# Sauvegarde coopérative entre pairs pour dispositifs mobiles*

Ludovic Courtès Marc-Olivier Killijian David Powell Matthieu Roy

*prénom.nom*@laas.fr
LAAS-CNRS
7 Avenue du Colonel Roche
31077 Toulouse cedex 4
France

**Résumé/Summary**

Nous présentons les fonctionnalités d'un système de sauvegarde coopérative pour dispositifs mobiles. Ce système repose sur la collaboration entre dispositifs pour assurer la sauvegarde et le recouvrement des données de chaque dispositif. Nous identifions les propriétés de sûreté de fonctionnement que l'on est en mesure d'attendre d'un tel système. Nous analysons les systèmes de sauvegarde coopérative pair-à-pair décrits dans la littérature afin d'identifier d'éventuelles fonctionnalités transposables à l'environnement mobile. Enfin, nous concluons sur les spécificités de cet environnement et identifions les axes de recherche à explorer.

**Mots-clef :** Sauvegarde, dispositifs mobiles, coopération, pair-à-pair.

We present the features of a collaborative backup system for mobile devices. This system relies on collaboration among peers in order to provide data backup and recovery. We identify the expected dependability properties for such a system. We then survey peer-to-peer collaborative backup systems described in the literature and identify mechanisms relevant to the mobile environment. We conclude on the specificities of this environment and identify future research directions.

**Keywords :** Mobile applications, data back-up, collaboration, peer-to-peer.

* Ces travaux s'intègrent dans le cadre du projet MoSAIC, partiellement financé par l'Action Concertée Incitative (ACI) Sécurité & Informatique du Ministère délégué à la Recherche et aux Nouvelles Technologies.

# Sauvegarde coopérative entre pairs pour dispositifs mobiles*

Ludovic Courtès Marc-Olivier Killijian David Powell Matthieu Roy

*prénom.nom*@laas.fr
LAAS-CNRS
7 Avenue du Colonel Roche
31077 Toulouse cedex 4
France

## RÉSUMÉ

Nous présentons les fonctionnalités d'un système de sauvegarde coopérative pour dispositifs mobiles. Ce système repose sur la collaboration entre dispositifs pour assurer la sauvegarde et le recouvrement des données de chaque dispositif. Nous identifions les propriétés de sûreté de fonctionnement que l'on est en mesure d'attendre d'un tel système. Nous analysons les systèmes de sauvegarde coopérative pair-à-pair décrits dans la littérature afin d'identifier d'éventuelles fonctionnalités transposables à l'environnement mobile. Enfin, nous concluons sur les spécificités de cet environnement et identifions les axes de recherche à explorer.

**Mots-clef :** Sauvegarde, dispositifs mobiles, coopération, pair-à-pair.

## SUMMARY

We present the features of a collaborative backup system for mobile devices. This system relies on collaboration among peers in order to provide data backup and recovery. We identify the expected dependability properties for such a system. We then survey peer-to-peer collaborative backup systems described in the literature and identify mechanisms relevant to the mobile environment. We conclude on the specificities of this environment and identify future research directions.

**Keywords :** Mobile applications, data back-up, collaboration, peer-to-peer.

## 1. INTRODUCTION

Nous abordons la problématique liée à la conception d'un intergiciel fournissant des mécanismes de sûreté de fonctionnement à des dispositifs mobiles dotés de moyens de communication sans fil. Nous considérons des dispositifs très dynamiques et mobiles, n'ayant accès à une infrastructure fixe de communication (un réseau local ou Internet) que par intermittence. Ces dispositifs mobiles doivent être capables de communiquer entre eux, lorsqu'ils sont à proximité physique, en utilisant des moyens de communication *ad hoc* à un saut ou à plusieurs sauts. Cependant, nous souhaitons que l'intergiciel développé, MoSAIC[1] soit utile à une large palette de systèmes, allant aussi bien de dispositifs très mobiles n'ayant que rarement accès à Internet, à l'autre extrême représenté par des machines connectées en permanence à Internet et peu ou pas mobiles. Il est important de noter que dans ce schéma, nous faisons l'hypothèse que les participants à ce service n'ont aucune relation de confiance au préalable.

MoSAIC a pour objectif de permettre aux dispositifs sur lesquels il s'exécute de tolérer les fautes logicielles ou matérielles pouvant entraîner la perte de données. Ces fautes sont le plus souvent permanentes : perte ou vol du dispositif mobile, effacement accidentel de données par l'utilisateur. Pour résister à des fautes permanentes, notre intergiciel doit fournir les moyens de sauvegarder les données de l'utilisateur sur un périphérique tiers, en utilisant les moyens de communication dont il dispose.

À l'heure actuelle, les utilisateurs de systèmes mobiles, qu'il s'agisse d'ordinateurs portables (*laptop*) ou d'assistants numériques personnels (PDA), effectuent le plus souvent la sauvegarde de leurs données lorsqu'ils ont accès à leur machine de bureau en synchronisant les données entre les deux machines. Typiquement, les premières générations d'assistants personnels avaient pour seul moyen de communication un système de courte portée, généralement un câble série ou un port infrarouge, demandant à l'utilisateur de se trouver à proximité de la machine sur laquelle s'effectue la sauvegarde. Les dispositifs mobiles actuels disposent généralement de plusieurs interfaces de communication (par exemple IEEE 802.11, Bluetooth). Lorsqu'une infrastructure réseau est disponible dans leur environnement, par exemple des points d'accès Wifi, ces dispositifs mobiles peuvent avoir l'occasion de se connecter à leur machine de bureau pour y sauvegarder leurs données.

Dans la pratique, il est pourtant rare que cette possibilité soit utilisée pour effectuer à distance des sauvegardes, et ce pour plusieurs raisons :

* Ces travaux s'intègrent dans le cadre du projet MoSAIC, partiellement financé par l'Action Concertée Incitative (ACI) Sécurité & Informatique du Ministère délégué à la Recherche et aux Nouvelles Technologies.

[1] *Mobile System Availability, Integrity and Confidentiality*, *http://www.laas.fr/mosaic/*.

- elle nécessite que la machine distante soit en fonctionnement, connectée à Internet et accessible ;
- l'accès à une infrastructure par des moyens de communication sans fil est encore rare (et cher) et il peut s'écouler longtemps avant que le système mobile soit en mesure de se connecter à Internet ;
- synchroniser toutes les données qui ont été modifiées depuis la dernière synchronisation peut s'avérer long à cause de la relative lenteur d'un tel accès réseau ;
- enfin, à notre connaissance, les logiciels capables de tenter automatiquement une sauvegarde sur la machine de bureau sont encore rares.

Une autre solution consiste à utiliser les services d'un tiers de confiance qui garantit la disponibilité de ses serveurs de sauvegarde. Différentes offres commerciales permettent de sauvegarder ses données moyennant un abonnement annuel pour des capacités de stockage souvent limitées.

