera sauvegardé dans le nœud. Sinon, il sera supprimé du réseau.

o la rapidité : l'intégrité du fragment est garantie sans aucun traitement supplémentaire du côté de l'émetteur puisque l'opération de hachage est déjà nécessaire pour associer une clé DHT au fragment.

Cette solution ne permet de détecter que les modifications effectuées sur le contenu d'une ressource. En effet, si un nœud malicieux modifie à la fois le contenu et la clé d'un fragment FB en respectant toujours le mécanisme décrit dans la figure 10, le nouveau fragment créé sera accepté par son nœud de sauvegarde et donc sauvegardé dans le réseau P2P même s'il est erroné (cf. Figure 11). Ce type d'attaque sera traité dans le chapitre suivant (Paragraphe 2).

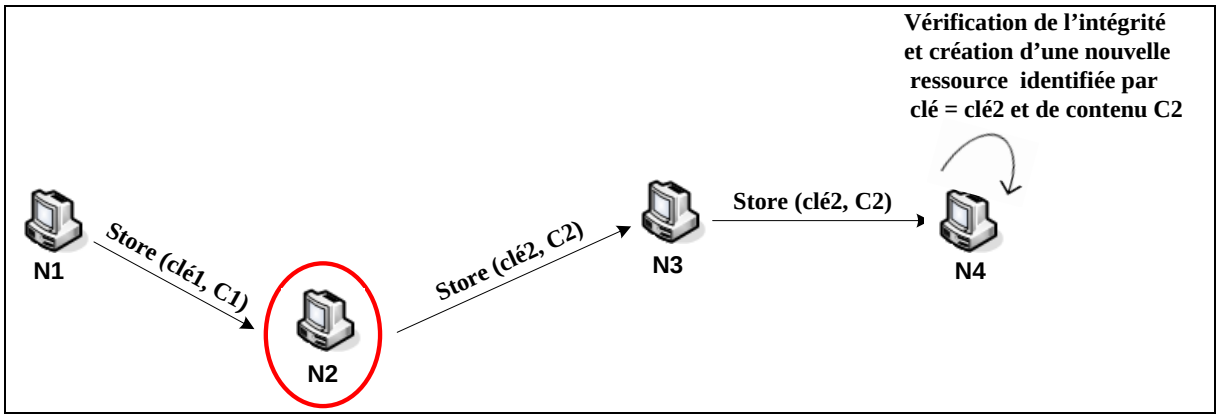

Figure 11. Modification du contenu et de la clé d'un fragment FB

$$\begin{cases} clé1 = H(C1) \\ clé2 = H(C2) \end{cases}$$

4.3 Solution complète

Avant de sauvegarder un de ses fragments FB dans le réseau P2P, un nœud doit (cf. Figure 12) :

1- calculer sa clé de chiffrement qui est le condensé du fragment en clair

2- chiffrer le fragment FB en clair en utilisant un algorithme de chiffrement symétrique et la clé déjà calculée

3- calculer la clé DHT qui est le condensé du fragment FB chiffré

4- envoyer une requête de sauvegarde Store (clé, données)

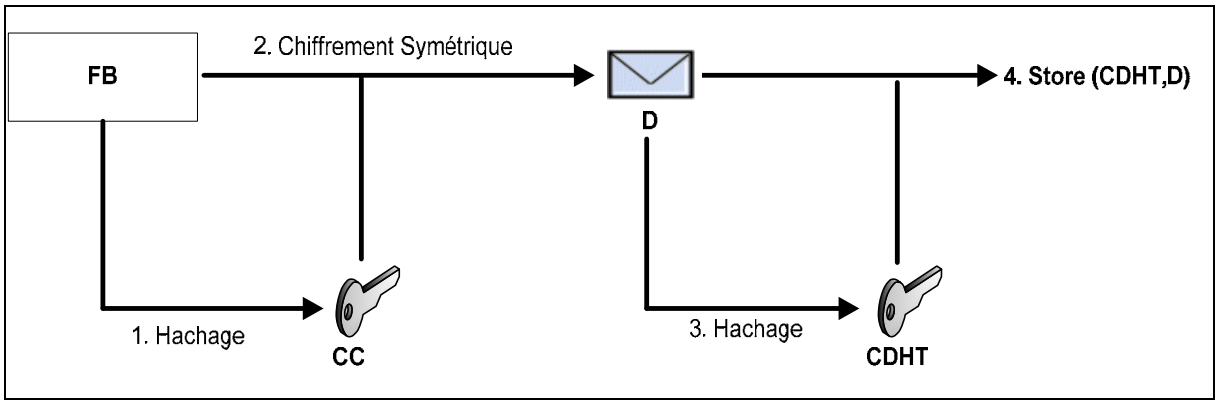

Figure 12. Traitement de sécurité appliqué par un nœud client sur un fragment FB

Cette méthode possède les avantages suivants :

- optimisation de l'espace disque utilisé : pour des fragments de données identiques, nous allons leur attribuer la même clé DHT et donc les fragments seront enregistrés dans le même nœud et une seule fois. Ceci permet d'éviter la duplication inutile des données dans l'espace (entre plusieurs utilisateurs) et dans le temps (entre plusieurs versions du même fichier).
- anonymat : les données d'un utilisateur seront enregistrées dans des nœuds répartis dans le réseau sans aucune indication sur son identité.
- simplicité et rapidité : le chiffrement est assuré par une méthode symétrique et l'intégrité est garantie uniquement à travers une fonction de hachage (qui est dans tous les cas nécessaire au calcul de la clé DHT du fragment).

Ce processus peut être résumé par le schéma suivant :

CC = H (FB)
D = ECC (FB)
CDHT = H (D)
Store (CDHT, D)

FB : le fragment FB en clair
H : fonction de hachage
E : méthode de chiffrement symétrique (cc est la clé de chiffrement)

5 Protection des fragments FBL

Lors de la phase de restauration d'un fichier, le fragment le plus important est son fragment FBL : c'est le point d'entrée pour récupérer le fichier tout entier. Ainsi, il faut être prudent dans le calcul de sa clé DHT (pour éviter les collisions) et il faut garantir en plus sa confidentialité et son intégrité.

Contrairement à un fragment FB, le contenu d'un fragment FBL est modifié après chaque mise à jour du fichier, et si nous le protégeons en utilisant le même processus décrit à la figure 12, nous aurons la création d'une nouvelle clé DHT et une nouvelle clé de chiffrement après chaque modification du fichier (ces deux clés sont calculées à partir du contenu du fragment). Pour résoudre ce problème, nous pouvons adopter la solution proposée dans le système pStore et qui consiste à déterminer la clé DHT d'un fragment FBL à partir de son nom et son chemin au lieu de son contenu.

5.1 Calcul de la clé DHT

Si plusieurs utilisateurs sauvegardent le même fichier (même contenu, même nom, …) dans le réseau P2P, ils partagent donc les mêmes fragments FB d'après le processus décrit à la figure 12. Supposons qu'ils partagent encore le même fragment FBL : quand un utilisateur met à jour son fichier, il va donc modifier le fragment FBL partagé et quand un autre utilisateur récupère ce fichier commun après le crash de son disque dur, il va récupérer en plus les modifications effectuées par les autres utilisateurs. Pour cela il faut éviter le partage des fragments FBL entre différents pairs, ceci peut être garanti par l'introduction d'une information relative à l'utilisateur dans la formule de calcul des clés DHT de ces fragments FBL. Dans le système pStore, cette information correspond à la clé privée de l'utilisateur, mais de façon générale elle peut consister en n'importe quelle information propre et secrète (mot de passe, f (clé privée), f (mot de passe), …). Dans la suite du rapport, cette information secrète sera nommée le « secret » de l'utilisateur.

Ainsi, le calcul de la clé DHT d'un fragment FBL peut se faire comme suit :

CDHTFBL = H (secret o pathname o filename)

Formule 1. Calcul de la clé DHT d'un fragment FBL

Les avantages de cette méthode sont :

- la clé DHT du fragment FBL est inchangeable même après la mise à jour du contenu du fichier
- l'utilisation d'un secret relatif permet d'associer un espace de nommage privé à chaque utilisateur et donc éviter la collision entre les clés DHT des fragments FBL des différents utilisateurs
- le nom et le chemin du fichier sont suffisants pour récupérer le fragment FBL après le crash du disque dur

5.2 Confidentialité

Dans le système pStore, un utilisateur chiffre ses différents fragments FBL par la même clé de chiffrement. Cette méthode est non robuste car la découverte de sa clé de chiffrement, par un nœud malicieux, entraîne le déchiffrement de ses différents fragments FBL. De plus, s'il veut partager l'un de ses fichiers avec d'autres pairs, il doit leur fournir la clé de chiffrement du fragment FBL associé au fichier à partager, mais l'obtention de cette clé entraîne le déchiffrement de tous ses autres fragments FBL.
Afin d'attribuer une clé de chiffrement différente à chaque fragment FBL, nous pouvons utiliser la formule suivante :

CCFBL = H (f (secret) o pathname o filename)

Formule 2. Calcul de la clé de chiffrement d'un fragment FBL

f : fonction appliquée sur le secret de l'utilisateur

Remarque : Le nom et le chemin du fichier sont suffisants pour permettre à l'utilisateur de déterminer la clé DHT et la clé de chiffrement du fragment FBL et par la suite de récupérer tout le fichier. Ces deux informations sont faciles à se souvenir.

5.3 Intégrité

Comme pour le fragment FB, il faut que la vérification de l'intégrité du fragment FBL soit possible par son propriétaire et son nœud de sauvegarde.
La solution utilisée dans le système pStore est la suivante : l'utilisateur signe ses fragments FBL en utilisant sa clé privée et il inclut sa clé publique dans le message à sauvegarder. C'est en utilisant cette deuxième clé que les autres nœuds du réseau P2P vont pouvoir vérifier l'intégrité de ces fragments FBL. Cette solution présente les inconvénients suivants :

- l'identité du propriétaire du fragment est connue (à travers sa clé publique), ceci est en contradiction avec le respect de l'anonymat des utilisateurs.
- un nœud malicieux peut modifier le contenu du fragment puis le signer avec sa clé privée et remplacer la clé publique de l'utilisateur par sa clé publique. Cette modification ne sera détectée que par le propriétaire du fragment car les autres nœuds vont vérifier l'intégrité du fragment en utilisant la clé publique du nœud malicieux.
- la nécessité d'une autorité de certification ou d'une entité de confiance pour la distribution des clés.

Pour résoudre ces problèmes, un utilisateur doit signer ses fragments FBL en suivant le processus décrit à la figure 13 : il ajoute au fragment FBL son condensé et puis il chiffre tout le message (fragment en clair + son condensé) en utilisant sa clé de chiffrement. Cette méthode ne permet qu'au propriétaire du fichier de vérifier l'intégrité de son fragment FBL.

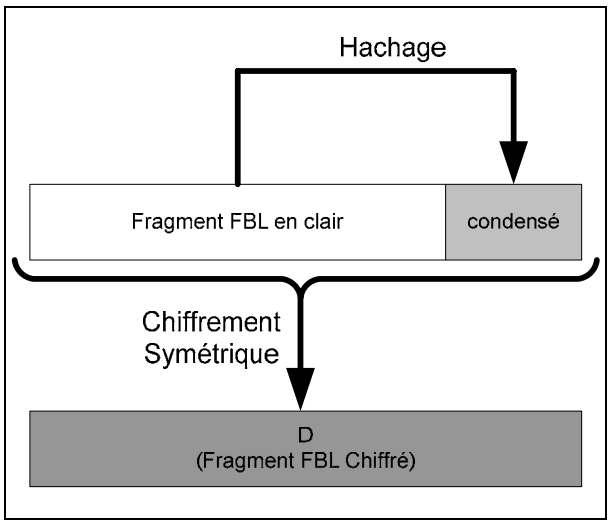

Figure 13. Intégrité du fragment FBL

Afin de permettre aux nœuds de sauvegarde de vérifier l'intégrité des fragments FBL, nous avons introduit une entité de confiance, appelée par la suite « Fournisseur de Service » ou SP *(Service Provider)*. C'est cette entité qui se charge de vérifier si les fragments FBL ont bien été sauvegardés dans le réseau P2P en suivant le processus décrit à la figure 14.

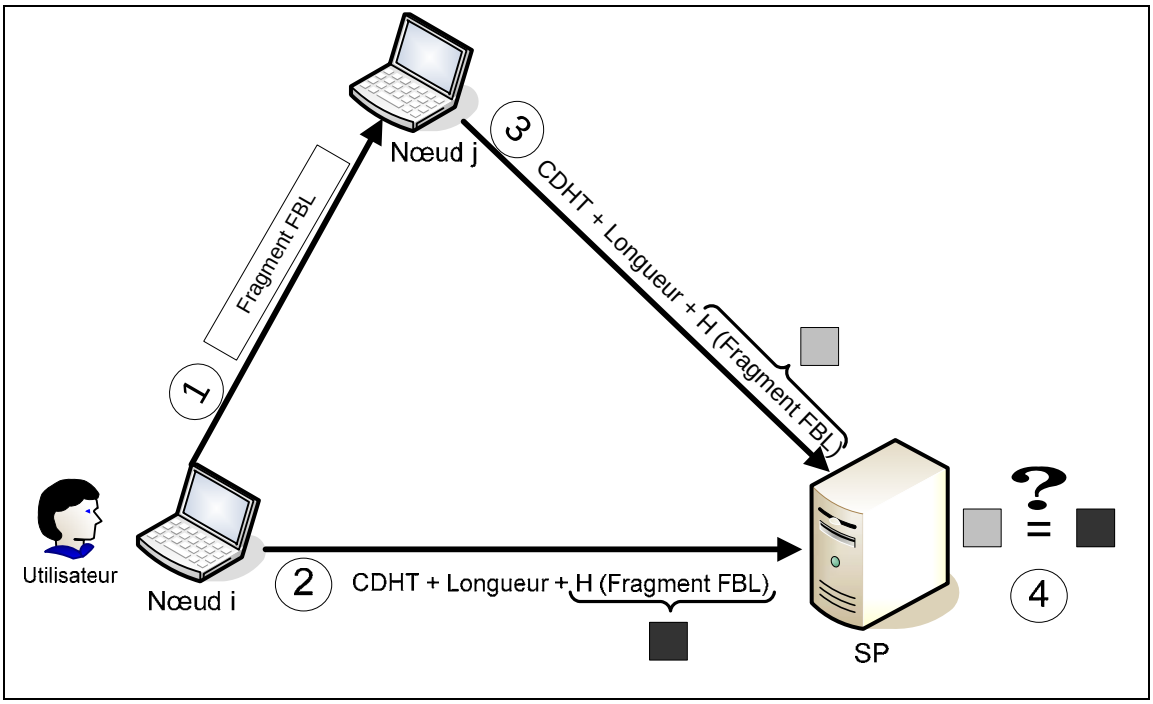

Figure 14. Vérification de l'intégrité d'un fragment FBL par son nœud de sauvegarde

1- L'utilisateur sauvegarde le fragment FBL dans le réseau P2P.
2- L'utilisateur envoie au SP la clé DHT, la longueur et le condensé du fragment FBL.
3- Le nœud de sauvegarde (nœud j) conserve le fragment FBL dans une mémoire cache jusqu'à ce qu'il vérifie son intégrité et il envoie la clé DHT, la longueur et le condensé au SP.
4- En comparant les deux condensés, le SP peut vérifier l'intégrité du fragment FBL reçu par le nœud j : si le fragment est cohérent, il informe le nœud j qui va donc le sauvegarder, sinon il demande au nœud i (le propriétaire du fragment) de le retransmettre.

