THÈSE

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

Présentée pour obtenir le titre de
Docteur de l'Université de Toulouse,
délivré par l'Institut National Polytechnique de Toulouse

École doctorale Systèmes, spécialité Systèmes Informatiques

Par M. Ludovic Courtès

Sauvegarde coopérative de données pour dispositifs mobiles

Cooperative Data Backup for Mobile Devices

Soutenue le 23 novembre 2007 devant le jury composé de :

M. David Powell	Directeur de thèse
M. Marc-Olivier Killijian	Co-directeur de thèse
M. Hans-Peter Schwefel	Rapporteur
M. Pierre Sens	Rapporteur
M. Michel Banâtre	Membre
M. Claude Castelluccia	Membre
M. Ivan Frain	Membre
M. Yves Roudier	Membre

Avant-propos

Les travaux de thèse présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS). Je remercie les directeurs successifs du LAAS-CNRS, Mallik Ghallab et Raja Chatila pour m'avoir accueilli dans ce laboratoire. Je remercie également Jean Arlat, responsable du groupe de recherche Tolérance aux fautes et sûreté de fonctionnement informatique (TSF) au sein duquel j'ai effectué mes travaux, pour son accueil et sa bonne humeur.

J'exprime ma reconnaissance à mes encadrants : David Powell, directeur de recherche au LAAS-CNRS, et Marc-Olivier Killijian, chargé de recherche au LAAS-CNRS. Je les remercie d'avoir supporté mon entêtement pendant trois ans, d'avoir toujours été présents quand il le fallait, d'avoir été si bons relecteurs, et de m'avoir fait confiance. Leurs conseils et leurs critiques ont sans nul doute beaucoup contribué à ces travaux.

Je remercie les personnes qui m'ont fait l'honneur de participer à mon jury de thèse :

- M. Hans-Peter Schwefel, maître de conférence à l'Université d'Aalborg, Danemark ;
- M. Pierre Sens, professeur à l'Université Paris 6 et chercheur au LIP6 ;
- M. Michel Banâtre, directeur de recherche INRIA à l'IRISA, Rennes ;
- M. Claude Castelluccia, directeur de recherche INRIA à l'INRIA Rhône-Alpes, Grenoble ;
- M. Yves Roudier, maître de conférences à l'Institut Eurécom, Sophia-Antipolis ;
- M. Ivan Frain, ingénieur recherche et innovation dans la société Seanodes, à Toulouse.

Hans-Peter Schwefel et Pierre Sens ont accepté la charge de rapporteur et je leur en sais gré.

Le sujet de thèse dont il est question dans ce mémoire a pour origine le projet MoSAIC[1] (*Mobile System Availability, Integrity and Confidentiality*), partiellement financé par l'Action Concertée Incitative Sécurité & Informatique (ACI S&I) de 2004. Ce projet regroupait avec le LAAS-CNRS l'IRISA et Eurécom. Les discussions, voire les débats, qui ont eu lieu au sein du projet MoSAIC ont été enrichissants. J'exprime ma reconnaissance aux personnes que j'ai côtoyées dans ce projet, Michel Banâtre et Paul Couderc pour l'IRISA, Yves Roudier pour Eurécom, Matthieu Roy, David et Marc-Olivier pour le LAAS-CNRS, et en particulier à mes

[1] *http://www.laas.fr/mosaic/*

collègues doctorants du projet MoSAIC, Nouha Oualha (Eurécom) et Damien Martin-Guillerez (IRISA).

Ces travaux ont également été partiellement financés par le projet européen Hidenets (*Highly Dependable IP-Based Networks and Services*) et le réseau d'excellence européen ReSIST (*Resilience for Survivability in IST*). Merci aux responsables de ces deux projets.