En revanche, nous pensons que l'avènement de dispositifs mobiles équipés de moyens de communication sans fil de courte portée offre des possibilités d'interactions *entre pairs* dont pourrait profiter un système de sauvegarde coopératif. En effet, l'utilisation répandue de systèmes mobiles communicant va permettre des interactions fréquentes mais de courte durée. Avec MoSAIC nous souhaitons tirer profit de ces interactions : à chaque rencontre de deux systèmes mobiles, le système de sauvegarde va automatiquement initier une demande de sauvegarde pour une partie de ses données. En contrepartie, il devra rendre le même service à la communauté. Pour être pratique, ces transactions devront s'effectuer *automatiquement*, sans aucune intervention de l'utilisateur. De cette façon, chaque utilisateur pourra sauvegarder tout ou partie de ses données, et ce avec une granularité assez fine compte-tenu de la fréquence des rencontres que l'on peut envisager.

Nous présentons dans la section suivante les objectifs que devra atteindre MoSAIC, notamment en termes de sûreté de fonctionnement, en fonction des problèmes nouveaux qu'il pose. La section 3 présente les travaux décrits dans la littérature en matière de réseaux pair-à-pair, de stockage réparti, et de sauvegarde coopérative dont pourra s'inspirer notre système de sauvegarde. Enfin, la section 4 présente nos conclusions quant à l'apport des systèmes de sauvegarde pair à pair par rapport à nos objectifs ainsi que nos pistes de recherche à venir.

# 2. OBJECTIFS ET PROBLÉMATIQUE

Un mécanisme de sauvegarde est une fonctionnalité *a priori* indépendante des applications et qui s'inscrit dans une problématique orthogonale à celles-ci. Dans cette section, nous décrivons les objectifs du système de sauvegarde coopérative pour dispositifs mobiles, MoSAIC. Ses objectifs peuvent eux-mêmes être décomposés en un ensemble de fonctionnalités (sous-section 2.1) et un ensemble de caractéristiques non fonctionnelles que l'on peut attendre d'un tel système, telles que des propriétés de sûreté de fonctionnement (sous-section 2.2).

Nous utiliserons par la suite la terminologie suivante : un *propriétaire de données* est un dispositif mobile tenant le rôle de client du service de sauvegarde coopérative ; les *contributeurs* sont les dispositifs participant à ce service, c'est-à-dire les systèmes stockant des données pour des propriétaires. D'une manière générale, un *participant* est un dispositif mobile qui participe au service (contributeur) ou en profite (propriétaire).

## 2.1. Fonctions du service

Nous présentons ici les principales fonctionnalités de MoSAIC : la découverte et l'allocation de ressources effectuée lors de la création de copies, puis le recouvrement des données lors de la restauration.

### *2.1.1. Découverte et allocation de ressources*

La première nécessité pour MoSAIC est de *découvrir* l'espace de stockage à disposition dans son environnement, puis d'obtenir l'*allocation* d'espace sur le support de stockage d'un tiers pour y stocker tout ou partie des données à sauvegarder. Compte-tenu des scénarios envisagés en termes de connectivité du dispositif, allant d'un accès permanent à une infrastructure jusqu'à une communication uniquement en mode *ad hoc*, plusieurs critères vont influencer le choix de la méthode de découverte et d'allocation de ressources :

- le *coût de la communication*, qui correspond à la quantité d'énergie requise et au nombre de sauts nécessaires pour atteindre un nœud dans le cas d'un réseau *ad hoc* ; lorsqu'un accès direct à Internet est disponible, ce coût est indépendant de la distance séparant deux nœuds ; de même la quantité d'énergie requise pour la communication n'est pas un critère déterminant pour des dispositifs branchés sur le secteur ;
- la *densité* de l'environnement dans lequel évolue le dispositif mobile, c'est-à-dire le nombre de nœuds avec lesquels peut communiquer un nœud influence la probabilité de trouver de l'espace de stockage à proximité ;
- la *qualité* de la communication, avec des déconnexions fréquentes et parfois permanentes dans le cas d'un réseau *ad hoc*, ou au contraire des connexions stables et quasi permanentes entre deux nœuds communiquant directement *via* une infrastructure, influence la volatilité des connexions et la granularité des fragments à sauvegarder.

Ces critères permettront de définir différentes stratégies de découverte et d'allocation de ressources suivant les scénarios considérés.

### *2.1.2. Recouvrement des données*

Comme nous l'avons vu en introduction, chaque propriétaire sauvegarde ses données par fragments, au gré des rencontres qu'il fait avec des contributeurs. De ce fait, la principale difficulté de MoSAIC est de pouvoir recouvrir les données sauvegardées. Nous faisons l'hypothèse que, dans la plupart des scénarios, chaque participant aura accès de manière intermittente à Internet et qu'il pourra mettre à profit ces moments pour récupérer ses propres données (rôle du propriétaire) et surtout pour mettre à disposition les données qu'il a lui même stockées (rôle du contributeur). Une autre possibilité est que les participants n'aient jamais accès à Internet et n'établissent des connexions entre eux qu'au travers d'un réseau *ad hoc*. Dans [16], nous envisageons deux approches au recouvrement des données :

- l'approche *pro-active* (ou *push*), où le contributeur, dès qu'il a accès à Internet, envoie les données à leur propriétaire dans une « boîte aux lettres » prédéterminée ;

- l'approche *réactive* (ou *pull*), où le propriétaire va interroger le réseau de participants ; cette recherche pourrait se faire par mots clef ou méta données tels que « l'ensemble des données que j'ai sauvegardées jeudi matin ».

L'approche réactive paraît adaptée surtout à des réseaux de petite échelle. Nous reviendrons dans la section 3 sur les solutions permettant de mettre en œuvre la boîte aux lettres dont il est question dans la première approche.

## 2.2. Sûreté de fonctionnement du service

MoSAIC a pour but d'améliorer la sûreté de fonctionnement des services et données supportés par des dispositifs mobiles. Dans cette section, nous cherchons donc à définir les propriétés de sûreté de fonctionnement que l'on peut attendre de ce système.

### *2.2.1. Intégrité et cohérence des données sauvegardées*

Le système de sauvegarde doit fournir la garantie que les données restaurées par un participant sont cohérentes et intègres. Toute corruption de données sauvegardées, qu'elle soit intentionnelle ou non (par exemple due à une faute logicielle ou matérielle sur la machine stockant les données), doit pouvoir être détectée par son propriétaire lors de la restauration.