Les avantages de cette solution sont :

- le nœud de sauvegarde vérifie l'intégrité du fragment FBL sans aucune connaissance de l'identité de son propriétaire (anonymat respecté)
- l'intégrité du fragment est garantie par une simple fonction de hachage (pas besoin ni d'un couple de clé publique/clé privée ni d'un chiffrement supplémentaire)
- toute modification du fragment est détectée
- pour vérifier l'intégrité d'un fragment FBL, le SP fait tout simplement une comparaison

Remarque : La sauvegarde du fragment FBL dans une mémoire cache jusqu'à la vérification de son intégrité évite le remplacement d'un fragment cohérent par un autre incohérent (lors d'une mise à jour du fichier ou à cause d'une attaque par un nœud malicieux).

6 Risque de collision

Dans le mécanisme décrit dans la figure 12, l'identité entre deux fragments FB est affirmée par la comparaison de leurs condensés. Cette méthode de comparaison n'est pas sûre puisqu'il existe un risque de collision : deux fragments différents peuvent aboutir au même condensé et donc peuvent être associés à la même clé DHT.

Malgré ce risque, la comparaison par condensé est utilisée dans plusieurs applications telles que : l'algorithme de synchronisation des fichiers à distance rsyn [14], le système de fichier distribué LBFS [15], les systèmes de sauvegarde distribués Pastiche [16] et pStore [11],…

Les concepteurs de ces applications jugent leur choix par [17]:

- la probabilité d'avoir une collision entre deux messages aléatoires est estimée beaucoup moins faible que la probabilité d'avoir une erreur hardware (erreur de disque dur,…)
- dans le cas pratique, les messages ne sont pas aléatoires (fichier texte,…), ce qui rend le risque de collision beaucoup plus faible

Dans le système DisPairSe, nous avons aussi adopté cette méthode de comparaison.

7 Conclusion

Chaque fichier sauvegardé dans le réseau P2P est constitué d'un ensemble de fragments FB et un seul fragment FBL. Pour assurer l'intégrité et la confidentialité des fragments FB nous avons exploité les caractéristiques d'une table de hachage distribuée (DHT). Tandis que, pour la sécurisation des fragments FBL, nous avons introduit une entité de confiance appelée le « fournisseur de service ».

Dans les chapitres suivants nous découvrirons que cette entité de confiance est indispensable pour contrôler le fonctionnement du système DisPairSe et assurer une bonne qualité de service aux usagers.

III PRINCIPE DE FONCTIONNEMENT DU SYSTEME DISPAIRSE

1 Introduction

Après la présentation des mécanismes de sécurité adoptés pour assurer la confidentialité et l'intégrité des données d'un utilisateur, nous allons définir, dans ce chapitre, les processus suivis par un pair pour sauvegarder, mettre à jour, supprimer ou restaurer un fichier du réseau P2P. Dans ces différentes opérations, nous devons toujours respecter l'anonymat des utilisateurs et préparer les informations nécessaires pour accomplir la fonction de comptabilisation (comptabiliser pour chaque pair l'espace disque fourni et l'espace disque consommé).

2 Sauvegarde d'un fichier

L'opération de base fournie par le système de sauvegarde distribuée DisPairSe est la sauvegarde d'un fichier dans le réseau P2P. Pour accomplir cette opération, un pair doit fragmenter le nouveau fichier à sauvegarder en des fragments FB, appliquer les traitements de sécurité sur chaque fragment FB, construire le fragment FBL, appliquer les traitements de sécurité sur le fragment FBL et enfin sauvegarder l'ensemble de ces fragments dans le réseau P2P. Bien que ce processus crée et protège l'ensemble des fragments nécessaires pour la restauration du fichier, il ne permet ni de garantir que les fragments créés ont bien été sauvegardés dans le réseau P2P, ni de contrôler le comportement de chacun des nœuds, ni d'offrir les informations nécessaires pour accomplir la fonction de comptabilisation.

Pour résoudre ces problèmes, nous avons ajouté au processus de sauvegarde d'un fichier les deux derniers types de communications définis dans la figure 15 (messages 2 et 3). Ainsi, le nouveau processus sera :

1- L'utilisateur crée et sauvegarde les différents fragments dans le système (fragment FBL et l'ensemble des fragments FB).

2- L'utilisateur envoie au SP une description du nouveau fichier qui contient la liste des clés DHT et les longueurs des différents fragments créés ainsi que le condensé du fragment FBL (ces informations sont nécessaires pour la comptabilisation).

3- Chaque nœud envoie au SP une description des nouveaux fragments acquis : s'il s'agit d'un fragment FB, le nœud envoie sa clé DHT et sa longueur et s'il s'agit d'un fragment FBL, il envoie de plus son condensé.

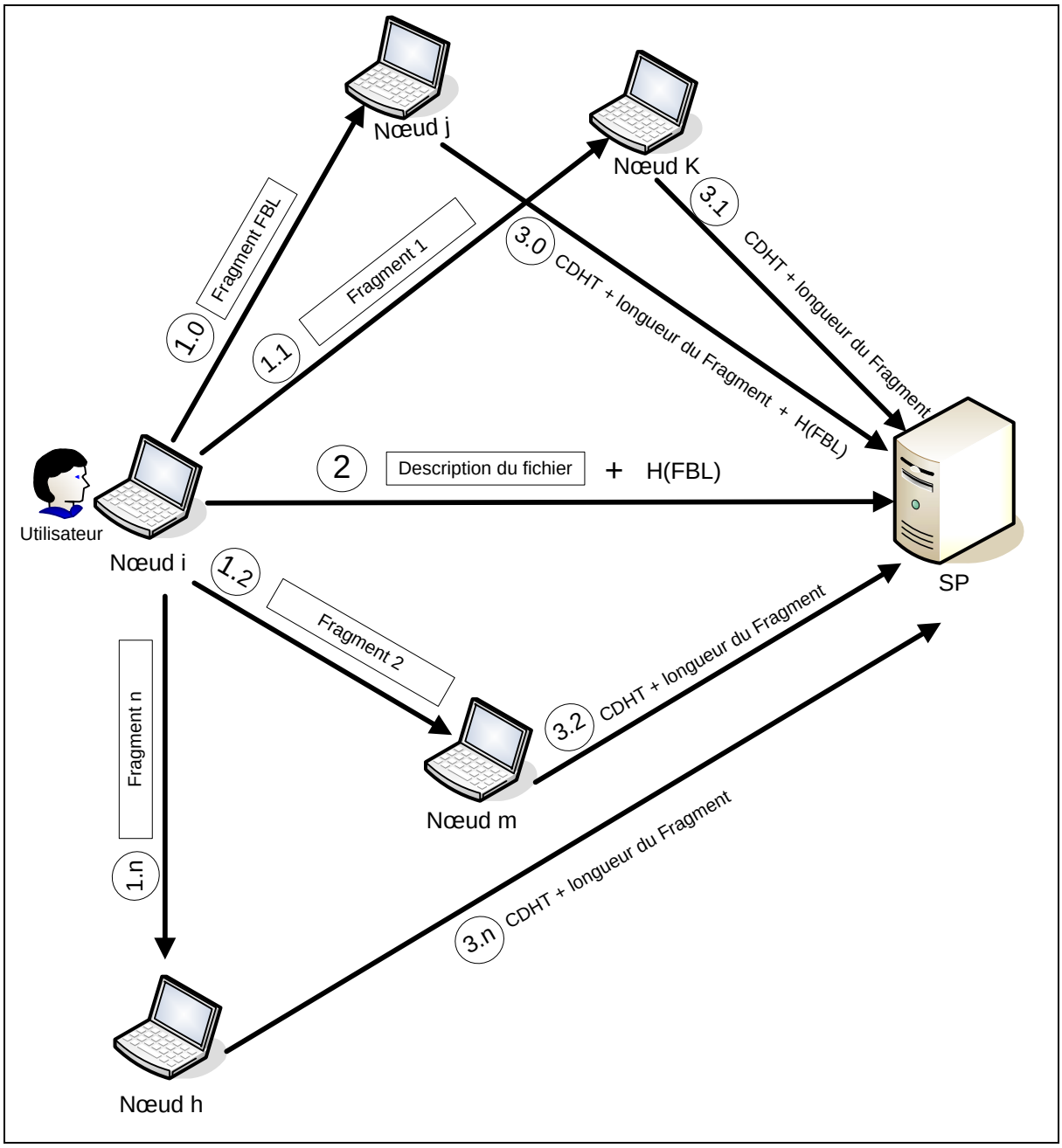

Figure 15. Sauvegarde d'un nouveau fichier

En recevant les déclarations du propriétaire du fichier et celles des nœuds de sauvegarde, le fournisseur de service, qui est une entité de confiance, aura toutes les informations nécessaires pour contrôler le bon fonctionnement du système DisPairSe et accomplir la fonction de comptabilisation. En effet, si nous prenons L1 la liste des clés DHT déclarées par le propriétaire du fichier et L2 la liste des clés DHT acquittées par les nœuds de sauvegarde, le fournisseur de service peut découvrir les cinq cas suivants :

- ***Cas 1 :*** si une clé DHT i appartient à L1 et à L2 et avec des longueurs de fragments identiques ➔ le fragment dont la clé DHT est i a bien été sauvegardé dans le système : l'espace disque consommé par son propriétaire et l'espace disque fourni par son nœud de sauvegarde augmentent.

- ***Cas 2 :*** si une clé DHT i appartient à L1 et n'appartient pas à L2 ➔ le fragment dont la clé DHT est i n'a pas été sauvegardé dans le système (il a été supprimé ou modifié par un nœud malveillant) : le SP demande à son propriétaire de le retransmettre.

- ***Cas 3 :*** si une clé DHT j appartient à L2 et n'appartient pas à L1 ➔ soit un pair n'a pas déclaré ce fragment pour le SP, soit le nœud de sauvegarde n'a pas acquis réellement ce fragment : le SP envoie une requête pour supprimer le fragment associé à cette clé DHT.

- ***Cas 4 :*** si une même clé DHT appartient à la fois à la liste L1 et L2 mais avec des longueurs de fragments différents ➔ soit le propriétaire du fragment, soit le nœud final est malveillant : le SP peut vérifier la longueur du fragment en le récupérant du réseau P2P et puis diminuer le niveau de fiabilité du nœud qui a modifié sa longueur réelle (un nœud à toujours intérêt à diminuer la longueur de ses fragments et à augmenter la longueur des fragments qu'il stocke).

- ***Cas 5 :*** si une même clé DHT d'un fragment FBL appartient à la fois à la liste L1 et L2 mais avec des condensés de fragments différents (le condensé envoyé par le propriétaire du fragment est différent de celui calculé par le nœud de sauvegarde) ➔ le fragment FBL a été modifié : le SP demande au nœud de sauvegarde de le supprimer et à son propriétaire de le retransmettre.

Remarque : Si un pair sauvegarde un fichier sans informer le SP de la liste des fragments qu'il a créée et sauvegardée dans le réseau P2P, la liste L1 sera vide et c'est la liste L2 qui contiendra la liste des clés DHT des différents fragments créés : il s'agit du troisième cas et donc le SP va demander la suppression de tous les fragments créés.

Cette solution possède les avantages suivants :

- conserver l'anonymat des utilisateurs : aucun fragment sauvegardé dans le système (soit FBL ou FB) ne contient d'information sur l'identité de son propriétaire
- supprimer tout fragment qui n'a pas été déclaré par son propriétaire
- le SP possède toutes les informations nécessaires pour comptabiliser l'espace disque fourni et consommé par chaque utilisateur
- le SP détecte les tentatives d'un nœud de modifier la longueur réelle des fragments. De plus, ces informations sont utiles pour déterminer le niveau de fiabilité de chaque nœud
- l'utilisateur est sûr que ces fragments ont été bien sauvegardés dans le système (la suppression des fragments par des nœuds intermédiaires sera détectée)

Problème de multi acquittement

Plusieurs nœuds peuvent affirmer l'acquisition du même fragment (dont un seul est un nœud de confiance). En se basant sur une propriété du protocole Pastry *« chaque fragment est sauvegardé dans le nœud dont l'identifiant est le plus proche numériquement de la clé DHT du fragment »* et en connaissant l'identifiant de chaque nœud, le fournisseur de service peut déterminer le nœud qui sauvegarde réellement le fragment.
Par exemple, dans le scénario présenté dans la figure 16, les trois nœuds N3, N4 et N5 affirment l'acquisition d'un fragment de clé 6. En comparant les identifiants de ces nœuds avec la clé DHT du fragment, le SP peut déterminer que c'est le nœud N5 qui sauvegarde réellement le fragment (ce nœud possède l'identifiant le plus proche numériquement à la clé 6). Ainsi, le fournisseur de service met à jour la quantité d'espace disque offerte par le nœud N5 au système DisPairSe et il diminue le niveau de fiabilité des deux autres nœuds.