Cette thèse a m'a parfois amené à explorer des sujets inattendus. Ainsi, suivant les conseils de David, je me suis lancé dans l'évaluation de la sûreté de fonctionnement à base de chaînes de Markov et de réseaux de Petri, où j'ai bénéficié du soutien de Mohamed Kâaniche, chargé de recherche au LAAS-CNRS. Je lui suis très reconnaissant pour son aide et sa disponibilité. Bien sûr, je n'oublie pas Ossama "Sem Sem" Hamouda, doctorant, qui m'a également beaucoup aidé à défricher ce terrain, toujours dans la bonne humeur. J'ai aussi apprécié le soutien de Thomas "Bob" Robert et de Benjamin Lussier, camarades doctorants, sur ces sujets épineux.

Je salue Minh Duc Nguyen, doctorant au LAAS-CNRS, dont j'ai suivi en 2005 le stage de mastère de recherche. Au printemps 2007, Frédérick Capovilla et Guillaume Vachon, deux étudiants venus du Québec, ont développé l'interface graphique MERLIn, évoquée au chapitre 6, et ce dans une ambiance de travail agréable dont je les remercie. Mon collègue Christophe Zanon, ingénieur d'étude au LAAS-CNRS, a continué le travail sur MERLIn, rajoutant des fonctionnalités toujours plus folles. Il s'est montré réactif et efficace alors que le jour J de la démonstration approchait, et ce tout en restant décontracté. Merci beaucoup pour cette aide précieuse.

Bien sûr, cette thèse eut été tout autre sans la présence de mes collègues doctorants au LAAS-CNRS. Je salue donc mes camarades du Bureau Dix, un îlot de quiétude et de résistance dans cette grande maison : Ben Lussier, l'auto-proclamé « Empereur du Bureau Dix » et en tout cas doyen du Bureau, Étienne Baudin, mon voisin d'en-face et la bonne humeur des lieux, Youssef Laarouchi dit "Youyou", toujours prompt à taquiner ses camarades, et Caroline Lu qui a su apporter un peu de finesse dans cette pièce. Ces joyeux drilles m'ont beaucoup aidé durant ces trois ans, y compris dans les moments difficiles. Je salue également Manel Sghairi qui a eu l'opportunité de séjourner quelques temps au Bureaux Dix.

On trouve aussi de sympathiques doctorants au-delà du Bureau Dix, notamment Thomas "Bob" Robert, un excellent conseiller scientifique et non moins excellent cuisinier, son turbulent mais chaleureux voisin Thomas Pareaud, mes brillants « conscrits » Ana Rugina et Éric Alata, Carlos Aguilar Melchior mon guide favori en cryptographie, Nicolas Salatgé qui fait maintenant partie de la population vraiment active, et les « jeunes » Géraldine Vache, Éric Lacombe, Amine Baina et Mohamed Gad El Rab. Avec les autres membres du groupe TSF, ils ont contribué à faire de ces trois ans un bon moment.

Enfin, je remercie les personnes en dehors du LAAS qui m'ont soutenu, famille et amis, et en particulier Ségo pour son soutien et sa compréhension, sans oublier celle « qui lui fait une bosse sous le nombril ».

Table des matières

Liste des figures

Introduction

Les systèmes mobiles tels que les ordinateurs portables, assistants personnels (PDA) et téléphones portables sont de plus en plus utilisés et dans des contextes où ils risquent d'être abîmés, perdus ou volés. Ces outils sont utilisés pour produire des données. Des appareils tels que des appareils photo numériques ou caméras peuvent produire des quantités de données importantes. En même temps qu'ils deviennent de plus en plus petits, polyvalents, et puissants, ces dispositifs mobiles sont de plus en plus utilisés dans des domaines variés de notre vie courante, nous rendant de plus en plus dépendants.

Malgré tout, relativement peu de mécanismes sont disponibles pour améliorer la *disponibilité* des données stockées sur ces appareils. En outre, les mécanismes disponibles tels que les mécanismes de « synchronisation » souffrent de limitations. Il requièrent notamment une intervention manuelle de l'utilisateur, et leur utilisation est souvent contraignante (par exemple, l'utilisateur doit être dans l'environnement physique de son ordinateur de bureau). Cette situation laisse les utilisateurs de dispositifs mobiles avec, au mieux, des possibilités de sauvegarde de données intermittentes.