Les protocoles réseau et les supports de stockage utilisent largement des codes détecteurs d'erreur ou correcteurs d'erreur pour tolérer les fautes matérielles ou logicielles. Pour résister aux corruptions intentionnelles, il est cependant nécessaire de garantir également l'authenticité des données : le ou les propriétaires des données doivent être sûrs qu'il s'agit bien des données qu'ils ont sauvegardées. Nous présenterons en section 3.3.1 des techniques permettant d'atteindre cet objectif.

### *2.2.2. Confidentialité des données*

Comme nous l'avons vu, les participants au système de sauvegarde coopérative sont amenés à stocker leurs données sur les machines d'autres participants en lesquels ils n'ont aucune relation de confiance *a priori*. Les informations sauvegardées peuvent être sensibles et, en tant que telles, ne doivent pas pouvoir être exploitées par celui qui les stocke. La technique de fragmentation, redondance, dissémination ou FRD [10] permet précisément de garantir la confidentialité des données. Les données sont découpées en blocs puis dispersées entre différents sites de stockage de telle sorte qu'aucun site de stockage n'ait d'information sur l'origine des fragments qu'il stocke. Les données, ou leurs fragments, peuvent être chiffrés pour en améliorer la confidentialité. Bien entendu, cette fragmentation complique le recouvrement des données sauvegardées (section 2.1.2). Nous reviendrons en section 3 sur la mise en œuvre de telles techniques.

La technique de FRD a en outre l'avantage de convenir à nos hypothèses de départ : compte-tenu des interactions éphémères entre systèmes mobiles, les données doivent nécessairement être fragmentées ; de plus, la mobilité entraînera de fait une dissémination des données.

### *2.2.3. Disponibilité des données*

Il y a deux manières d'aborder la question de la disponibilité des données sauvegardées. Du point de vue du contributeur se pose la question du choix des données qu'il devra tôt ou tard effacer pour libérer son espace de stockage. Du point de vue du propriétaire, il s'agira d'avoir une assurance dans sa capacité à restaurer ses données ultérieurement. Enfin, la disponibilité des données ne doit pas être mise en danger par des malveillances tel que des attaques en déni de service.

**Choix des données à effacer.** Malgré l'utilisation d'algorithmes visant à optimiser l'utilisation de l'espace de stockage, il sera très vite nécessaire pour chaque participant d'effacer certaines données. Le mécanisme qui permettra de décider des données à effacer n'est cependant pas évident compte tenu de la contradiction entre les deux hypothèses suivantes :

1. celui le plus à même de savoir quelle version sauvegardée peut être effacée est le propriétaire des données lui-même ;

2. les participants peuvent être déconnectés et il n'existe *a priori* aucune relation de confiance entre eux.

La première hypothèse nous donne à penser que le propriétaire des données sauvegardées devrait choisir lui-même la version de ses données à effacer : il peut s'agir simplement de la plus vieille sauvegarde, ou bien d'une version plus récente mais moins importante à ses yeux parce qu'elle ne représente pas une étape importante de son travail [28]. Cependant, d'après la deuxième hypothèse formulée, un contributeur (i) ne peut pas compter sur le propriétaire des données pour le prévenir de leur péremption et (ii) ne souhaitera pas dépenser d'énergie pour essayer de reprendre contact avec le système pour lequel il sauvegarde des données.

Par conséquent, les participants *doivent* prévoir la possibilité d'une prise de décision unilatérale d'effacer des données si, par exemple, l'utilisateur des ressources n'a pas pris contact avec le contributeur *depuis un certain temps*. Une sémantique précise des obligations mutuelles devra donc être définie et nous en verrons des exemples par dans la section 3.3.3.

**Duplication.** La fréquence à laquelle un propriétaire souhaitant restaurer ses données pourra entrer en relation avec ses contributeurs a un impact direct sur la disponibilité de ses données. Pour résoudre ce problème, une solution évidente est de faire des sauvegardes redondantes, sur des participants indépendants[1], permettant ainsi de récupérer ses données malgré la défaillance d'un ou plusieurs de ses contributeurs.

**Résistance au déni de service.** Plusieurs types d'attaques par déni de service peuvent être envisagés sur le système de sauvegarde coopérative :

[1] Notons qu'il est par ailleurs nécessaire d'optimiser le volume global des données sauvegardées. En particulier, il est important d'éviter une duplication inutile des données. Nous reviendrons sur cet aspect dans la section 3.2.2.

- attaque par *égoïsme*, où un participant profite du service tout en refusant d'y contribuer ;
- attaque par *inondation*, où un propriétaire inonde de requêtes de sauvegarde les autres participants, empêchant de fait les autres propriétaires de bénéficier du service ;
- attaque par *rétention de données*, où un contributeur refuse de rendre les données qu'il stocke, intentionnellement ou pas (par exemple suite à une défaillance).

Nous reviendrons dans la section 3 sur des mécanismes largement répandus dans les systèmes pair-à-pair qui peuvent être mis en œuvre pour tolérer ce genre de comportement.

# 3. SYSTÈMES DE SAUVEGARDE PAIR-À-PAIR

Dans cette section, nous étudions dans quelle mesure l'état de l'art en systèmes de sauvegarde pair-à-pair fournirait des mécanismes aptes à faciliter la mise en œuvre d'un système de sauvegarde coopérative de données pour dispositifs mobiles.Nous donnons, dans un premier temps, une présentation rapide des différents systèmes de sauvegarde pair-à-pair étudiés, puis nous évoquons leur apport dans différents domaines par rapport à nos objectifs.

## 3.1. Systèmes étudiés

Plusieurs travaux récents traitent de la sauvegarde coopérative. Ils s'inspirent des nombreux travaux portant sur le stockage réparti de fichiers [9,18] et le partage de fichiers [2,4,26]. Tous s'intéressent à la sauvegarde coopérative pour des stations fixes, ayant un accès très régulier à Internet. Il n'existe à notre connaissance, aucun projet qui prenne en compte des systèmes mobiles, avec une connexion à une infrastructure intermittente.

Les premiers travaux décrivant un système de sauvegarde entre pairs sont ceux de Elnikety et al. [11]. Par rapport aux fonctions d'un service de sauvegarde (localisation des ressources, création de copies des données, restauration des données), ce système est assez simple. Un serveur central est utilisé pour trouver des partenaires. Aucun effort n'est fait pour ne sauvegarder que des incréments afin de réduire le temps nécessaire pour effectuer la sauvegarde ; l'intégralité des données est envoyée aux partenaires de sauvegardes. Les auteurs décrivent bon nombre d'attaques possibles dont certaines sont présentées en section 2.2. Nous reviendrons sur les autres attaques, plus spécifiques, par la suite.