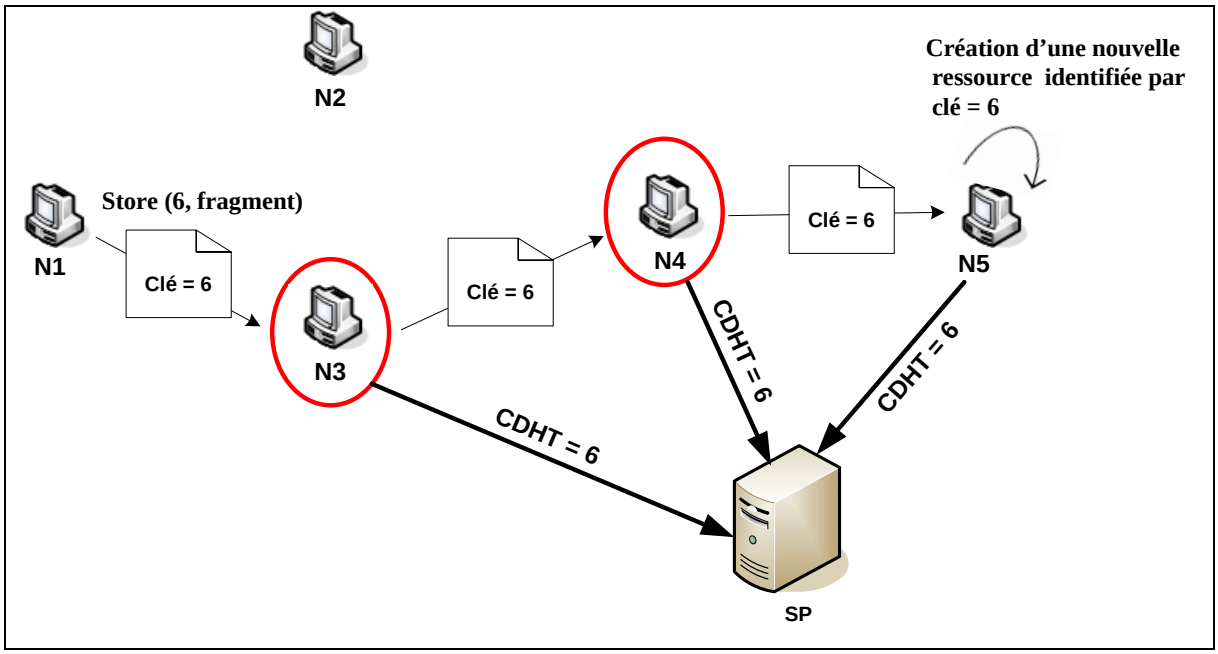

Figure 16 : Problème de multi acquittement

Remarque : Les différents échanges entre un pair et le SP sont périodiques. Par exemple toutes les deux heures, chaque nœud envoie au SP une description des nouveaux fichiers créés et une description des nouveaux fragments acquis.

3 Mise à jour d'un fichier

Pour la mise à jour d'un fichier, l'utilisateur suit le même scénario décrit à la figure 15. La seule différence, c'est que la nouvelle description du fichier contient uniquement les nouveaux fragments à créer et que l'utilisateur ne sauvegardera (étape 1) que les fragments qui n'existent pas dans les anciennes versions (cette liste peut être déterminée par l'algorithme rsync [18]).
Le seul fragment qui sera mis à jour est le fragment FBL, le nœud qui le sauvegardera doit vérifier son intégrité avant de supprimer son ancienne version.

4 Suppression d'un fichier

Le système DisPairSe doit donner à l'utilisateur la possibilité de supprimer d'anciennes versions d'un fichier ou un fichier tout entier afin de libérer de l'espace disque pour sauvegarder de nouvelles versions ou des fichiers plus importants.
Cette tâche est très difficile à réaliser dans un réseau P2P totalement anonyme. En effet, un nœud de sauvegarde doit s'assurer de l'identité d'un pair et vérifier ses droits d'accès avant d'accepter sa demande de suppression d'un fragment, et ceci se contredit avec le respect de l'anonymat des utilisateurs. À cause de cette complexité, le système PeerStore [19] n'autorise pas la suppression des fichiers. Les autres systèmes de sauvegarde distribués adoptent les solutions suivantes :

- supprimer les fichiers rarement utilisés ou associer une date d'expiration pour chaque fichier. Cette méthode est utilisée par plusieurs systèmes tels que Freenet [20], Chord [7],… mais elle ne peut pas être adoptée pour un système P2P dont le but initial est de récupérer les données perdues pour les raisons suivantes :
  - par définition, dans ce type de systèmes, l'utilisateur accède rarement à ses fichiers : uniquement après la perte de ses données en local

- l'utilisateur ne peut pas préciser une date d'expiration pour ses fichiers distribués dans le réseau car il ne sait pas quand ses données locales seront perdues (le crash du disque dur est un événement aléatoire).

➢ Associer à chaque fragment la clé publique de son propriétaire afin que la demande de suppression soit signée par la clé privée correspondante. Cette solution est adoptée par la plupart des systèmes de sauvegarde distribuée tels que : Farsite [21], pStore [11],…bien qu'elle possède les inconvénients suivants :

- viol de l'anonymat des utilisateurs : en connaissant la clé publique du propriétaire du fragment, il est facile de déterminer son identité
- un nœud malicieux peut remplacer la clé publique du propriétaire du fragment par sa clé publique pour pouvoir par la suite supprimer le fragment

Pour permettre la suppression des fragments tout en assurant l'anonymat des utilisateurs, nous avons proposé les trois solutions suivantes :

Solution 1 :

Dans cette solution, la demande de suppression d'un fragment est signée par le fournisseur de service (une entité de confiance) avant d'être acceptée et exécutée par son nœud de sauvegarde. Le processus par lequel un utilisateur supprime un fichier (ou une version du fichier) est décrit à la figure 17.

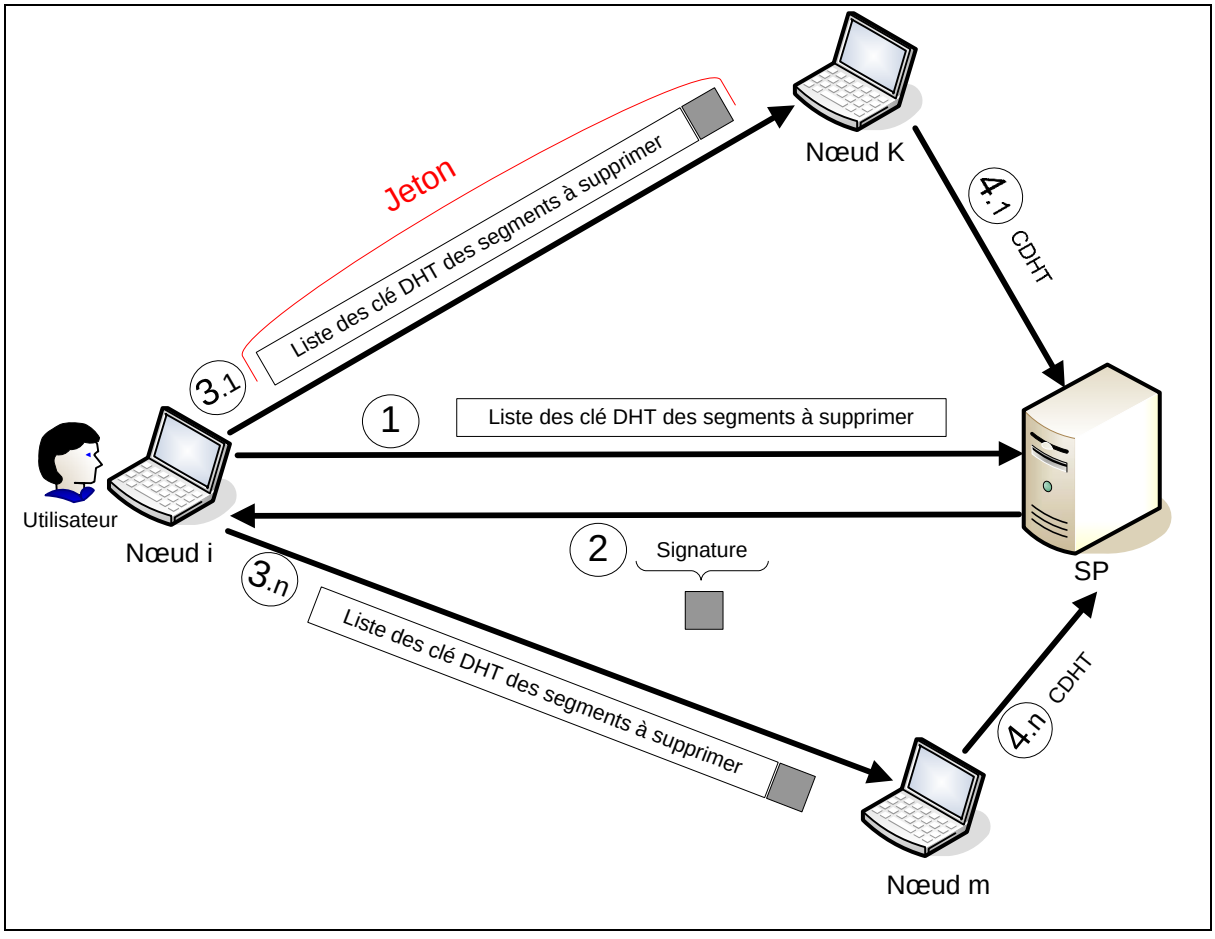

Figure 17. Processus de suppression d'un fichier en utilisant les jetons

1- L'utilisateur envoie au SP la liste des clés DHT des fragments qu'il veut supprimer.

2- Si l'utilisateur a le droit de supprimer ces différents fragments, alors le SP lui envoie son autorisation (la signature électronique des clés DHT des différents fragments à supprimer), sinon la demande de l'utilisateur est ignorée.

3- L'utilisateur envoie une requête de suppression contenant le jeton de suppression (liste des clés DHT + la signature du SP) pour chaque nœud contenant un fragment à supprimer.

4- Chaque nœud recevant la requête de l'utilisateur vérifie la validité du jeton de suppression (en utilisant la clé publique du SP) et que la clé DHT du fragment à supprimer est incluse dans la liste définie dans le jeton. Si ces deux contraintes sont vérifiées, le nœud supprime le fragment et informe le SP de la clé DHT du fragment supprimé, sinon la demande de l'utilisateur est ignorée.

En réalité, le nœud de sauvegarde associe, pour chaque fragment acquis, un compteur qui comptabilise le nombre d'utilisateurs qui le partagent, et lors de la réception d'une demande de suppression, il décrémente le compteur associé au fragment à supprimer. Le fragment ne sera supprimé réellement que s'il n'est plus référencé par aucun nœud (son compteur égal à 0). En plus, pour éviter les attaques par rejeu, le jeton de suppression comporte un numéro de séquence défini par le fournisseur de service.

Remarque : Le but des acquittements envoyés par les nœuds de sauvegarde (messages 4) est de déterminer la liste des clés DHT des fragments qui n'ont pas été supprimés (les nœuds qui les sauvegardent ne sont pas accessibles ou un nœud malicieux a supprimé la demande de l'utilisateur). Le SP se charge donc de les supprimer dès que possible afin de ne pas garder de fragments inutiles dans le réseau P2P.

L'accès au fournisseur de service lors des opérations de suppression ne va pas le charger. En effet, dans un système de sauvegarde distribuée, le but principal est de sauvegarder des fichiers pour pouvoir les récupérer en cas de crash du disque dur et donc ces opérations sont rares. En plus, l'accès au SP ne se fait pas par fragment mais par ensemble de fragments (un jeton de suppression peut contenir la liste des clés DHT des fragments de plusieurs versions ou fichiers à supprimer).

Solution 2 :

Cette solution consiste à définir les droits de suppression sous forme de couples (clé DHT, clé de suppression) sauvegardés dans les nœuds de sauvegarde. Le couple (s1, c1) signifie que le fragment s1 peut être supprimé en utilisant la clé de suppression c1, cette information ne contient aucune indication sur le propriétaire du fragment, d'où le respect de l'anonymat des utilisateurs.
La clé de suppression d'un fragment est créée par son propriétaire et envoyée vers le nœud de sauvegarde à travers le fournisseur de service (cf. Figure 18).

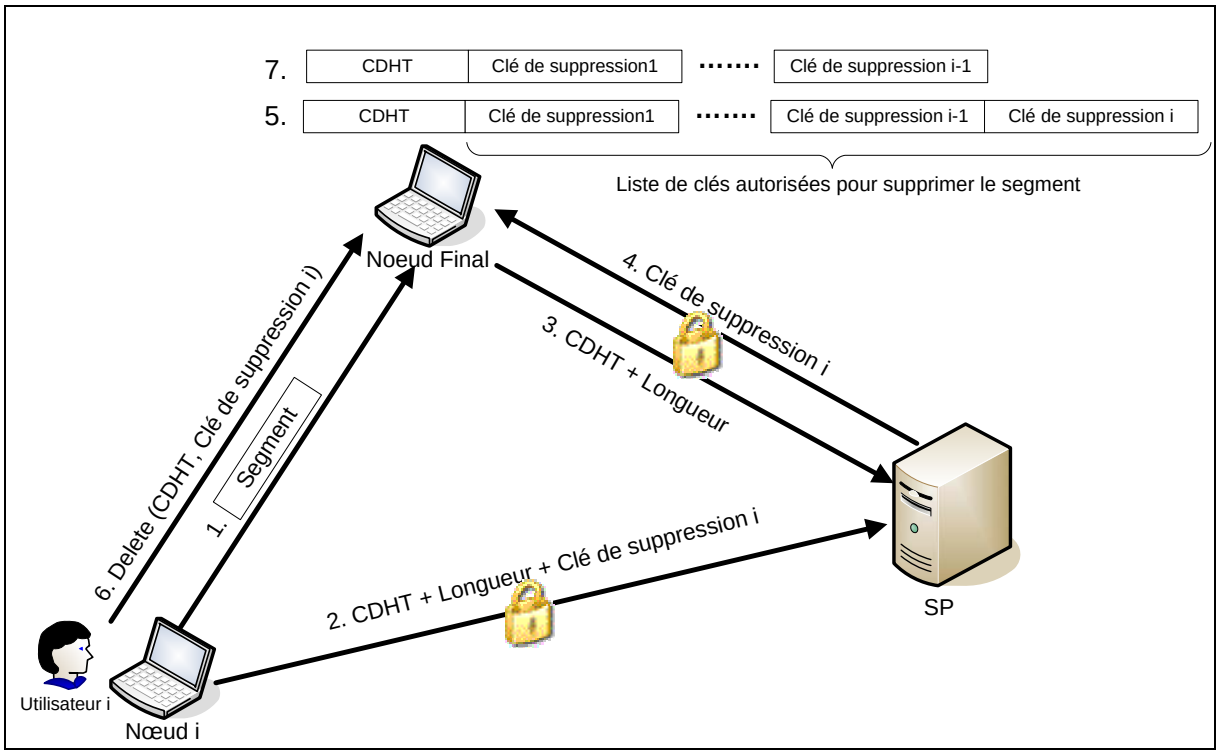

Figure 18. Processus de suppression d'un fragment en utilisant les clés de suppression

Le scénario de suppression d'un fragment est le suivant :

1- L'utilisateur sauvegarde ses fragments dans le réseau.