Nous pensons que cette situation appelle de nouveaux mécanismes de sauvegarde tirant davantage parti des possibilités offertes par les dispositifs mobiles et leurs contextes d'utilisations. Ces dispositifs deviennent *omniprésents* et *communicants.* Avec l'arrivée de technologies de communication réseau *ad hoc*, les réseaux spontanés deviennent une réalité, permettant à des périphériques proches les uns des autres de communiquer entre eux sans aucune intervention humaine. Ces observations nous ont poussé à explorer les possibilités permettant de tirer parti des interactions spontanées dans l'objectif de créer un *service de sauvegarde coopérative.*

Ce mémoire de thèse est divisé en deux parties respectivement en français en anglais. Les deux parties suivent la même structure, la première étant un résumé de la deuxième. Le chapitre 1 fournit un aperçu des sujets abordés dans cette thèse. Les chapitres suivants abordent des aspects spécifiques de la conception, mise en œuvre, et évaluation du service de sauvegarde coopérative. Le chapitre 2 décrit nos motivations et nos objectifs de sûreté de fonctionnement. Le chapitre 3 s'intéresse à l'évaluation analytique de la sûreté de fonctionnement du service. Le chapitre suivant se concentre sur la conception de mécanismes de stockages répartis adaptés à nos objectifs et fournit une évaluation expérimentale de certains mécanismes. Le chapitre 5 traite des problèmes liés à la coopération entre participants sans relation de confiance préalable. Enfin, le chapitre 6 décrit comment nous avons assemblé les différents résultats et propositions des chapitres précédents dans un prototype du service de sauvegarde coopérative.

Chapitre 1. Démarche et contributions

Ce chapitre décrit le contexte de nos travaux et donne un aperçu des thèmes abordés dans cette thèse, en faisant référence aux chapitres qui les abordent. À la fin de chaque chapitre se trouve un résumé des travaux connexes.

1.1. Contexte

Cette section présente les deux principaux domaines abordés dans cette thèse : la *sûreté de fonctionnement* et *l'informatique ubiquiste.*

1.1.1. Sûreté de fonctionnement et tolérance aux fautes

Les travaux présentés dans cette thèse ont pour but d'améliorer la *sûreté de fonctionnement* des dispositifs mobiles, en proposant des mécanismes de *tolérance aux fautes* permettant de limiter le risque des perte des données.

La sûreté de fonctionnement est un domaine de recherche actif, avec ses propres concepts et terminologie [Avizienis *et al.* 2004]. Ainsi, la tolérance aux fautes est définie comme un *moyen* permettant d'éviter la défaillance d'un service en présence de fautes. On définit la *défaillance d'un service* comme l'évènement qui survient lorsque le service rendu s'éloigne du service correct. Enfin, une *faute* est la cause supposée ou adjugée d'une erreur, une *erreur* étant elle-même la partie de l'état total du système pouvant conduire à sa défaillance. Il y a donc une chaîne causale, où une faute peut engendrer une erreur, qui à son tour peut conduire à la défaillance du service.

1.1.2. Informatique ubiquiste et réseaux mobiles *ad hoc*

Au cours de la dernière décennie, les dispositifs informatiques mobiles se sont multipliés, devenant de plus en plus petits et de plus en plus puissants (PDA, téléphones mobiles, appareils photo et vidéo). Dans le même temps, des moyens de communication sans fil de courte à moyenne portée se sont répandus. Il est devenu possible de connecter très facilement différents dispositifs entre eux ou avec un ordinateur de bureau, en utilisant des protocoles tels que Bluetooth ou Zigbee. Les réseaux locaux sans fil (WLAN) sont également devenus courants.

Enfin, des mécanismes permettant la création *spontanée* de réseaux sans fil ont fait leur apparition. En particulier, les réseaux mobiles *ad hoc* sans fils (MANET) permettent à un ensemble de dispositifs de maintenir de manière *coopérative* un réseau d'interconnexion.