Le système *Pastiche* et son extension *Samsara* [7,8], sont beaucoup plus exhaustifs. Les mécanismes de découverte de ressources, de stockage, et de localisation des données qui sont proposés sont totalement décentralisés et autogérés. Chaque nouvel entrant choisi un ensemble de *partenaires* en prenant en compte différents critères dont la latence des communications, et traite ensuite directement avec eux. Des mécanismes pour minimiser la quantité de données à échanger lors de sauvegardes successives sont également proposés. Enfin, Samsara met en œuvre une solution qui résout les problèmes de juste contribution au service et de résistance aux attaques par déni de service.

D'autres travaux cherchent à résoudre certaines limitations de Pastiche et Samsara, ou proposent de méthodes alternatives, moins complexes. C'est le cas de *Venti-DHash* [30], inspiré par le système d'archivage *Venti* [25] du système d'exploitation Plan 9. Il propose un stockage totalement réparti entre tous les participants, au lieu de sélectionner une fois pour toute un ensemble de partenaires comme le propose Pastiche. Une approche hybride à la localisation et au stockage de données est apportée par *PeerStore* [20] où chaque participant traite en priorité avec un ensemble de partenaires sélectionnés au départ (comme dans Pastiche) mais est capable de ne sauvegarder que les données n'ayant pas encore été sauvegardées. Enfin, *pStore* [1] ainsi que ABS [5] s'inspirent des systèmes de gestion des révisions pour proposer, entre autres, une meilleure utilisation des ressources.

Le système de sauvegarde coopérative *FlashBack* présenté dans [21] est particulièrement proche de nos préoccupations puisqu'il cible la sauvegarde au sein d'un réseau personnel (PAN). Cependant, de par les caractéristiques d'un tel réseau, ses objectifs diffèrent sensiblement des nôtres. Au sein d'un PAN, tous les dispositifs se font confiance puisqu'ils appartiennent à une même personne. En outre, les auteurs admettent que l'ensemble des dispositifs constitutifs d'un PAN est appelé à rester relativement constant dans le temps puisqu'il s'agit des dispositifs qu'un utilisateur transporte avec lui. Ces deux hypothèses sont en contradiction avec celles que nous faisons pour MoSAIC où aucune relation de confiance préalable n'est requise entre les participants et où on suppose une forte mobilité des dispositifs mobiles participant. Pour ces raisons, nous nous concentrons essentiellement sur les systèmes de sauvegarde pair-à-pair évoqués précédemment.

Par la suite, nous évoquerons les architectures adoptées par les différents projets, puis nous présenterons les techniques qu'ils mettent en œuvre pour atteindre certains des objectifs évoqués au chapitre précédent.

## 3.2. Aspects fonctionnels

Nous abordons successivement deux aspects spécifiques à ces systèmes de sauvegarde pair-à-pair, à savoir : les mécanismes de découverte et d'allocation de ressources, et les techniques visant à réduire une duplication inutile des données.

### *3.2.1. Découverte et allocation de ressources*

Parmi les systèmes étudiés, on retrouve deux principales approches à la répartition des blocs de données à sauvegarder :

- stockage réparti au sein de groupes particuliers de participants ou *partenaires* ;
- stockage réparti entre tous les participants au moyen d'une *tables de hachage répartie* (ou DHT) qui a pour propriété de répartir de manière homogène toutes les données sauvegardées eux ;

Dans le premier cas, les relations entre partenaires sont assez simples : chaque participant choisit, à son entrée, un ensemble de partenaires puis leur envoie directement, à chaque sauvegarde, ses données. Le système proposé par Elnikety et al. [11] se contente d'envoyer à intervalle régulier

l'ensemble des données à sauvegarder. Les autres systèmes, quant à eux, choisissent les données à transmettre en fonction des versions précédemment sauvegardées. Dans Pastiche [8], chaque participant choisi également un ensemble de partenaires qui ne changera pas ou peu par la suite. Enfin, les dispositifs participant à FlashBack [21], au sein d'un PAN, choisissent en priorité pour partenaires les dispositifs qui sont le plus souvent à proximité.

La deuxième approche repose sur une technique fondamentale des réseaux de partage des données pair-à-pair, les « *réseaux virtuels* » ou *overlay networks*, abondamment décrits dans la littérature [27]. Une DHT [26] est un mécanisme de stockage réparti de blocs de données basé sur un réseau virtuel (voir figure 1). Chaque nœud du réseau est responsable des blocs dont l'identifiant est proche (numériquement) de son identifiant. Les blocs sont donc répartis de manière homogène dans la DHT si leurs identifiants sont bien répartis. Comme dans une table de hachage, l'utilisation d'une bonne fonction de hachage pour calculer ces identifiants garantit une bonne répartition. Venti-DHash [30] et pStore [1] suivent cette approche.

L'utilisation d'une DHT pour stocker les données sauvegardées a deux inconvénients :

- le coût de la migration des données lors de l'entrée ou du départ d'un participant peut être élevé en termes d'utilisation de la bande passante [20];
- une DHT répartit automatiquement les données de manière homogène entre tous les participants, indépendamment de l'espace de stockage qu'il consomme; par conséquent, utiliser une DHT empêche d'assurer la justesse des contributions de chacun.

Pour ces raisons, PeerStore [20] propose une approche hybride où les données sont échangées directement entre partenaires, tandis que les méta-informations relatives aux blocs (en l'occurrence, les associations entre les identifiants de bloc et la liste des participants qui en stockent un exemplaire) sont stockées dans une DHT. Comme nous le verrons par la suite, cela procure en outre une plus grande flexibilité à PeerStore.

### *3.2.2. Réduction de la duplication inutile de données*

Alors qu'un certain niveau de redondance des données sauvegardées est nécessaire pour tolérer diverses fautes comme nous l'avons vu en section 2.2.3, il est nécessaire aussi de ne pas dupliquer inutilement des données entre partenaires ou participants (c'est-à-dire dans l'espace) ni entre versions successives d'un même fichier (c'est-à-dire dans le temps). Stocker de manière superflue des données entre plusieurs nœuds implique une utilisation inutilement accrue de l'espace de stockage et de la bande passante. Par conséquent, les différents systèmes étudiés utilisent des mécanismes permettant de contrôler le nombre d'instances d'un même bloc de données. On parle de *support de stockage à instance unique* ou *stockage convergent*. Cette propriété est obtenue en indexant les données en fonction de leur contenu [25].

Cette approche peut s'avérer intéressante lorsque les participants ont beaucoup de données en commun. Dans Pastiche [8], par exemple, les auteurs font l'hypothèse que les participants vont vouloir sauvegarder toutes les données présentes sur leur machine. Parmi ces données, une bonne

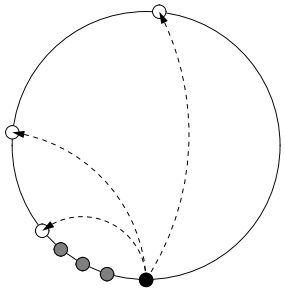

**Figure 1. Topologie de réseau virtuel en cercle dans *Chord* [9]. Le nœud noir connaît ses voisins (cercles gris) ainsi que d'autres participants éloignés (cercles blancs).**

partie du système d'exploitation et des applications a des chances d'être commune à de nombreux participants; des participants collaborant sur un même projet ont également beaucoup de données en commun. Dans cette optique, le stockage à instance unique a donc un apport considérable.