2- Pour chaque fragment créé, l'utilisateur envoie sa clé DHT, sa longueur et sa clé de suppression au fournisseur de service (ces informations sont inclues dans la description du nouveau fichier sauvegardé).

3- Pour chaque fragment acquis, le nœud de sauvegarde envoie sa clé DHT et sa longueur au fournisseur de service.

4- Pour chaque fragment acquitté, le SP vérifie s'il est déjà déclaré par un nœud :

- o si oui, le SP envoie au nœud de sauvegarde la clé de suppression du nouveau fragment reçu (message 4)
- o sinon, le SP demande au nœud de sauvegarde la suppression du nouveau fragment reçu

5- Le nœud final ajoute la nouvelle clé de suppression à l'ensemble des clés autorisées pour supprimer le fragment.

6- Pour supprimer un fragment, l'utilisateur doit présenter sa clé DHT et sa clé de suppression.

7- Quand un nœud final reçoit une demande de suppression d'un fragment, il vérifie si la clé de suppression contenue dans cette commande appartient à l'ensemble des clés autorisées :

- o si oui la clé de suppression utilisée est supprimée de l'ensemble pour éviter les attaques par rejeu (comme dans cet exemple)
- o sinon la demande de suppression est ignorée

Comme dans la première solution, un fragment ne sera supprimé réellement que si son ensemble de clés de suppression autorisées est vide. En plus, la sécurisation de la communication entre un pair et le fournisseur de service ne permet à aucun nœud malicieux d'intercepter les clés de suppression des autres pairs.

Remarque : Chaque pair choisit aléatoirement les clés de suppression de ces fragments et il les sauvegarde dans les fragments FBL associés afin de les récupère lors d'une demande de suppression.

Solution 3 :

Dans cette solution, c'est le fournisseur de service qui se charge de supprimer du réseau P2P les fragments qui ne sont plus référencés par aucun pair. Le mécanisme de suppression des fragments est le suivant :

- chaque nœud inclut dans ses messages périodiques envoyés au fournisseur de service les clés DHT des fragments à supprimer.
- pour chaque fragment, le fournisseur de service applique le processus décrit à la figure 19.
- le fournisseur de service inclut dans ses messages périodiques envoyés aux nœuds de sauvegarde, les clés DHT des fragments à supprimer.

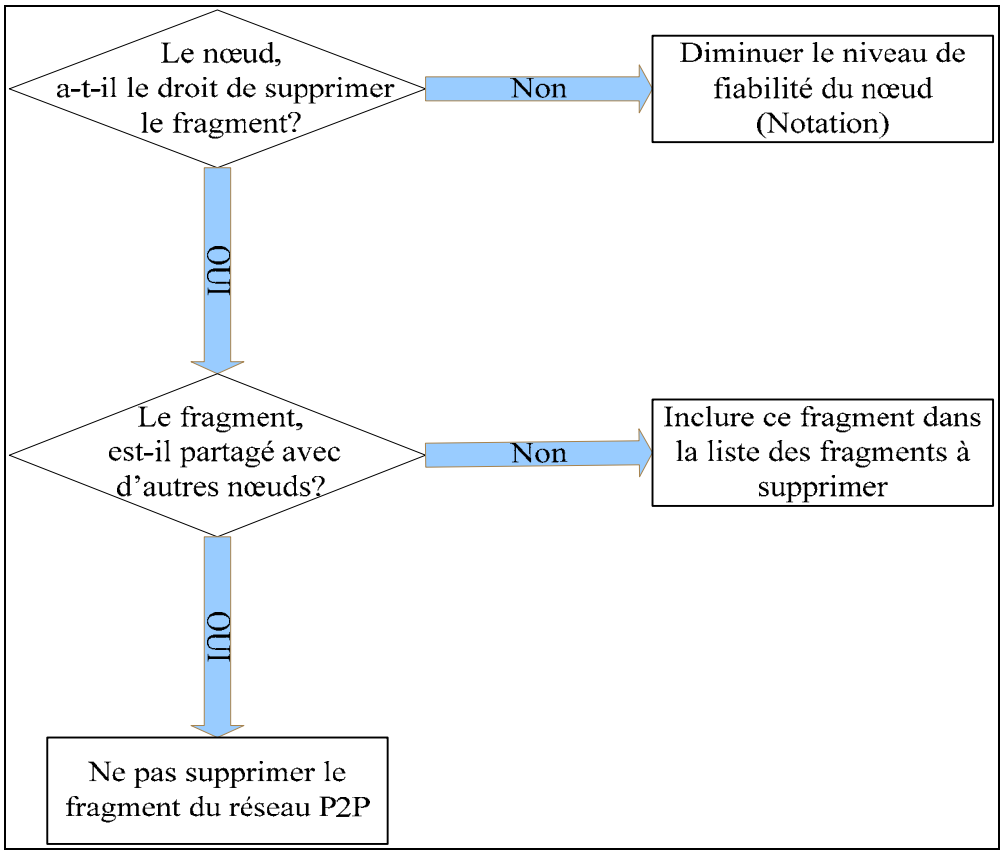

Figure 19. Contrôle effectué par le SP sur chaque demande de suppression

Comparaison entre les trois solutions

Bien que ces trois solutions assurent l'anonymat des utilisateurs, elles diffèrent sur les points mis en évidence dans le tableau 1.

	Solution 1	Solution 2	Solution 3
Intervention du SP	Il faut accéder au SP pour supprimer un fragment	Pour supprimer un fragment, il suffit de récupérer sa clé de suppression à partir du fragment FBL (sans accéder au SP)	Les demandes de suppression sont envoyées par le SP
Consommation des ressources du système	Les droits de suppression sont gérés sans consommation supplémentaire des ressources du système (les espaces disque fournis par les différents pairs)	Il faut définir une clé de suppression pour chaque couple (utilisateur, fragment). Cette clé est sauvegardée dans le nœud de sauvegarde et le fragment FBL, ce qui entraîne une consommation supplémentaire des ressources du système	Les droits de suppression sont gérés sans aucune consommation supplémentaire des ressources du système
Fiabilité	Solution fiable : pour pouvoir supprimer un fragment sans autorisation il faut déterminer la clé privée de l'utilisateur pour pouvoir créer un jeton de suppression	La fiabilité de cette solution dépend de la longueur de la clé de suppression	Solution très fiable car la communication entre un nœud et le fournisseur de service est sécurisée

Tableau 1. Comparaison entre les trois procédures de suppression proposées

Dans notre implémentation nous avons adopté la troisième solution puisqu'elle n'exige aucun transfert supplémentaire ni entre les nœuds, ni entre un nœud et le fournisseur de service (elle exploite uniquement les transferts périodiques déjà définis) et en plus elle permet de détecter les demandes de suppression non autorisées. Ce type de demande est utilisé pour déterminer le niveau de fiabilité de chaque nœud (cf. Figure 19).

5 Restauration des fichiers

En cas de perte d'un fichier, l'utilisateur n'a besoin que de préciser son nom et son chemin afin que le système P2P puisse le récupérer (cf. Figure 20). En effet, à partir de ces deux informations, le système détermine tout d'abord la clé DHT et la clé de chiffrement du fragment FBL du fichier à restaurer (voir Formules 1 et 2), puis il extrait toutes les clés DHT et les clés de chiffrement associées aux fragments FB du fichier. À la fin, le système P2P récupère et déchiffre tous les fragments de données nécessaires du réseau P2P et il reconstitue le fichier dans son entier.

En cas de crash du disque dur, l'utilisateur peut récupérer toutes ces données déjà sauvegardées dans le réseau P2P en répétant le processus de la figure 20 autant de fois que le nombre de fichiers à restaurer. Les noms et les chemins des fichiers précédemment sauvegardés dans le réseau peuvent être récupérés à partir du fournisseur de service (ces informations sont déjà contenues dans les descriptions des fichiers envoyées par les différents nœuds).

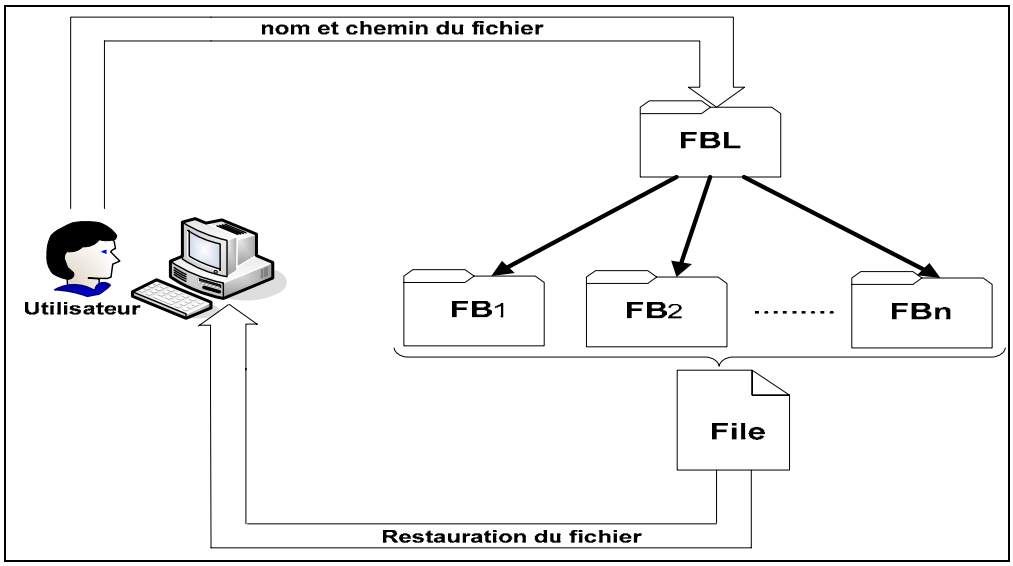

Figure 20. Processus de restauration d'un fichier

6 Conclusion

À travers les processus de sauvegarde et de suppression d'un fichier, nous remarquons que le fournisseur de service joue un rôle très important dans le fonctionnement du système DisPairSe. En effet, il permet de vérifier si les fragments d'un utilisateur ont bien été sauvegardés dans le réseau P2P, il se charge de supprimer les fragments inutiles et il collecte les informations nécessaires pour assurer la fonction de comptabilisation.

Dans le chapitre suivant, nous décrivons l'architecture que nous avons proposée afin de protéger le fournisseur de service contre les attaques de déni de service.

V ARCHITECTURE ET SERVICE AAA DANS DISPAIRSE

1 Introduction

Les différents systèmes de sauvegarde distribuée qui ont été élaborés (tels que pStore, PeerStore, Pastiche,…), et comme la plupart des systèmes P2P, offrent un service gratuit et sans contrôle d'accès.

Dans le nouveau système DisPairSe, nous avons résolu aussi ces deux problèmes en introduisant les services AAA (***A**uthentication*, ***A**uthorization*, ***A**ccounting*). En effet, le premier service permet d'assurer une authentification mutuelle entre un pair et le fournisseur de service afin d'établir par la suite une communication sécurisée entre eux, le deuxième service permet de donner l'accès au service de sauvegarde uniquement pour les nœuds autorisés et le troisième service comptabilise pour chaque pair l'espace disque fourni et l'espace disque consommé et ceci afin de lui attribuer une facture débitrice lorsqu'il consomme plus de ressources qu'il n'en fournit ou bien une rémunération dans le cas contraire.

Dans ce chapitre nous introduisons, tout d'abord, le protocole PANA qui sera utilisé pour transporter les messages d'authentification des nœuds vers le réseau d'accès. Puis nous présentons l'architecture que nous avons proposée pour le système DisPairSe et les techniques permettant d'introduire les services AAA. Enfin, nous traitons la sécurisation de la communication entre un nœud du réseau P2P et le fournisseur de service.

2 Le protocole PANA

Dans un système P2P, les pairs appartiennent à des réseaux IP différents et donc les messages d'authentification qui seront échangés entre un pair et le serveur d'authentification seront relayés par plusieurs routeurs. Par conséquent, il est plus facile d'utiliser un mécanisme d'authentification qui opère au-dessus de la couche réseau (couche 3 dans le modèle OSI) pour bénéficier du service de routage de cette dernière. Ce besoin est rempli par le protocole PANA [22].

2.1 Présentation

Le protocole PANA (Protocol for Carrying Authentication and Network Access) offre aux clients un moyen d'authentification au dessus de la couche IP. Ainsi, il n'est pas nécessaire que le serveur d'authentification et le système à authentifier soient dans le même réseau (contrairement aux protocoles 802.1x [23] et 802.11 [24] qui ne sont utilisables que dans des réseaux locaux ou métropolitains).

PANA ne définit ni de nouvelles méthodes d'authentification, ni de nouveaux algorithmes de génération de clés. Il s'appuie tout simplement sur l'encapsulation EAP (Extensible Authentication Protocol) [25] pour assurer l'authentification des clients.

La figure 21 présente les correspondances entre les piles protocolaires lors de l'échange des messages d'authentification entre un client (appelé client PANA) et le serveur d'authentification (appelé serveur AAA) à travers une entité de relais (appelée PANA Authentication Agent).