Des techniques similaires ont été proposées pour créer des réseaux sans fils urbains (MAN) [Bicket *et al.* 2005], pour permettre l'accès à Internet dans des zones reculées [Jain & Agrawal 2003, One Laptop Per Child Project 2007]. Étant intrinsèquement décentralisées, ces techniques donnent plus de pouvoir aux utilisateurs et promeuvent la coopération. Les technologies offrant ces possibilités incluent le standard IEEE 802.11 [IEEE-SA Standards Board 1999] (*Wi-Fi*) et les algorithmes de routage de paquets tels que AODV [Perkins *et al.* 2003] et OLSR [Clausen & Jacquet 2003].

Les capacités de stockage des dispositifs informatiques mobiles ont aussi été considérablement accrues. Souvent, les dispositifs mobiles jouent un rôle de « cache », leurs données étant ensuite copiées sur d'autres dispositifs tels qu'un ordinateur de bureau. Dans cette thèse, nous nous focalisons sur des méthodes permettant de tirer parti des ces possibilités afin de réduire le risque de perdre les données créées avec des dispositifs mobiles.

1.2. Objectif de la thèse

Compte-tenu de ces observations, nous nous proposons de démontrer l'affirmation suivante :

> Il est possible d'améliorer la sûreté de fonctionnement de dispositifs mobiles en tirant parti de coopération spontanée entre de tels dispositifs.

Concrètement, nous proposons un *service de sauvegarde coopérative* qui copie de manière opportuniste les données critiques de dispositifs mobiles sur les dispositifs mobiles voisins en utilisant des moyens de communication sans fils spontanés.

1.3. Aperçu de la thèse

tolérance aux fautes

Dans cette thèse, nous nous intéressons à des moyens de *tolérance aux fautes* pour dispositifs mobiles, et plus précisément à la sauvegarde de données stockées sur ces dispositifs. Les mécanismes de synchronisation de données actuellement disponibles pour ces dispositifs mobiles sont souvent *ad hoc*, contraignants, et requièrent une intervention manuelle de l'utilisateur. En particulier, il est souvent nécessaire d'avoir accès à une infrastructure réseau, voire d'être à proximité physique du dispositif sur lequel on souhaite sauvegarder ses données. Cette situation laisse peu d'occasions pour effectuer des sauvegardes. Les données produites sur les dispositifs mobiles courent donc le risque d'être perdues avant qu'une occasion de les sauvegarder se soit présentée. Le chapitre 2 décrit notre approche plus en détail et le chapitre 3 propose une évaluation analytique de la sûreté de fonctionnement du service proposé.

stockage réparti

L'approche de sauvegarde *coopérative* que nous proposons nous a amenés à aborder des thèmes relatifs au *stockage réparti*. En effet, les mécanismes de stockage dont nécessite

un tel service diffèrent sensiblement de ceux trouvés dans les systèmes centralisés. Ils se rapprochent davantage des méthodes de stockage utilisées par les systèmes *pair-à-pair* sur Internet, ou encore de ceux que l'on trouve dans les systèmes de fichiers répartis pour réseaux mobiles *ad hoc*. Dans ce cadre, nous nous intéressons également à la définition de *stratégies de réplication*, et considérons l'utilisation de *codes d'effacement*. Le chapitre 4 aborde ces questions et contribue au domaine en proposant des mécanismes adaptés à notre contexte.

stratégies de réplication

codes d'effacement

Notre objectif est de permettre à des dispositifs mobiles appartenant à des personnes différentes, n'ayant pas de relation de confiance préalable, de coopérer pour fournir le service de sauvegarde. Par conséquent, un certains nombres de problèmes de *sécurité* se posent : un participant pourrait accéder sans y être autorisé aux données d'un autre, il pourrait renier sa promesse de stocker des données pour un autre, ou bien encore il pourrait mener des attaques en déni de service contre le service de sauvegarde. Des mécanismes permettant d'*imputer* les actes de chacun à leur auteur sont nécessaires. Le chapitre 5 propose des mécanismes de base permettant aux utilisateurs de mettre en œuvre une large gamme de *politiques de coopération.*

sécurité

politiques de coopération

Enfin, nous proposons une mise en œuvre du service de sauvegarde coopérative faisant appel aux résultats précédents ainsi qu'à des techniques de gestion de version. Notre prototype est constitué d'un programme *démon de sauvegarde* et d'un ensemble d'*outils clients* permettant d'interagir avec lui. Le démon est responsable de la sauvegarde des données locales et de leur réplication de manière opportuniste sur d'autres dispositifs. Le chapitre 6 décrit ce prototype et propose une évaluation préliminaire.