Toutefois, dans le cadre du système de sauvegarde coopérative que nous envisageons, seules les données critiques des utilisateurs, le plus souvent des données personnelles, sont à sauvegarder. On peut donc penser qu'il y aura peu ou pas de duplication inutile des données entre dispositifs mobiles participant. Le phénomène de duplication inutile dans le temps peut néanmoins être observé également dans les scénarios que nous considérons, par exemple lorsqu'un propriétaire est amené à sauvegarder plusieurs versions successives d'un même fichier auprès d'un même contributeur.

## 3.3. Sûreté de fonctionnement

Nous abordons ici les principales techniques employées dans la littérature pour garantir les propriétés de sûreté de fonctionnement que nous avons identifiées en section 2.2.

### *3.3.1. Intégrité et cohérence*

Ces propriétés de sûreté de fonctionnement sont prises en charge par l'*encodage* des fichiers à sauvegarder, c'est-à-dire par le choix des structures de données représentant un fichier. Chacun des systèmes présentés, mis à part celui d'Elnikety et al. [11], fragmente systématiquement les fichiers à sauvegarder. Cela est nécessaire pour réduire la duplication inutile des données dans l'espace et dans le temps. C'est aussi nécessaire pour assurer une répartition homogène des données sur une DHT, et permettre l'établissement de contributions justes comme nous le verrons par la suite.

pStore [1] utilise des structures de données simples pour représenter les fichiers sauvegardés. Les fichiers sont fragmentés en blocs dont la taille peut varier. Dans pStore, les participants sauvegardent en plus des blocs eux-mêmes une liste des blocs (*file block list* ou FBL) qui contient, pour chaque version du fichier considéré, la liste des identifiants des blocs qui le constituent.Cependant, chaque liste de blocs est indexée par un condensé du chemin du fichier qu'elle représente concaténé à la clef privée de son propriétaire, créant ainsi des espaces de noms de fichiers propres à chaque utilisateur, à la manière des sous-espaces dans Freenet et des « pseudonymes » dans GNUnet [2]. En pratique, pour restaurer un fichier, il faut donc connaître son chemin et disposer de la clef privée de son propriétaire. Sans cela, il est impossible d'accéder à la liste des blocs du fichier, et

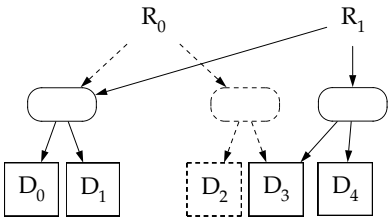


**Figure 2. Encodage de plusieurs versions d'une séquence d'octets (fichier) sous forme d'un arbre [25]. Ici, une i-node est partagée entre les deux versions du fichier (dont les racines sont $R_0$ et $R_1$). Le bloc de données $D_3$ est également partagé.**

donc à ses blocs. C'est la même technique qui est utilisée par PeerStore et une technique similaire pour Pastiche.

Dans ABS [5], chaque fragment sauvegardé est accompagné d'un bloc de méta-informations. Ces méta-informations sont celles du fichier dont provient le fragment, ainsi que la position du fragment dans le fichier. La clef sous laquelle est stockée l'ensemble est le condensé du contenu du fragment, ce qui permet de maintenir la propriété d'instance unique de chaque fragment. Les méta-informations sont chiffrées avec la clef publique du propriétaire du fichier et l'ensemble (bloc et méta-informations) est signé avec la clef privée du propriétaire. Cette signature garantit l'intégrité de l'ensemble bloc et méta-informations.

Dans Venti-DHash [30], le codage des fichiers est assuré par Venti [25]. Comme dans un système de fichiers classique, les fichiers sont représentés sous la forme d'un arbre dont les feuilles sont les blocs de données issus de la fragmentation du fichier (figure 2). Les nœuds intermédiaires sont des nœuds d'indirection (souvent appelés *i-nodes*, appelés ici *pointer blocks*) qui contiennent des pointeurs vers leurs blocs fils, c'est-à-dire les identifiants des blocs fils. Ici, tous les blocs sont indexés de la même manière, c'est-à-dire par leur condensé. De plus, tous les blocs ont la même taille et le support de stockage sous-jacent (en l'occurrence, DHash) ne connaît pas leur sémantique. Pour restaurer une version d'un fichier, il suffit de connaître l'identifiant de l'i-node racine pour cette version. Notons que contrairement à la solution proposée par pStore et Pastiche, les méta-informations permettant de reconstituer un fichier (ici les i-nodes) peuvent également être partagées. Cette technique de codage est aussi celle utilisée par GNUnet [2].

Toutes ces techniques garantissent dans une certaine mesure l'*intégrité* des données sauvegardées puisque la manière d'adresser les blocs est dépendante de leur contenu (utilisation d'un condensé). Lorsqu'un bloc est restauré, on peut donc immédiatement vérifier qu'il s'agit bien du bloc demandé. Cependant, seul Pastiche [8] aborde la question de la cohérence des données perçue par l'utilisateur telle que nous l'évoquions en section 2.2.1. Étant en partie implémenté sous la forme d'un système de fichiers, Pastiche a la possibilité de copier les blocs qui seront modifiés pendant le processus de sauvegarde[1], garantissant ainsi la cohérence de ce qui est sauvegardé.

[1] Pendant le processus de sauvegarde, les blocs sont marqués *copy-on-write*.

### 3.3.2. Confidentialité des données

L'optimisation vue en section 3.2.2 qui consiste à partager les données communes à plusieurs participants ne doit donc pas porter atteinte à la confidentialité des dites données.

La technique utilisée par tous les systèmes de sauvegarde cités et également par la plupart des systèmes de partage de fichiers est celle du *chiffrement convergent*. L'objectif est de disposer d'un méthode de chiffrement qui ne dépende pas de celui qui effectue le chiffrement. Le chiffrement convergent est donc un chiffrement symétrique dont la clef est le condensé du bloc à chiffrer. C'est ensuite ce bloc chiffré qui est stocké chez les partenaires ou autres participants. Pour désigner puis déchiffrer un bloc, il est donc nécessaire de connaître son identifiant, c'est-à-dire le plus souvent le condensé du bloc chiffré, et sa clef, c'est-à-dire le condensé du bloc en clair. Ce couple est parfois appelé *clef-condensé* ou CHK (pour *content hash key* [2]).

### 3.3.3. Disponibilité

Dans cette section, nous décrivons les techniques décrites dans la littérature pour améliorer la disponibilité des données : la duplication et le choix des données à effacer.