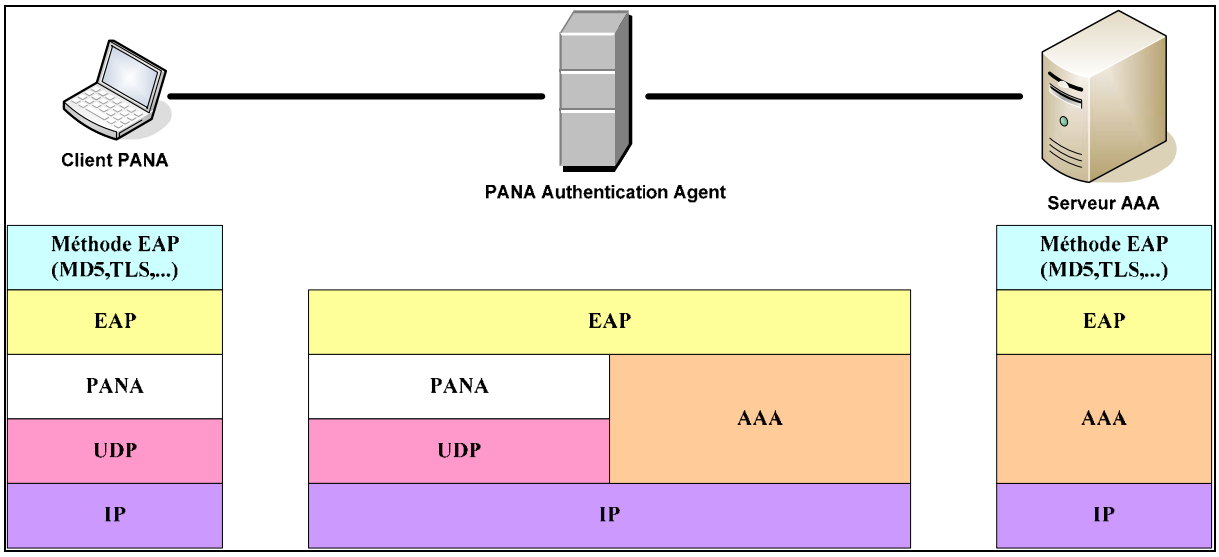

Figure 21. Encapsulation des paquets EAP durant une authentification PANA

2.2 Architecture

Une architecture de contrôle d'accès basée sur le protocole PANA contient quatre entités (cf. Figure 22) [26]:

- ***Client PANA (PaC) :*** la partie client du protocole PANA. Il réside dans l'entité cherchant à s'authentifier auprès du serveur AAA afin d'accéder à un service ou un réseau (tel que Internet).

- ***Serveur d'authentification (AS) :*** l'entité responsable de vérifier les paramètres d'un client cherchant à accéder au réseau et d'envoyer le résultat de la phase d'authentification au PAA. Généralement, un protocole AAA (tel que : RADIUS ou Diameter) assure la communication entre AS et PAA.

- ***Agent d'Authentification PANA (PAA) :*** la partie serveur du protocole PANA. Il se charge de relayer les échanges liés à l'authentification entre PAC et AS, et de transmettre le résultat de la phase d'authentification, reçu du serveur d'authentification, à l'EP.

- ***Enforcement Point (EP) :*** l'entité responsable du contrôle d'accès. En effet, l'EP ne laisse passer le trafic de données d'un utilisateur qu'une fois une autorisation est reçue du AS. Par contre, le trafic de configuration et d'authentification (tels que : DHCP, PANA) peut passer librement.

Remarque : Dans une architecture PANA, le PAA et l'EP peut résider dans un même équipement, appelé NAS (Network Access Server), et dans ce cas leur communication est assurer par un API.

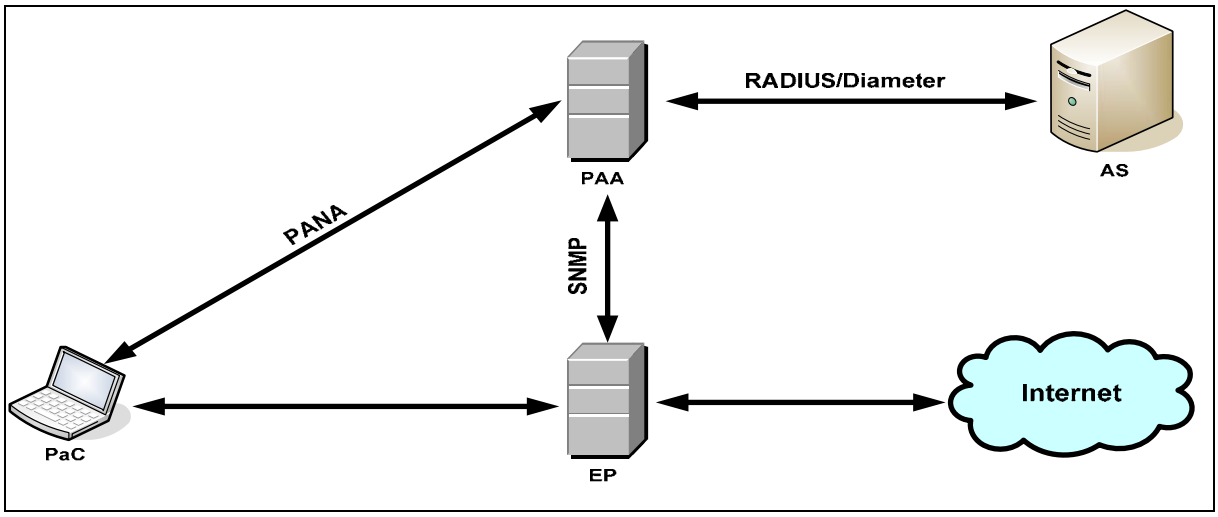

Figure 22 : Architecture du protocole PANA

2.3 Les différentes phases de PANA

Une session PANA comporte les phases suivantes [27] :

- ***Phase d'authentification et d'autorisation***

Cette phase permet d'initier une nouvelle session PANA. Son rôle principal est de transporter les messages EAP entre le PaC et le PAA. À la fin de cette phase, le résultat de l'authentification est transmis du PAA vers le PaC.

- ***Phase d'accès***

Après une authentification réussie, l'EP est configurée pour autoriser le trafic de données provenant du PaC. Par conséquent, ce dernier peut envoyer et recevoir un trafic IP à travers l'EP. Durant cette phase, le PaC et le PAA peuvent échanger des messages afin de tester la validité de la session PANA.

- ***Phase de réauthentification***

Cette phase permet d'étendre la durée de vie d'une session PANA. Elle peut être démarrée par le PaC ou le PAA.

- ***Phase de terminaison***

Cette phase permet de terminer une session PANA et donc la libération de toutes les ressources réservées. Elle peut être démarrée par le Pac ou le PAA.

Remarque : Une session PANA peut être précédée par une phase de découverte dans laquelle le PaC découvre le PA, mais elle ne fait pas partie de la spécification du protocole PANA [27].

3 Architecture du système DisPairSe

L'architecture que nous avons proposée pour le système DisPairSe se base sur le protocole PANA (cf. Figure 23). Ce protocole permet à toute machine connectée au réseau Internet de s'authentifier auprès de notre serveur d'authentification et donc d'accéder au service DisPairSe. Contrairement à l'architecture de la figure 22 où les utilisateurs s'authentifient afin d'accéder à un réseau (le réseau Internet), dans le système DisPairSe, les pairs s'authentifient afin d'obtenir le droit d'accès à un serveur (le fournisseur de service).

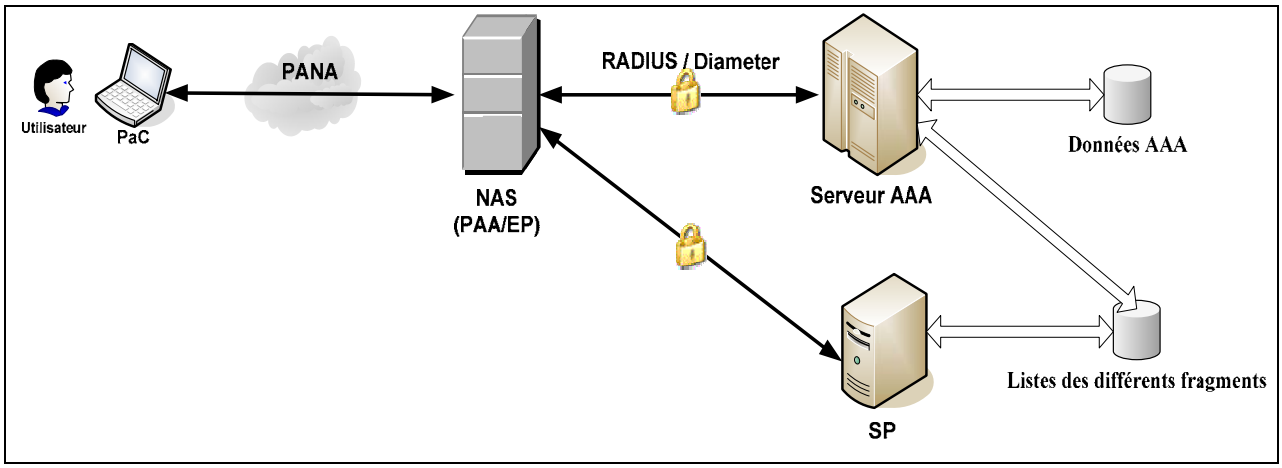

Figure 23. Architecture du système DisPairSe

L'avantage de cette architecture, c'est qu'elle permet de protéger le fournisseur de service, qui est l'entité la plus impliquée dans la fourniture du service DisPairSe, contre les attaques de déni de service. En effet, tous les messages provenant des nœuds non autorisés sont supprimés par le NAS. En plus, il est possible de dupliquer le nombre de NAS dans le système afin de garantir toujours l'accessibilité du SP (on n'aura plus un seul point de contrôle).

À travers cette architecture, comment pouvons-nous introduire les services AAA ? C'est la réponse que nous tentons d'apporter dans la section suivante.

4 Service AAA dans DisPairSe

AAA signifie ***A****uthentication*, ***A****uthorization*, ***A****ccounting*, soit authentification, autorisation et comptabilisation. La signification de ces termes est la suivante :

- Authentification : consiste à vérifier qu'une entité (personne/équipement) est bien celle qu'elle prétend être.
- Autorisation : consiste à permettre uniquement aux utilisateurs autorisés l'accès à certains services ou ressources.
- Comptabilisation : consiste à collecter des informations sur l'utilisation des ressources. Ceci permet par exemple à un opérateur de facturer un utilisateur en fonction de sa consommation.

Dans les réseaux P2P utilisant une DHT, les pairs sont strictement équivalents, ils n'ont aucune relation de confiance entre eux, chaque nœud peut accéder librement au service fourni et il n'y a aucun nœud de confiance. Ceci rend difficiles l'introduction des services AAA et tout particulièrement le service de comptabilisation. Pour résoudre ce problème, nous avons introduit dans le fonctionnement du système DisPairSe une entité centrale de confiance : le fournisseur de service.
Dans ce paragraphe, nous expliquons les solutions que nous avons proposées pour contrôler l'accès au service DisPairSe et assurer la fonction de comptabilisation.

4.1 Authentification

Chaque pair du réseau P2P DisPairSe doit s'authentifier auprès du serveur AAA afin de pouvoir par la suite accéder au fournisseur de service. L'authentification est assurée par l'une des méthodes EAP [25] telles que : EAP-MD5, EAP-TLS…
L'identité d'un pair est conservée dans le domaine administratif (formé par le SP et le serveur AAA) et elle ne sera jamais transmise aux autres nœuds du réseau P2P. Ainsi, nous garantissons

l'anonymat entre les nœuds et l'authentification mutuelle entre un pair (un utilisateur) et le système administratif.

4.2 Autorisation

D'après le processus de sauvegarde d'un nouveau fichier (voir Paragraphe III.2), l'autorisation d'accès au service DisPairSe est équivalente à l'autorisation d'accès au fournisseur de service (SP). En effet :

- tout fragment créé sans informer le SP de sa clé DHT est supprimé. Ainsi, chaque pair sera obligé de s'authentifier afin de pouvoir déclarer la liste des fragments qu'il crée au fournisseur de service.
- chaque nœud recevant un nouveau fragment doit informer le SP de sa clé DHT et sa longueur pour qu'il soit rémunéré. Par conséquent, un pair n'a pas accès au fournisseur de service n'a aucun intérêt à sauvegarder les fragments d'autres nœuds.

Initialement le NAS (PAA/EP) ne laisse passer que le trafic d'authentification. Après l'authentification réussie d'un utilisateur, le serveur AAA envoie une requête Access-Accept au NAS pour permettre à l'utilisateur de joindre le fournisseur de service et donc l'accès au service DisPairSe.
La requête Access-Accept peut contenir des informations supplémentaires pour contrôler la session de l'utilisateur telles que :

- *Session-Timeout* : la durée maximum de la session.
- *Idle-Timeout* : le délai d'inactivité maximum (cette valeur est faiblement supérieure à la période des échanges entre un pair et le SP).

Remarque : Afin d'assurer le bon fonctionnement du service DisPairSe, le serveur AAA peut interdire les nœuds non fiables ou les nœuds qui n'ont pas payé leurs factures de consommation de joindre le fournisseur de service et donc de bénéficier encore du service (même s'ils sont authentifiés avec succès).

4.3 Accounting

Dans tous les systèmes de sauvegarde distribuée, chaque utilisateur a intérêt à ce que les autres mettent à disposition leur espace mémoire, mais il n'y a pas d'incitation directe à offrir son propre espace mémoire. Pour résoudre ce problème, le service de facturation comptabilise, pour chaque utilisateur, l'espace disque fourni et l'espace disque consommé, et ceci afin de lui attribuer une facture débitrice lorsqu'il consomme plus de ressources qu'il n'en fournit, ou bien une rémunération dans le cas contraire.

Dans une architecture AAA classique, la facturation se base sur la durée d'accès au service ou la quantité de données échangées et elle exploite principalement deux types de paquets : Accounting Start et Accounting Stop (cf. Figure 24) [28].
Le paquet Accounting Start est émis par le NAS après une connexion effective de l'utilisateur suite à une phase d'authentification réussie. Ce paquet contient des données de base telles que : adresse IP affectée à l'utilisateur, date et heure de connexion, type de connexion, etc.