Chapitre 2. Service de sauvegarde coopérative pour dispositifs mobiles

Les dispositifs informatiques mobiles sont de plus en plus utilisés mais les mécanismes de sauvegarde de données existant sont limités. Par conséquent, les données stockées sur ces dispositifs risquent d'être perdues. Ce chapitre présente les motivations et objectifs de sûreté de fonctionnement qui nous ont poussé à concevoir un service de sauvegarde coopérative. Il décrit ensuite le service envisagé.

2.1. Motivations

Nous décrivons d'abord le problème que nous cherchons à résoudre puis notre approche.

2.1.1. Description du problème

Les dispositifs tels que les ordinateurs portables, assistants personnels (PDA) ou téléphones mobiles sont de plus en plus utilisés, mais sont sujets à la perte, au vol, ou aux dommages physiques. Cependant, les mécanismes de tolérance aux fautes, et en particulier de sauvegarde des données disponibles sur ces dispositifs ont des limitations. Les mécanismes de « synchronisation » des données, tels que SyncML [Open Mobile Alliance 2001] requièrent généralement que la machine avec laquelle on synchronise les données soit physiquement accessible ou au moins accessible par le biais d'une infrastructure réseau. Une autre solution consiste à utiliser des serveurs tiers, tels que ceux fournis par `box.net`. Là encore, l'accès à une infrastructure réseau est un requis.

Malheureusement, dans beaucoup de scénarios, il est difficile de compter sur l'accès à une infrastructure réseau, celui-ci étant intermittent, rare, ou trop coûteux pour une utilisation à des fins de sauvegarde (comme c'est parfois le cas pour les liaisons GSM/GPRS ou UMTS). Dans certaines situations, comme dans des zones très reculées [Jain & Agrawal 2003] ou dans les milieux auxquels se consacre le projet *One Laptop Per Child* [One Laptop Per Child Project 2007], il peut être tout simplement très difficile d'accéder à une infrastructure réseau.

Par conséquent, les occasions de sauvegarde des données stockées sur des dispositifs mobiles peuvent être rares, ce qui engendre un risque de perte.

2.1.2. Une approche coopérative de la sauvegarde

Le service que nous envisageons et que nous appelons MoSAIC[1] a pour objectif d'améliorer la disponibilité des données stockées sur les dispositifs mobiles, en leur fournissant des moyens pour tolérer aussi bien les fautes permanentes (perte, vol, endommagement) que les fautes transitoires (effacement accidentel des données ou corruption). Pour tolérer les fautes permanentes, notre service doit être capable de stocker les données d'un dispositif sur d'autres dispositifs.

Les moyens de communication sans fils dorénavant ubiquistes fournissent le support à un tel service. Le service de sauvegarde envisagé est donc coopératif, décentralisé, et n'importe quel périphérique peut y participer. On attend d'un périphérique qui utilise le service qu'il y contribue en retour. Par la suite, nous utilisons le terme *contributeur* pour désigner un périphérique jouant le rôle de fournisseur d'espace de stockage ; on utilise le terme de *propriétaire des données* pour désigner un périphérique dans son rôle d'utilisateur ou de « client » du service. Évidemment, dans un souci d'équité, chaque dispositif doit jouer les deux rôles.

contributeur

propriétaire des données

Cette approche s'inspire des réseaux pair-à-pair coopératifs tels que les réseaux de partage de fichiers largement répandus sur Internet [Kügler 2003, Clarke *et al.* 2001, Dabek *et al.* 2001, Kubiatowicz *et al.* 2000]. Une approche similaire a déjà été appliquée à la sauvegarde de données sur Internet [Cox *et al.* 2002, Cox & Noble 2003, Lillibridge *et al.* 2003, Landers *et al.* 2004, Goldberg & Yianilos 1998]. Du point de vue de la tolérance aux fautes, ces approches bénéficient de l'hétérogénéité logicielle et matérielle des ordinateurs participant.