**Duplication.** Pour les systèmes de sauvegarde coopérative où chaque nœud choisit un ensemble de partenaires (Pastiche, PeerStore), le mécanisme de duplication est relativement simple. Dans Pastiche, chaque nouveau participant recherche 5 autres participants ayant un grand nombre de données en commun avec lui (section 3.2.2); ces 5 participants deviennent alors ses partenaires de sauvegarde. Lors du processus de sauvegarde, un nœud envoie les mêmes données (une sauvegarde incrémentale) à tous ses partenaires. Il peut donc tolérer la défaillance de 4 de ces nœuds. Dans PeerStore, le choix des partenaires s'effectue d'une manière différente. Toutefois, les auteurs précisent qu' idéalement chaque participant a autant de partenaires que d'exemplaires de ses données, ce qui nous ramène au même schéma que pour Pastiche.

Pour les systèmes basés sur une DHT, par hypothèse, l'ensemble des données stockées dans la DHT est réparti de manière homogène entre les nœuds. Par conséquent, pour tolérer le départ ou la défaillance de participants, les données stockées *doivent* être dupliquées pour ne pas être perdues. En pratique, les blocs sont généralement dupliqués par le nœud qui en est responsable sur un petit nombre de ses voisins dans l'espace des identifiants; les blocs sont également gardés en cache (donc dupliqués) par les nœuds se trouvant sur le chemin permettant d'accéder aux blocs [9,26,27]. Le coût en espace de stockage pour l'ensemble de la communauté est donc potentiellement plus élevé que dans le cas précédent.

Enfin, il est aussi possible d'utiliser des blocs de parité ajoutés aux fichiers à sauvegarder [5] ou des codes protégeant contre la perte de données (*erasure codes*) qui permettent de reconstruire un bloc simplement à partir d'un sous-ensemble des fragments de ce bloc. Venti-DHash [30] utilise cette seconde possibilité en stockant des fragments de ce bloc sur les successeurs du nœud qui en est responsable. La tolérance à la perte de fragments peut être ajustée en trouvant un compromis avec la taille de ces fragments.

ABS, où les données sont stockées dans une DHT, propose une solution alternative pour tolérer la défaillance ou le départ d'un participant. Dans ABS, les propriétaires peuvent choisir la clef sous laquelle stocker un bloc de don-

nées. En premier lieu, lors de l'insertion d'un bloc dans la DHT, les participants essayent de s'insérer en prenant pour clef son condensé. Si cela échoue, par exemple parce que la machine responsable de cette clef s'est retirée, il est possible de choisir pour nouvelle clef un condensé de la clef précédente (on parle de *rehashing*), déplaçant de ce fait les données sur un autre nœud.

**Choix des données à effacer.** Pastiche, pStore et ABS donnent la possibilité d'effacer des données sauvegardées. Seuls les propriétaires des données peuvent le faire car les demandes d'effacement doivent être signées avec la clef privée du propriétaire. De plus, à chaque bloc est associée une liste des propriétaires afin qu'un bloc ne soit effacé que lorsque tous ses propriétaires l'ont demandé.

PeerStore, en revanche, ne permet pas d'effacer des données sauvegardées. L'inclusion d'une telle fonctionnalité est compliquée par le fait que tous les blocs sauvegardés par un participant n'ont pas fait l'objet d'un accord avec un partenaire (section 3.2.2).

### *3.3.4. Résistance aux attaques en déni de service*

La mise en œuvre de mécanismes permettant de résister aux attaques par déni de service est un sujet à part entière qui demanderait plus de place pour être traité en détail. Nous n'aborderons donc ici que les principales solutions proposées pour chacune des attaques que nous avons identifiées en section 2.2.3.

**Égoïsme et inondation.** L'attaque par égoïsme est un problème rencontré par tous les systèmes de partage de ressources, en particulier les systèmes de partage de fichiers. De nombreuses solutions ont été proposées. Nous ne nous intéressons ici qu'à celles dédiées aux systèmes de sauvegarde.

Il faut d'abord remarquer qu'il est pratiquement impossible de garantir l'équité des contributions dans une DHT : par hypothèse, toutes les données sont réparties de manière homogène entre les nœuds, indépendamment des ressources utilisées par le nœud. Par conséquent, les systèmes pStore et Venti-DHash ne sont pas résistants à ce type d'attaque. Ils se placent dans le schéma de la « tragédie des biens communs » décrite par Hardin [14] : l'espace de stockage est un bien commun, son coût d'utilisation est partagé entre tous et l'intérêt de chaque participant est donc de profiter de cette ressource. La technique de *rehashing* d'ABS (cf. section 3.3.3) peut servir à équilibrer la charge sur la DHT mais elle ne prend pas en compte l'utilisation des ressources de chacun et est difficilement contrôlable.

PeerStore propose une solution assez simple : tous les échanges au sein d'un couple de partenaires doivent être *symétriques*, c'est-à-dire que chacun doit offrir la même quantité d'espace que ce qu'il consomme. Pour trouver des partenaires, chaque nouvel entrant diffuse une offre pour une certaine quantité d'espace de stockage et reçoit par les intéressés des propositions incluant une offre qui peut être différente. C'est au demandeur de décider si l'échange lui convient.

Pastiche ne traite pas ce problème mais Samsara [7], conçu comme une extension à Pastiche, apporte des solutions. Les auteurs y proposent l'établissement d'échanges symétriques par l'obtention d'un droit au stockage représenté par une sorte de capacité (*capability*, appelée ici *storage claim*) lorsqu'un contributeur s'engage à stocker des données. Les droits obtenus par un contributeur peuvent être cédés, lorsqu'il prend le rôle de propriétaire, à un autre participant. Enfin, il est possible à chaque participant de vérifier que ses droits au stockage sont respectés. La littérature des systèmes pair à pair de partage de fichiers décrit des solutions plus élaborées basées sur les notions de *confiance*, de *réputation*, ou de *micro-économie* [13,19].

Les propositions basées sur des rapports symétriques [7,20] ont l'avantage d'être résistantes aux attaques par inondation, où un nœud cherche à utiliser toutes les ressources du réseau. Au contraire, les DHT ne sont pas résistantes aux attaques par inondation de par la répartition homogène des données qu'elles font.

**Rétention de données.** Elnikety et al. [11] insistent sur la nécessité de pouvoir tolérer la rétention non intentionnelle, par exemple lorsqu'elle est due à des déconnexions, tout en étant en mesure de « punir » les abus.