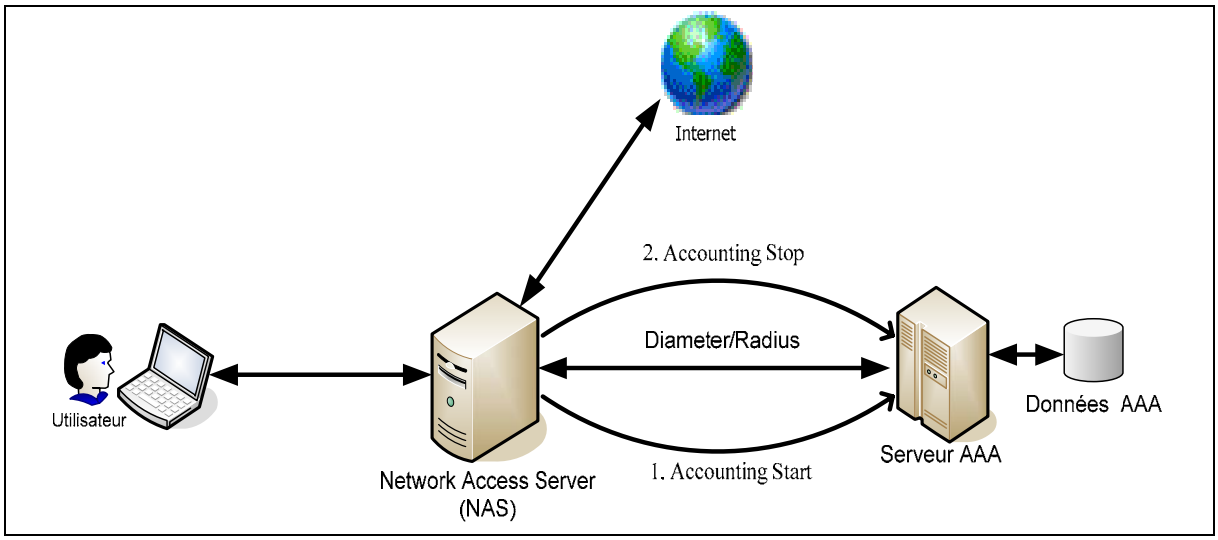

Figure 24. Architecture AAA classique

Quand l'utilisateur se déconnecte du service, le NAS envoie un paquet Accounting Stop au serveur AAA contenant plusieurs paramètres tels que : temps de connexion, nombre de paquets et d'octets échangés selon les divers protocoles, et éventuellement des informations plus confidentielles sur les sites visités ou contenus échangés.

Cette méthode de facturation ne peut pas être appliquée dans le système DisPairSe où la durée d'accès au réseau P2P ainsi que la quantité de données échangées n'intervient pas dans la comptabilisation. Dans ce type de cas, il faut ajouter au serveur AAA des modules supplémentaires dépendant du service fourni afin qu'il réalise ces différentes fonctions. Un tel module est nommé ASM (Application Specific Module) dans [29].
Dans notre système, les modules spécifiques sont déployés au niveau du fournisseur de service. En effet toutes les informations utiles pour la comptabilisation (la liste des différents fragments sauvegardés dans le système ainsi que leurs longueurs, leurs nœuds d'origines et leurs nœuds de sauvegarde) sont récupérées, vérifiées et sauvegardées par le SP dans une base de données (cf. Figure 23). Pour achever la fonction de comptabilisation, il suffit que le serveur AAA accède périodiquement à cette base de données pour déterminer la liste des nouveaux fragments créés et mettre à jour les données AAA de facturation de chaque utilisateur.

Remarque : Les deux paquets Accounting Start et Accounting Stop peuvent être exploités pour déterminer le degré de disponibilité de chaque nœud.

5 Sécurisation de la communication entre un pair et le SP

Après une authentification réussie, il faut sécuriser les communications entre un pair et le fournisseur de service, c'est-à-dire garantir l'intégrité, la confidentialité et l'authentification de l'origine des messages. Ceci peut être assuré par l'une des solutions suivantes :

Solution 1 : Établissement d'un tunnel IPsec

Dans le cas où la méthode EAP utilisée génère une clé secrète à la fin de la phase d'authentification, le serveur AAA envoie cette clé au NAS afin qu'il l'utilise pour établir un tunnel IPsec avec l'utilisateur [30].

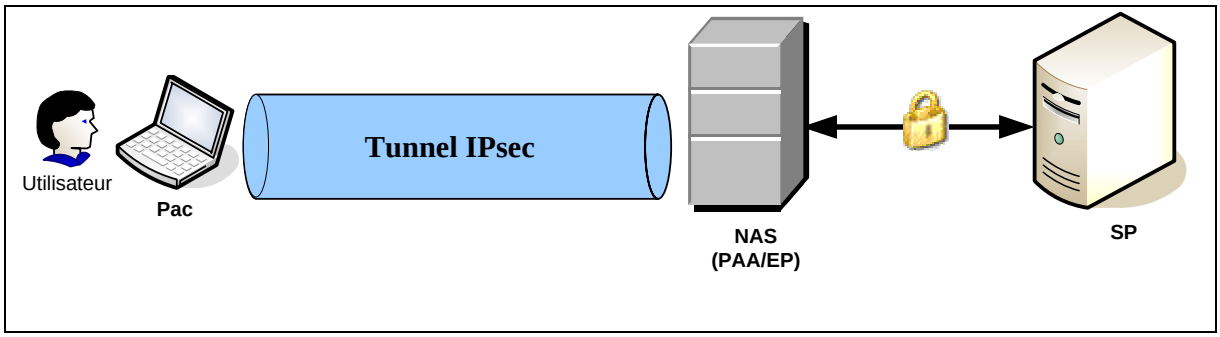

Figure 25. Établissement d'un tunnel IPsec entre un pair et le NAS

L'établissement du tunnel IPsec entre le Pac et le NAS et non entre le Pac et le SP permet de protéger le fournisseur de service contre les attaques par usurpation d'identité (ces attaques seront détectées et bloquées au niveau du NAS) et de le décharger des mécanismes de chiffrement et de hachage nécessaires au protocole IPsec.

Cette solution nécessite un traitement énorme au niveau des pairs (définition des associations de sécurité, application de plusieurs fonctions de chiffrements et de hachage, génération des clés dans chaque nouvelle session,…). Ainsi, elle ne peut pas être adoptée dans le système DisPairSe où les nœuds peuvent être de faible puissance (PDA,…).

Solution 2 : Utilisation d'un secret partagé

Cette solution consiste à utiliser le secret de l'utilisateur pour sécuriser ses communications avec le SP comme suit :

$$Message' = E_{secret}(numéro\_de\_séquence \ o \ Message \ o \ H(Message))$$

E : une fonction de chiffrement symétrique

numéro_ de_séquence : correspond au numéro du message, il permet d'éviter les attaques par rejeu.

Message : le contenu initial du message à envoyer

Message' : le message à envoyer (le contenu final)

H : une fonction de hachage

La robustesse de cette solution se base sur le secret de l'utilisateur. Pour cela, il faut être prudent dans sa génération, par exemple nous pouvons délivrer ce secret à partir du mot de passe de l'utilisateur ou bien prendre la première clé secrète générée par la méthode EAP (après la première authentification).

Dans notre implémentation, nous avons appliqué cette deuxième solution. En effet, elle n'a pas de paramètres de sécurité à recalculer après chaque réauthentification (comme les clés de session et les associations de sécurité dans la première solution). En plus la protection des messages échangés est assurée tout simplement par une fonction de hachage et une méthode de chiffrement symétrique, ce qui est adapté pour les pairs de faible puissance.

6 Conclusion

Le fonctionnement d'un service de sauvegarde distribuée se base principalement sur la collaboration de ses nœuds, mais ceci ne suffit pas à garantir le bon fonctionnement du système. Pour résoudre ce problème, nous avons introduit les services AAA dans le fonctionnement du système DisPairSe. En effet, à travers ses services, nous avons pu contrôler l'accès au service de sauvegarde et inciter les pairs à offrir une partie de leurs ressources au système.

Dans le chapitre suivant, nous présentons les autres solutions que nous avons proposées pour assurer le bon fonctionnement du système DisPairSe et garantir une bonne qualité de service aux usagers.

VI PROCEDURES DE CONTROLE DANS DISPAIRSE

1 Introduction

Plusieurs systèmes de sauvegarde distribuée ont été élaborés, plus ou moins différents sur certains aspects de leur architecture. Cependant, peu ont atteint le stade du prototype et aucun ne l'a dépassé pour être exploité en conditions réelles. Dans le nouveau système DisPairSe, nous avons évoqué et résolu plusieurs problèmes tels que : assurer entièrement l'anonymat des utilisateurs, introduire les services AAA, assurer une bonne qualité de service, contrôler le comportement des nœuds,… et ceci afin de motiver son utilisation.

Dans ce chapitre, nous présentons les techniques que nous avons proposées pour contrôler le fonctionnement du système et assurer une bonne qualité de service aux usagers. Tout d'abord, nous décrivons les services que nous avons introduits afin de contrôler le comportement des différents nœuds au cours du fonctionnement du système DisPairSe. Puis, nous présentons la méthode de facturation que nous avons proposée et qui oblige les nœuds à bien respecter le fonctionnement du système. Enfin, nous montrons que même avec tous les contrôles effectués, le fonctionnement du service de sauvegarde dans DisPairSe reste toujours autonome.

2 Contrôle du comportement des pairs

Pour contrôler le comportement des pairs au cours du fonctionnement du système DisPairSe, nous avons introduit les trois services suivants :

- ***Service de contrôle (S1) :*** il compare les informations déclarées par les propriétaires des fichiers et celles définies par les nœuds de sauvegarde afin de déterminer les comportements malveillants des pairs.
- ***Service de test de disponibilité (S2) :*** il vérifie périodiquement la disponibilité d'un ensemble de fragments choisis aléatoirement.
- ***Service de notation (S3) :*** il attribue un niveau de fiabilité pour chaque pair.

Ces nouveaux services seront déployés dans le SP. De cette façon, nous équilibrons les charges entre le fournisseur de service et le serveur AAA (cf. Figure 26).

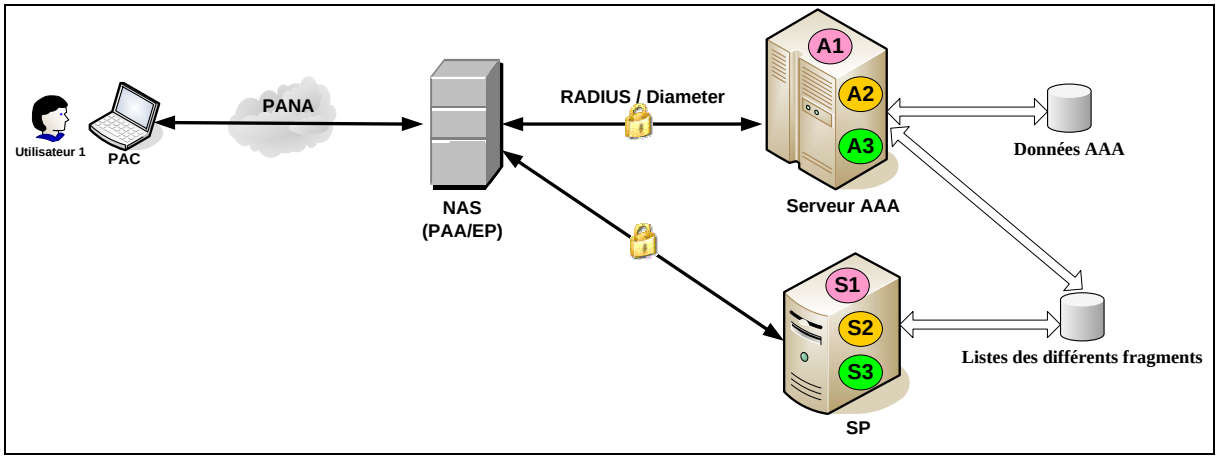

Figure 26. Problème après la modification de la politique de respect de la vie privée

A1 : Service d'authentification

A2 : Service d'autorisation
A3 : Service de comptabilisation

Dans les paragraphes suivants nous décrivons le principe de fonctionnement de chacun de ces trois nouveaux services.

2.1 Service de contrôle

Ce service analyse périodiquement l'ensemble des informations collectées par le fournisseur de service (liste des fragments créés, liste des fragments acquis par les nœuds de sauvegarde,…) afin de découvrir les incohérences entre les déclarations des propriétaires des fichiers et celles des nœuds de sauvegarde. À chaque type d'incohérence correspond, en fait, un comportement malveillant de l'un des nœuds du système.

Parmi les comportements malveillants qui peuvent être découverts par ce service, on peut citer :

- la modification de la longueur réelle d'un fragment
- la sauvegarde d'un fragment dans le réseau P2P sans le déclarer au fournisseur de service
- la déclaration double de l'acquisition d'un même fragment, et ceci afin d'obtenir une double rémunération
- la déclaration de l'acquisition d'un fragment fictif, c'est-à-dire d'un fragment qui n'a été créé par aucun nœud

2.2 Service de test de disponibilité

Ce service permet de vérifier périodiquement la disponibilité d'un sous-ensemble aléatoire de fragments.

Le test de disponibilité d'un fragment peut être réalisé par l'une des méthodes suivantes :

Solution 1 : *Communication directe entre le fournisseur de service et le nœud de sauvegarde*

Dans cette solution, le fournisseur de service envoie directement une demande de récupération d'un fragment à son nœud de sauvegarde, c'est-à-dire sans passer par la phase de recherche du protocole P2P « Pastry » (cf. Figure 27).

Cette solution exige que le fournisseur de service connaisse la clé DHT et le nœud de sauvegarde (son adresse IP et son port) de chaque fragment sauvegardé dans le réseau P2P.

Solution 2 : *Le fournisseur de service est un nœud du réseau P2P*

Dans cette solution, le fournisseur de service fait partie du réseau P2P et donc il peut récupérer n'importe quel fragment du réseau sans besoin de connaître auparavant l'adresse IP de son nœud de sauvegarde (cf. Figure 28). La localisation du nœud de sauvegarde est effectuée par le protocole P2P.