Un mécanisme de sauvegarde coopérative pour dispositifs mobiles permet la sauvegarde de données même en l'absence d'accès à une infrastructure réseau. Même lorsque qu'une infrastructure est disponible (comme GSM/GPRS, UMTS, WiMAX, etc.), il fournit une alternative meilleur marché, aussi bien financièrement que, dans une certaine mesure, en termes de consommation énergétique. En outre, les moyens de communication à faible portée offrent souvent des débits plus élevés que les moyens de communication à longue portée. Enfin, comme les périphériques participants sont hétérogènes et appartiennent à des domaines d'administration différents, aucun périphérique n'est un *point singulier de défaillance* ni un *point singulier de confiance*. En d'autres termes, la défaillance ou compromission d'un dispositif participant ne saurait mettre en péril le service de sauvegarde.

La consommation énergétique des interfaces réseaux sans fils telles que Wi-Fi reste une préoccupation. Wi-Fi en mode *ad hoc* consomme autant d'énergie, que l'on s'en serve ou non [Feeney & Nilsson 2001]. Toutefois, les technologies sans fils à venir portent la promesse de consommation énergétique réduite. ZigBee, par exemple, consomme peu d'énergie, moyennant des débits plus faibles [Zheng & Lee 2004]. Les ordinateurs du projet OLPC,

---

[1] *Mobile System Availability, Integrity and Confidentiality*, *http://www.laas.fr/mosaic/*. MoSAIC est un projet partiellement financé par l'Action Concertée Incitative Sécurité & Informatique (ACI S&I). Nos partenaires sont l'IRISA (Rennes) et Eurécom (Sophia-Antipolis).

qui utilisent un protocole de réseau *ad hoc* proche de 802.11s, atteignent des consommations relativement faibles en ayant intégré le protocole de routage *ad hoc* au niveau matériel, ce qui leur permet de continuer à router des paquets même lorsque qu'ils sont en veille [One Laptop Per Child Project 2007]. Par ailleurs, les fournisseurs d'accès téléphonique UMTS offrent de plus en plus de services de téléchargement de flux audio ou vidéo, des applications pourtant très gourmandes en bande passante, ce qui illustre une réduction de la consommation énergétique de ce type d'interfaces. Enfin, des propositions ont été faites pour utiliser de la prédiction de trafic réseau afin de permettre l'arrêt des interfaces réseaux lorsque cela est jugé opportun ; des expériences ont montré que cette approche permettait de réduire significativement la consommation énergétique [Zhang *et al.* 2005].

Évidemment, l'intérêt du service de sauvegarde coopérative dépendra grandement de la fréquence de rencontre de périphériques participants. Néanmoins, dès lors qu'il y a rencontre, il y a opportunité de sauvegarde, et donc de diminution du risque de perte de données. Entre autres choses, notre analyse de la disponibilité des données présentée au chapitre 3 cherche à identifier les scénarios où notre approche est profitable, comparé à une sauvegarde uniquement lorsqu'une infrastructure réseau est disponible.

2.2. Objectifs de sûreté de fonctionnement

Notre principal objectif est d'améliorer la sûreté de fonctionnement des dispositifs mobiles. Toutefois, la sûreté de fonctionnement d'un service coopératif ouvert tel que nous l'envisageons est elle aussi sujette à un certain nombre de menaces. Il s'agit donc de rendre le service lui-même également sûr de fonctionnement. Dans cette section, nous détaillons les menaces issues de la coopération entre dispositifs ne se faisant pas confiance et les mécanismes envisagés pour y remédier.

2.2.1. Menaces contre la confidentialité et le respect de la vie privée

S'agissant des stocker des données critiques sur des dispositifs auxquels on ne fait pas confiance, des menaces évidentes pèsent sur le service : un utilisateur malveillant pourrait essayer d'accéder aux données qu'il stocke pour d'autres participants. Les mécanismes de stockage utilisés doivent donc permettre le chiffrement des données de bout en bout, comme nous le verrons dans le chapitre 4.