Les solutions qu'ils proposent sont d'une part de vérifier que les partenaires stockent bel et bien les données qu'ils sont censés stocker, et d'autre part d'introduire des règles pour la tolérance des fautes temporaires. La vérification des données sauvegardées se fait par des *défis* réguliers, c'est-à-dire par l'envoi de demandes de lecture d'un bloc choisi au hasard au contributeur. La tolérance aux fautes temporaires (déconnexions) se fait par l'établissement d'une *période de grâce* pendant laquelle un participant peut être indisponible sans pour autant que les données qu'il a sauvegardées soient effacées. Cependant, ce mécanisme pourrait être utilisé pour profiter des ressources disponibles sans y contribuer. Par conséquent, les auteurs proposent en outre de définir une période d'essai, plus longue que la période de grâce, pendant laquelle les sauvegardes et défis sont autorisés mais pas les restauration.

Cette technique des défis est reprise par l'ensemble des autres systèmes de sauvegarde coopérative étudiés. En revanche, au lieu de demander la lecture d'un seul bloc par défi, les autres systèmes envoient typiquement une liste de blocs ; en réponse, ils reçoivent une simple signature de l'ensemble de ces blocs, ce qui limite l'utilisation de la bande passante [7,20].

En revanche, d'autres systèmes proposent des méthodes de tolérance aux déconnexions et de punition associée. Samsara [7] en particulier propose une punition progressive pour les participants ne répondant pas, plutôt que l'approche « tout ou rien » de la période de grâce. Dans Samsara, lorsqu'un nœud n'arrive pas à joindre un de ses partenaires, il détruit un bloc choisi au hasard. La probabilité d'effacement d'un bloc donné est choisie telle que, compte tenu du nombre de copies des données du nœud déconnecté, la probabilité qu'un bloc ait été effacé de chaque copie ne devient significative qu'après un nombre important de non-réponses auprès de ses partenaires. Là encore, PeerStore utilise la même approche.

## 4. CONCLUSIONS ET FUTURES DIRECTIONS

Les systèmes de sauvegarde coopérative que nous venons de présenter sont une source d'inspiration importante pour le système de sauvegarde coopérative pour dispositifs mobiles que nous envisageons. Toutefois, certaines solutions proposées par ceux-ci ne correspondent pas à nos objectifs et un certain nombre de questions restent ouvertes pour prendre en compte le fait que les dispositifs mobiles seront peu, voire jamais, connectés à Internet. Nous présentons

ici les pistes de recherche envisagées pour chacun de ces problèmes.

**Points communs et différences entre pair-à-pair et *ad hoc*.** Les réseaux *ad hoc* de dispositifs mobiles peuvent être vus comme une forme de réseau pair-à-pair puisque les dispositifs interagissent d'égal à égal, de manière décentralisée. Cependant, ce parallèle a ses limites dont certaines sont décrites dans la littérature [17]. La qualité et la bande passante des connexions sont évidemment moins bonnes dans un réseau *ad hoc* qu'elles ne peuvent l'être sur Internet. Les chemins à plusieurs sauts connectant deux dispositifs sont particulièrement instables. De plus, l'ensemble des dispositifs présents dans l'entourage d'un dispositif est en constante évolution. Beaucoup de réseaux pair-à-pair font au contraire l'hypothèse de connexions relativement stables d'un taux de renouvellement des participants assez faible. Cela est particulièrement le cas pour les réseaux virtuels dits « structurés » tels que ceux sur lesquels reposent les DHT[1]. Les réseaux virtuels font également généralement l'hypothèse d'un mécanisme de désignation fixe (IP ou autre [12]) qui n'est pas forcément disponible sur un réseau *ad hoc*.

La consommation énergétique est une préoccupation cruciale dans le cadre de systèmes mobiles, alors qu'elle est ignorée des travaux portant sur les systèmes pair-à-pair classiques. Il nous faudra tenir compte de cet aspect dans la conception de MoSAIC et de ses protocoles, en nous inspirant d'autres systèmes ayant des buts et des contraintes similaires [21,22].

**Découverte et allocation de ressources en mode *ad hoc*.** La découverte de ressources sur les réseaux pair à pair diffère également grandement ce qui peut se faire sur un réseau *ad hoc*. Pour participer à un réseau virtuel, il est généralement nécessaire et suffisant de connaître un participant, par exemple par son adresse IP, laquelle peut être obtenue auprès d'un serveur central de listes de participants, ou encore par diffusion de requêtes de découverte dans un réseau local [27]. Dans un réseau *ad hoc*, la découverte de dispositifs mobiles dans son environnement physique est fournie par le protocole réseau [17]. La découverte de ressources est cependant plus complexe du fait de l'instabilité du réseau et fait l'objet de nombreux travaux dont nous pourrons nous inspirer [15].

Bien entendu, à cause du caractère dynamique des réseaux *ad hoc*, les algorithmes d'allocation de ressources des réseaux pair-à-pair (DHT, ou au sein d'un groupe de partenaires) ne sont pas adaptés à un contexte *ad hoc*. Pour la sauvegarde et le recouvrement de données en mode *ad hoc*, nous pourrons tirer profit d'un grand nombre de travaux relatifs à la dissémination de données au sein d'une communauté de systèmes mobiles communiquants [3,22,23].

**Adaptation aux moyens de communication disponibles.** Dans la majorité des scénarios considérés, les dispositifs participant auront un accès intermittent à Internet, ou seront de temps en temps connectés à une station elle-même connectée. Ces instants devront donc être mis à profit pour améliorer la qualité du service de sauvegarde. Un scénario possible est que le système mobile envoie dans la « boîte aux lettres » de leur propriétaire (cf. section 2.1.2) les données qu'il a sauvegardées. Dans cette situation, les travaux présentés sur la sauvegarde et le stockage répartis pair à pair pourraient constituer une source d'inspiration importante dans la mise en œuvre du mécanisme de boîte aux lettres. Cependant, le système mobile lui-même ne pourrait pas contribuer à ce service de stockage réparti compte-tenu du fait qu'il ne sera que rarement connecté à Internet.

**Modèle de confiance.** Les modèles d'échange présentés dans la section précédente (mécanisme d'offre et de demande, échanges symétriques [20], établissement de liens symétriques [7], mécanisme de défis) font tous l'hypothèse que les participants sont le plus souvent connectés entre eux. Pour les scénarios que nous envisageons (forte mobilité, renouvellement constant du voisinage des dispositifs, établissement de connexions éphémères entre participants), ces mécanismes ne sont pas applicables. Des solutions pour dispositifs mobiles, inspirées des modèles utilisés dans les réseaux pair à pair, existent : dans [29], les auteurs proposent que des informations de réputation soient échangées entre participants, au gré des rencontres. Dans [16], nous proposons que chacun porte sa propre information de réputation, et que celle-ci puisse être vérifiée par les autres participants. La définition d'un tel mécanisme de réputation auto-portée est encore un problème ouvert.