Solution 3 : *Le fournisseur de service charge un nœud de confiance de vérifier la disponibilité d'un ensemble de fragments*

Dans cette solution, le SP choisit le nœud le plus sincère du réseau P2P (en se basant sur le service de notation), et il le charge de vérifier la disponibilité d'un ensemble de fragments (cf. Figure 29). Le nœud sélectionné sera rémunéré après l'accomplissement du travail demandé.

Le tableau 2 présente une comparaison entre ces trois solutions.

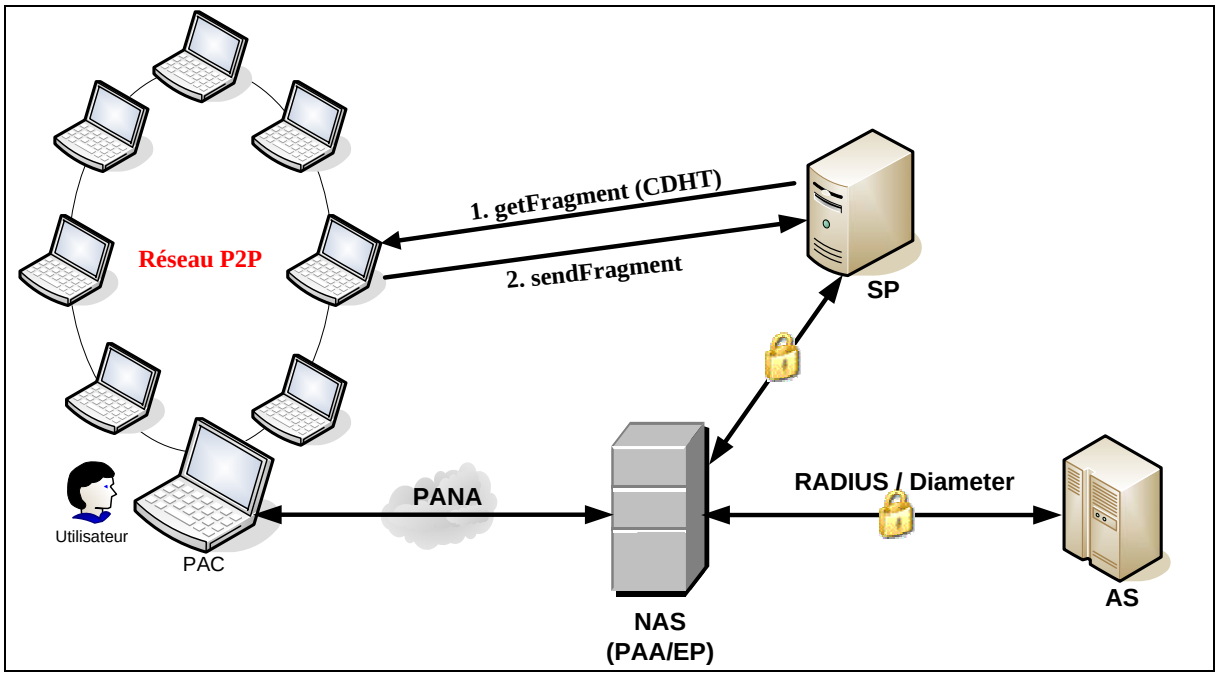

Figure 27. Le SP communique directement avec le nœud de sauvegarde

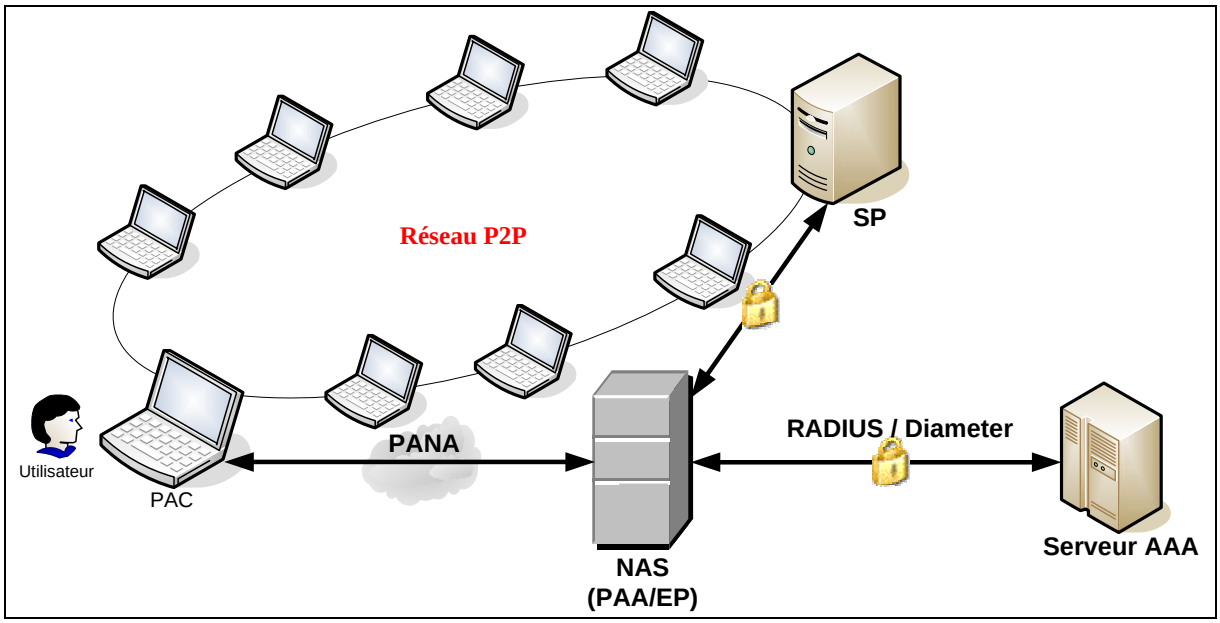

Figure 28. Le SP est un nœud du réseau P2P

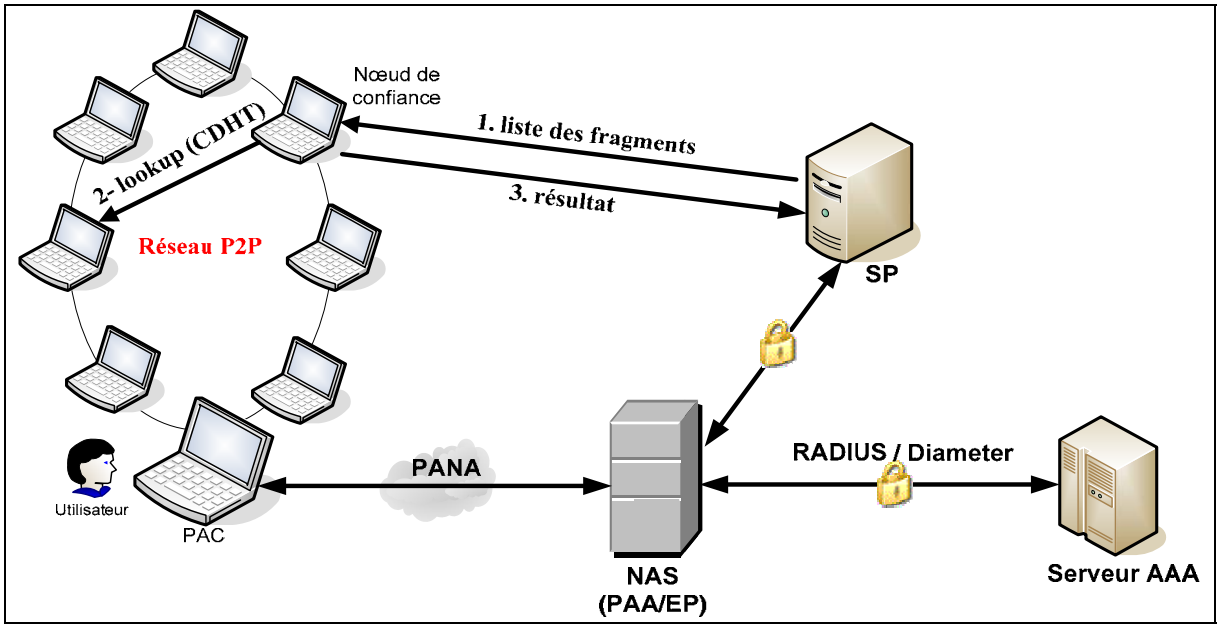

Figure 29. Le SP charge un nœud de vérifier la disponibilité d'un ensemble de fragments

	Solution 1	Solution 2	Solution 3
Rapidité	rapide : il n'y a pas une phase de recherche de la localisation d'un nœud de sauvegarde	lente : il faut passer par le protocole « Pastry » pour déterminer la localisation d'un nœud de sauvegarde	très lente : le test de disponibilité d'un ensemble de fragments est réalisé en trois phases
Fiabilité	fiable : le test de disponibilité d'un fragment est réalisé sans faire intervenir les nœuds du réseau P2P	moins fiable : le test de disponibilité d'un fragment fait intervenir les nœuds du réseau P2P	non fiable : le test de disponibilité d'un fragment fait intervenir les nœuds du réseau P2P et encore, il se base sur la fiabilité du nœud de confiance choisi
Performance	performante : il n'y a pas de charges supplémentaires pour le fournisseur de service	non performante : en tant que nœud du réseau P2P, le fournisseur de service doit participer à la recherche et la sauvegarde des fragments	performante : il n'y a pas de charges supplémentaires pour le fournisseur de service
Sécurité	sécurisée : le fournisseur de service n'accepte pas de trafic P2P	non sécurisée : le fournisseur de service est plus facilement accessible depuis le réseau public puisqu'il accepte du trafic P2P. Il est donc de ce fait plus vulnérable	sécurisée : le fournisseur de service n'accepte pas de trafic P2P

Tableau 2. Comparaison entre les trois procédures de fonctionnement du service de test de disponibilité

Dans notre implémentation, nous avons adopté la première solution puisqu'elle est la plus rapide, la plus simple et la plus fiable. De plus, dans notre cas, le fournisseur de service a toutes les informations nécessaires sur les différents fragments sauvegardés dans le réseau P2P (clé DHT, nœud de sauvegarde,…), ainsi que les adresses IP des différents nœuds (puisqu'il est en communication périodique avec eux).

2.3 Service de notation

Le but de ce service est d'attribuer un niveau de fiabilité à chaque nœud du réseau P2P en se basant sur les informations fournies par le service de test de disponibilité et les comportements malveillants découverts par le service de contrôle.

Initialement, chaque nœud est considéré comme fiable (son niveau de fiabilité vaut 1). Après chaque détection de comportement malveillant, son niveau de fiabilité décroît.

Le nœud qui obtient un niveau de fiabilité égal à 0 est considéré comme totalement non fiable et par la suite il ne sera pas payé sur les fragments qu'il sauvegarde (voir la formule de facturation).

Le niveau de fiabilité d'un nœud est toujours gardé entre 0 et 1.

Les résultats de ce service seront exploités par le service de facturation car le prix d'un octet d'espace disque fourni au système dépend de la fiabilité de son propriétaire : plus le nœud est fiable, plus son espace disque est cher.

3 Encouragement des nœuds de confiance et pénalisation des nœuds malicieux

Le rôle du service de comptabilisation déployé par le serveur AAA est d'attribuer à chaque utilisateur une facture débitrice lorsqu'il consomme plus de ressources qu'il n'en fournit ou une rémunération dans le cas contraire. Nous avons exploité, en plus, ce service afin d'encourager les nœuds de confiance et pénaliser les nœuds malicieux. En effet, dans la formule de facturation appliquée pour un pair (Formule 3), le prix d'un octet d'espace disque fourni au système DisPairSe est proportionnel à son niveau de fiabilité. De cette façon, chaque pair sera obligé d'éviter tout comportement malveillant afin de garder un niveau de fiabilité élevé et donc d'augmenter le prix de son espace disque. Par conséquent, nous garantissons le bon fonctionnement du système et une bonne qualité de service aux utilisateurs.

$$\text{Somme} = \text{Somme} + \Big[\Big(\text{espace\_disque\_fourni} * \overbrace{\text{prix\_un\_octet} * \text{niveau\_fiabilité}}^{\text{prix d'un octet fourni au système}} \Big) - \big(\text{espace\_disque\_consommé} * \text{prix\_un\_octet} \big) \Big]$$

Formule 3. Formule de facturation

Cette formule est appliquée pour chaque nœud et après chaque période T, avec :

- **espace_disque_fourni** : le volume de l'espace disque (exprimé en octets) mis à disposition par le nœud pour le service de sauvegarde
- **espace_disque_consommé** : le volume de l'espace disque (exprimé en octets) consommé par le nœud
- **Somme** : le montant accordé au nœud ; s'il est positif, il s'agit d'une rémunération et s'il est négatif, il s'agit du montant à payer
- **prix_un_octet** : le prix de consommation ou de fourniture d'un octet pendant une durée T
- **niveau_fiabilité** : le niveau de fiabilité du nœud, c'est un réel compris entre 0 et 1

4 Fonctionnement autonome du service de sauvegarde

Dans le système DisPairSe, il existe deux types d'échanges de données :

- *échange entre un nœud et le fournisseur de service* : c'est un échange périodique concernant les informations nécessaires pour le contrôle et la facturation (liste de fragments créés, liste des fragments acquis,…). Pour participer à ce type d'échange un nœud doit s'authentification auprès du serveur AAA.
- *échange entre les nœuds* : cet échange concerne les données et les métadonnées des fichiers (les fragments FB et FBL). Ce type d'échange n'exige aucune authentification.

Le fait de garder la communication entre les pairs sans aucune authentification permet d'augmenter la disponibilité du service de sauvegarde et d'assurer son autonomie (cf. Figure 30). En effet, un pair peut toujours récupérer ses données et sauvegarder de nouveaux fichiers dans le réseau P2P même en cas d'indisponibilité du système administratif (indisponibilité du SP ou du serveur d'authentification,…), la seule différence c'est que le contrôle effectué par le fournisseur de service sera retardé jusqu'à que le système administratif soit de nouveau accessible.

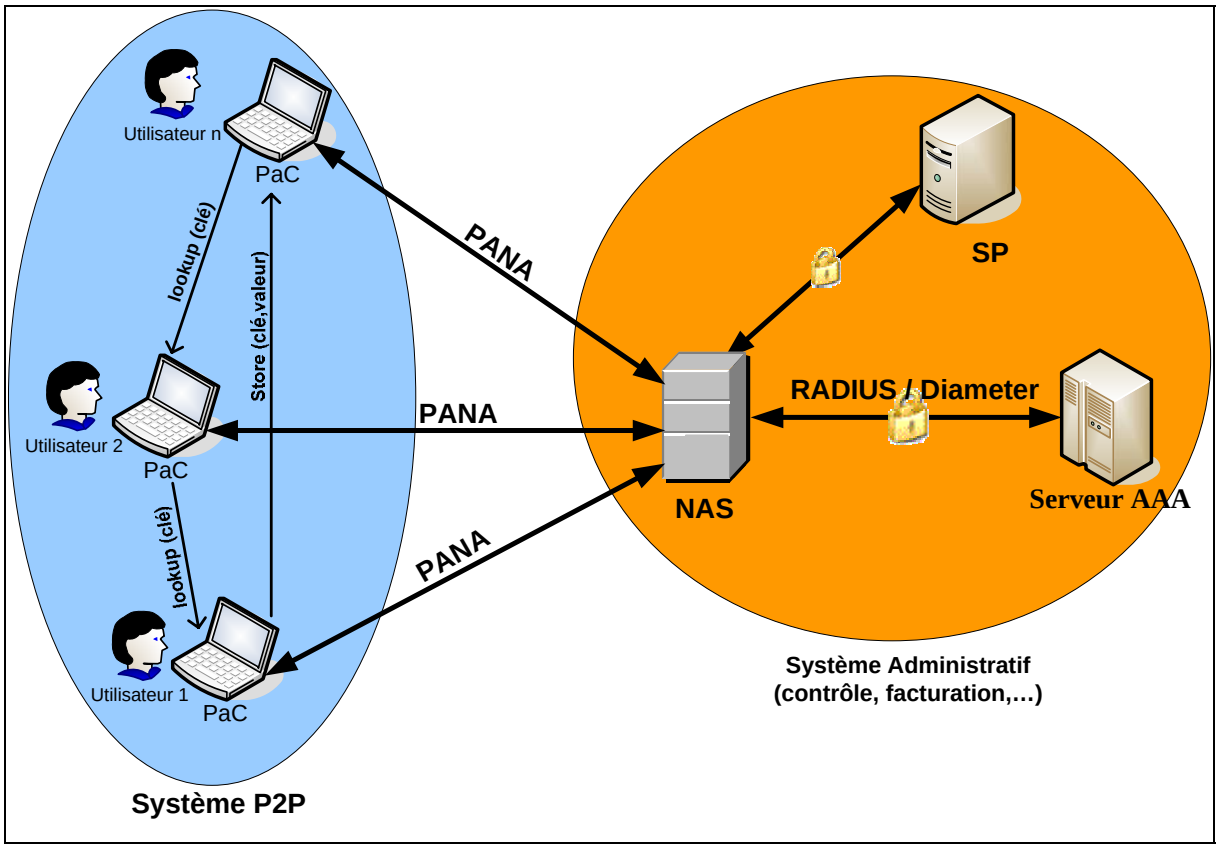

Figure 30. Fonctionnement autonome du système de sauvegarde

5 Conclusion

Un réseau P2P se caractérise par l'absence de confiance entre ses nœuds et les comportements malveillants de certains pairs. Pour cela nous avons introduit dans ce chapitre des services supplémentaires permettant de contrôler le comportement des différents nœuds au cours du fonctionnement du système DisPairSe. De plus, nous avons exploité le service de comptabilisation pour obliger les pairs à bien respecter les règles de fonctionnement du système. À la fin du chapitre, nous avons montré que malgré le caractère centralisé de la plupart de nos solutions de contrôle, le fonctionnement du service de sauvegarde reste toujours autonome.

CONCLUSIONS

Le modèle P2P distribué offre de nombreux avantages comparativement à un modèle classique client-serveur. En effet, contrairement à un système centralisé où les capacités et les ressources sont très limitées, un système P2P peut contenir un très grand nombre de pairs, ce qui offre de grandes capacités (capacité de stockage, capacité de calcul, …) aux utilisateurs. Parmi les nouvelles utilisations du modèle P2P, on trouve les systèmes de sauvegarde distribuée qui permettent à un utilisateur de sauvegarder une copie de ses données dans les autres pairs du réseau P2P afin de les récupérer en cas de perte de ses données locales ou en cas de crash de son disque dur. L'utilisation d'un modèle P2P dans un service de sauvegarde distribuée permet à un utilisateur de bénéficier d'une grande capacité de stockage, mais en contrepartie, l'absence de relations de confiance entre les différents nœuds du système pose les questions suivantes : Comment protéger les données d'un utilisateur ? Comment assurer son anonymat ? Comment éviter les nœuds égoïstes ?, Comment contrôler l'accès au système ?… Dans le cadre du projet DisPairSe, nous avons résolu l'ensemble de ces problèmes.

Tout d'abord, nous avons défini la structure d'un fichier sauvegardé dans le réseau P2P et les traitements de sécurité à appliquer sur chaque type de donnée afin d'assurer son intégrité et sa confidentialité. Dans la plupart des solutions proposées, nous avons exploité les caractéristiques d'une table de hachage distribuée et nous avons associé à chaque utilisateur une information propre et secrète appelée son « secret ». En plus, nous avons introduit une entité de confiance appelée le « fournisseur de service » qui se charge de contrôler le bon fonctionnement du système.

Dans le chapitre 3, nous avons défini le principe de fonctionnement du système DisPairSe. En effet, nous avons décrit les processus suivis par un utilisateur pour sauvegarder, modifier, restaurer ou supprimer un fichier du réseau P2P. Dans ces différentes opérations, nous avons toujours respecté l'anonymat des utilisateurs et nous avons permis au fournisseur de service de collecter toutes les informations nécessaires pour assurer la fonction de comptabilisation et contrôler le comportement des différents nœuds.

Dans le chapitre 4, nous avons décrit l'architecture que nous avons proposée afin de protéger le fournisseur de service contre les attaques de déni de service et d'introduire les services AAA (**A**uthentication, **A**uthorization, **A**ccounting) dans le système DisPairSe. À travers ces trois services, nous avons résolu les problèmes de contrôle d'accès et de nœuds égoïstes. En effet, dans le principe de fonctionnement du système DisPairSe, nous avons pu rendre l'accès au fournisseur de service indispensable pour qu'un nœud du réseau P2P puisse bénéficier du service offert. Ainsi, chaque pair est obligé de s'authentifier auprès du serveur AAA afin de pouvoir accéder au fournisseur de service et donc au service de sauvegarde distribuée. De plus, le service de comptabilisation permet d'attribuer à chaque utilisateur une facture débitrice lorsqu'il consomme plus de ressources qu'il n'en fournit ou bien une rémunération dans le cas contraire. Tout ceci évite que des nœuds égoïstes ne puissent profiter des ressources du système tout en refusant d'y contribuer.

Comme un réseau P2P peut contenir plusieurs nœuds malicieux, nous avons introduit dans le dernier chapitre des nouveaux services permettant de contrôler les comportements des différents pairs au cours du fonctionnement du système DisPairSe. De plus, nous avons proposé une solution permettant d'obliger un pair à bien respecter les règles de fonctionnement du système. Cette solution consiste à attribuer à chaque pair un niveau de fiabilité et à le faire intervenir dans le

service de comptabilisation de telle façon que le prix d'un octet d'espace disque offert au système soit proportionnel au niveau de fiabilité de son propriétaire.

BIBLIOGRAPHIE

[1] https://dispairse.point6.net

[2] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu, "Peer-to-peer computing", Technical Report HPL-2002, HP Laboratories, 2002.

[3] K. Aberer and M. Hauswirth, "Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems", Tutorial ICDE'2002, Distributed Information Systems Laboratory (LSIR), 2002.

[4] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", September 2001.

[5] R. Rivest, "The MD5 Message-Digest Algorithm", April 1992.

[6] G. Doyen, "Supervision des réseaux et services pair à pair", Thèse de l'université Henri Poincaré – Nancy 1, décembre 2005.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, "Chord : A scalable peer-to-peer lookup service for internet applications", MIT Laboratory for Computer Science, 2001.

[8] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy : A scalable and dynamic emulation of the Butterfly", 2002.

[9] A. Rowstron and P. Druschel, "Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems", Microsoft Research, 2001.

[10] J. Paschoud, S. Baehni and R. Guerraoui "Implémentation de Pastry dans le framework Borrow/Lend", juin 2005.

[11] C. Batten, K. Barr, A. Saraf and S. Trepetin, "pStore: A secure peer-to-peer backup system", Technical Memo MIT-LCS-TM-632, Massachusetts Institute of Technology Laboratory for Computer Science, October 2002.

[12] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon and M. Theimer, "Reclaiming Space from Duplicate Files in a Serverless Distributed File System", Technical Report MSR-TR-2002-30, July 2002.

[13] J. Cooley, C. Taylor and A. Peacock, "ABS: the apportioned backup system. Technical report", IT Laboratory for Computer Science, 2004.

[14] A. Tridgell, "Efficient Algorithms for Sorting and Synchronization", PhD thesis, The Australian National University, 1999.

[15] A. Muthitacharoen, B. Chen and D. Maziéres. "A low-bandwidth network file system", In Proceedings of the 18th ACM Symposium on Operating Systems Principles, 2001.

[16] L.P. Cox, C.D. Murray and B.D. Noble, "Pastiche: Making backup cheap and easy", In Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002.

[17] V. Henson, "An analysis of compare by hash", in 9th Workshop on Hot Topics in Operating Systems (HotOS IX), (Lihue Hawaii), Sun Microsystems, May 2003.

[18] http://rsync.samba.org/

[19] M. Landers, H. Zhang and K-L. Tan, "PeerStore : Better Performance by Relaxing in Peer-to-Peer Backup", Proceedings of the Fourth International Conference on PeertoPeer Computing, 2004.

[20] I. Clarke, O. Sandberg, B. Wiley, T. W. Hong, "Freenet : A Distributed Anonymous Information Storage and Retrieval System", July 2000.

[21] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J. R. Lorch, M. Theimer, R. P. Wattenhofer, "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment", Microsoft Research, December 2002.

[22] A. Yegin, Y. Ohba, R. Penno, G. Tsirtsis, C. Wang, "Protocol for Carrying Authentication for Network Access (PANA) Raquirements", May 2005.

[23] IEEE Standard for Local and metropolitan area networks: Port-Based Network Access Control, December 2004.

[24] IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements, June 2007.

[25] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson and H. Levkowetz, "Extensible Authentication Protocol EAP)", June 2004.

[26] P. Jayaraman, R. Lopez, Y. Ohba, M. Parthasarathy and A. Yegin "Protocol for Carrying Authentication for Network Access (PANA) Framework ",draft-ietf-pana-framework-10, September 2007.

[27] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", draft-ietf-pana-pana-18, September 2007.

[28] C. Rigney, "RADIUS Accounting", June 2000.

[29] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence, "Generic AAA Architecture", August 2000.

[30] M. Parthasarathy, "PANA Enabling IPsec based Access Control", draft-ietf-pana-ipsec-07**,** July 2005.

GLOSSAIRE

802.1X

Norme de l'IEEE pour le contrôle d'accès au réseau basé sur les ports. Dans cette norme, l'authentification des utilisateurs est assurée par une méthode EAP et en utilisant un serveur RADIUS.

802.11

Norme conçue par l'IEEE pour les réseaux locaux sans fil (les réseaux LAN et MAN). Elle donne des spécifications relatives à l'implémentation de la couche physique et de la sous-couche MAC (Couche liaison de données du modèle OSI) pour les réseaux locaux à liaison sans fil.

3DES (Triple Data Encryption Standard)

Algorithme de chiffrement symétrique enchaînant 3 applications successives de l'algorithme DES sur le même bloc de données de 64 bits, avec 2 ou 3 clés DES différentes.

AES (Advanced Encryption Standard)

Algorithme de chiffrement symétrique utilisant des clés de 128, 192 ou 256 bits.

API (Application Programming Interface)

Interface de programmation.

Association de sécurité (SA)

Accord entre deux ou plusieurs entités sur les traitements de sécurité à appliquer afin de sécuriser la communication entre eux.

Déni de service

Attaque sur un serveur informatique ou sur un réseau destinée à l'empêcher de remplir sa fonction (retard dans le traitement des requêtes, injoignabilité, impossibilité d'accès même pour les utilisateurs autorisés).

DES (Data Encryption Standard)

Algorithme de chiffrement symétrique utilisant des clés de 56 bits.

DHCP (Dynamic Host Configuration Protocol)

Protocole permettant de configurer des machines et notamment leur adresse IP.

DHT (Distributed Hash Table)

Infrastructure P2P de routage et de localisation qui repose sur une table de hachage distribuée.

EAP (Extensible Authentification Protocol)

Protocole très générique permettant l'identification d'utilisateurs selon diverses méthodes (mot de passe, certificat, carte à puce).

IP (Internet Protocol)

Protocole de niveau 3 du modèle OSI (niveau 2 du modèle TCP/IP) permettant un service d'adressage unique pour l'ensemble des terminaux connectés.

MD5 (Message Digest 5)
Fonction de hachage cryptographique.

P2P (Pair à Pair ou Peer-to-Peer)
Modèle distribué où les entités appelées pairs jouent le double rôle de client et serveur.

PANA (Protocol for Carrying Authentication and Network Access)
Protocole agissant au dessus de la couche IP et permettant de transporter les messages d'authentification, typiquement des messages EAP, entre un client et un agent (appelé agent PANA).

PDA (Personal Digital Assistant)
Ordinateur de poche ou assistant personnel.

RADIUS (Remote Authentification Dial-In-User Service)
Protocole de type AAA.

Rejeu
Action consistant à envoyer un message intercepté précédemment, en espérant qu'il sera accepté comme valide par le destinataire une autre fois.

SHA1 (Secure Hash Algorithm)
Une fonction de hachage cryptographique conçue par la National Security Agency des États-Unis (NSA).

SNMP (Simple Network Management Protocol)
Protocole dédié à la gestion de réseau. Il est standardisé par l'IETF.

TTL (Time To Live)
Temps de vie d'un paquet émis sur un réseau.