L'utilisation du mécanisme de boîte aux lettres sur Internet engendre des défis supplémentaires quant à la conception d'un modèle de confiance. En particulier, il est important de ne oublier les dispositifs contributeurs qui ont rendu les données accessibles à leur propriétaire sur Internet. Lorsqu'un dispositif propriétaire recouvre ses données *via* Internet, il doit ainsi être capable d'être reconnaissant envers (i) les machines de bureau contribuent au service de boîte aux lettres sur Internet et (ii) le dispositif mobile contributeur qui a porté ses données jusqu'à Internet. Il s'agit là aussi d'un problème ouvert.

**Architecture.** En termes d'architecture logicielle, l'intégration du système de sauvegarde au système d'exploitation nous semble être une voie à explorer. La sauvegarde et l'archivage ont beaucoup en commun avec la gestion des versions. La littérature en matière de systèmes de fichiers gérant les versions [6,8,24,28] a montré combien cette approche offrait d'une part un meilleur contrôle sur les données (par exemple la possibilité de faire de l'« archivage exhaustif » des versions), et d'autre part des opportunités pour optimiser la sauvegarde de versions successives d'un fichier, en termes d'espace disque consommé et de performance.

[1] Il existe aussi des réseaux virtuels pair-à-pair dits « non structurés », c'est-à-dire ne reposant pas sur l'utilisation d'algorithmes de routage déterministes [4].

## BIBLIOGRAPHIE


[1] C. Batten, K. Barr, A. Saraf, S. Treptin. pStore : A secure peer-to-peer backup system. MIT-LCS-TM-632, MIT Laboratory for Computer Science, December 2001.

[2] K. Bennett, C. Grothoff, T. Horozov, J. T. Lindgren. An Encoding for Censorship-Resistant Sharing. 2003.

[3] M. Boulkenafed, V. Issarny. AdHocFS : Sharing Files in WLANs. *Proceedings of the 2nd International Symposium on Network Computing and Applications*, 2003.

[4] Y. Chawathe, S. Ratnasamy, L. Breslau, S. Shenker. Making Gnutella-like P2P Systems Scalable. *Proceedings of ACM SIGCOMM 2003*, pages 407–418, 2003.

[5] J. Cooley, C. Taylor, A. Peacock. ABS : The Apportioned Backup System. MIT Laboratory for Computer Science, 2004.

[6] B. Cornell, P. Dinda, F. Bustamante. Wayback : A User-level Versioning File System for Linux. *Proceedings of the*

*USENIX Annual Technical Conference, FREENIX Track*, pages 19–28, 2004.

[7] L. P. Cox, B. D. Noble. Samsara : Honor Among Thieves in Peer-to-Peer Storage. *Proceedings 19th ACM Symposium on Operating Systems Principles*, pages 120–132, 2003.

[8] L. P. Cox, B. D. Noble. Pastiche : Making backup cheap and easy. *Fifth USENIX Symposium on Operating Systems Design and Implementation*, pages 285–298, 2002.

[9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica. Wide-area cooperative storage with CFS. *Proceedings 18th ACM Symposium on Operating Systems Principles*, pages 202–215, 2001.

[10] Y. Deswarte, L. Blain, J-C. Fabre. Intrusion Tolerance in Distributed Computing Systems. *Proc. of the IEEE Symposium on Research in Security and Privacy*, pages 110–121, 1991.

[11] S. Elnikety, M. Lillibridge, M. Burrows. Peer-to-peer Cooperative Backup System. *Proceedings of USENIX FAST 2002*, 2002.

[12] R. Ferreira, C. Grothoff, P. Ruth. A Transport Layer Abstraction for Peer-to-Peer Networks. *Proceedings of IEEE/ACM Third International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems*, pages 398-405, 2003.

[13] C. Grothoff. An Excess-Based Economic Model for Resource Allocation in Peer-to-Peer Networks. *Wirtschaftsinformatik*, 3-2003, June 2003.

[14] G. Hardin. The Tragedy of the Commons. *Science*, (162), 1968.

[15] A. Helmy. Efficient Resource Discovery in Wireless AdHoc Networks : Contacts Do Help. *Kluwer Academic Publishers*, May 2004.

[16] M-O. Killijian, D. Powell, M. Banâtre, P. Couderc, Y. Roudier. Collaborative Backup for Dependable Mobile Applications. *Proceedings of 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (Middleware 2004)*, pages 146–149, 2004.

[17] G. Kortuem, J. Schneider, D. Preuitt, T. G. C. Thompson, S. Fickas, Z. Segall. When Peer-to-Peer comes Face-to-Face : Collaborative Peer-to-Peer Computing in Mobile Ad-hoc Networks. *Proceedings of the International Conference on Peer-to-Peer Computing (P2P2001)*, 2001.

[18] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao. OceanStore : An Architecture for Global-Scale Persistent Storage. *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, pages 190–201, 2000.

[19] K. Lai, M. Feldman, J. Chuang, I. Stoica. Incentives for Cooperation in Peer-to-Peer Networks. *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[20] M. Landers, H. Zhang, K-L. Tan. PeerStore : Better Performance by Relaxing in Peer-to-Peer Backup. *Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, pages 72–79, 2004.

[21] B. T. Loo, A. LaMarca, G. Borriello. Peer-To-Peer Backup for Personal Area Networks. IRS-TR-02-015, UC Berkeley ; Intel Seattle Research (USA), May 2003.

[22] E. B. Nightingale, J. Flinn. Energy-Efficiency and Storage Flexibility in the Blue File System. *Proceedings of the Sixth Symposium on Operating Systems Design and Implementation (OSDI'04)*, 2004.

[23] M. Papadopouli, H. Schulzrinne. Seven Degrees of Separation in Mobile Ad Hoc Networks. *IEEE Conference on Global Communications (GLOBECOM)*, 2000.

[24] Z. Peterson, R. Burns. Ext3cow : The Design, Implementation, and Analysis of Metadata for a Time-Shifting File System. HSSL-2003-03, Hopkins Storage Systems Lab, Department of Computer Science, Johns Hopkins University, USA, 2003.

[25] S. Quinlan, S. Dorward. Venti : A new approach to archival storage. *Proceedings of the First USENIX conference on File and Storage Technologies*, pages 89–101, 2002.

[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker. A Scalable Content-Addressable Network. *Proceedings of ACM SIGCOMM 2001*, 2001.

[27] A. Rowstron, P. Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, 2001.

[28] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, J. Ofir. Deciding when to forget in the Elephant file system. *Proceedings 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, pages 110–123, 1999.

[29] J. Schneider, G. Kortuem, J. Jager, S. Fickas, Z. Segall. Disseminating Trust Information in Wearable Communities. *2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K)*, 2000.

[30] E. Sit, J. Cates, R. Cox. A DHT-based Backup System. MIT Laboratory for Computer Science, August 2003.