Le respect de la vie privée des participants peut être également mis en danger. Il est aisé pour un observateur de savoir *si* un utilisateur donné participe, et de savoir quelles quantités de données il échange et avec qui. Des protocoles de routage anonymes pour réseaux mobiles *ad hoc* ont été proposés [Rahman *et al.* 2006, Aad *et al.* 2006]. Toutefois, nous ne traitons pas en détail de cette problématique dans notre thèse. Nous espérons néanmoins garantir un minimum le respect de la vie privée en permettant aux utilisateurs

d'utiliser des identifiants autogérés (des *pseudonymes*) plutôt que de forcer l'utilisation d'identifiants délivrés par une autorité centrale.

2.2.2. Menaces contre l'intégrité et l'authenticité

Un contributeur malveillant pourrait bien sûr chercher à modifier les données qu'il stocke pour d'autres participants, voire injecter de nouvelles données en faisant croire qu'elles proviennent d'un autre participant. Ces menaces doivent être écartées par la couche de stockage.

Des menaces similaires pèsent sur les communications entre participants : un participant pourrait chercher à se faire passer pour un autre, conduisant ainsi à l'imputation de ses actes au participant dont il a usurpé l'identité. Enfin, un participant malveillant pourrait vouloir modifier le contenu des message échangés entre participants. Les mécanismes de communication doivent donc traiter ces menaces.

2.2.3. Menaces contre la disponibilité

Les menaces contre la disponibilité peuvent être classées dans deux catégories : l'indisponibilité des données résultant de la perte accidentelle de données (y compris de la perte des exemplaires stockés par des contributeurs), et l'indisponibilité des données ou du service résultant d'*attaques en déni de service* (DoS) perpétrées par des utilisateurs malveillants.

attaques en déni de service

La première catégorie est évidemment celle qui nous intéresse en premier lieu, tandis que la deuxième découle du modèle de coopération ouverte que nous envisageons. Une attaque en déni de service contre un participant est la *rétention de données* où un contributeur malveillant refuse volontairement de rendre les données à leur propriétaire (cela peut être simplement parce qu'il ne les a jamais stockées). Des attaques contre le service dans son ensemble sont l'*inondation* (l'exploitation volontaire jusqu'à l'épuisement des ressources offertes par le service) et l'*égoïsme* (l'utilisation du service sans contribution en retour). Nous reviendrons sur ces aspects au chapitre 5.

2.2.4. Discussion

Le chapitre 3 décrira et évaluera des stratégies de réplication visant à améliorer la disponibilité des données. Le chapitre 4 montrera comment traiter les problèmes de confidentialité et d'intégrité des données au niveau de la couche de stockage. Enfin, le chapitre 5 proposera des solutions pour contrer les menaces contre la coopération.

2.3. Processus de sauvegarde et de recouvrement

Cette section détaille les processus et de recouvrement dans le cadre du service coopératif de sauvegarde tel que nous l'envisageons.

2.3.1. Processus de sauvegarde

Dans un premier temps, les utilisateurs sont supposés « synchroniser » leurs données avec leur ordinateur fixe, lorsqu'ils y ont accès. Par la suite, alors qu'ils se déplacent avec leurs dispositifs mobiles, ceux-ci devront spontanément se connecter aux dispositifs voisins participant au service dans le but d'échanger des données à sauvegarder, et ce de manière transparente à l'utilisateur. Des *protocoles de découverte de services* seront utilisés à cette fin. De tels protocoles sont aujourd'hui répandus dans le domaine des ordinateurs de bureau [Cheshire & Krochmal 2006a, Guttman *et al.* 1999, Goland *et al.* 1999]. Des protocoles de découverte de services adapatés aux réseaux *ad hoc* ont été proposés [Sailhan & Issarny 2005, The UbiSec Project 2005, Helmy 2004, Kim *et al.* 2005, Poettering 2007] mais aucun n'a été largement déployé à l'heure actuelle. Nous ne nous focalisons pas sur la conception de tels protocoles et considérons plutôt l'utilisation d'un protocole existant.

protocoles de découverte de services

Une fois un contributeur découvert, un propriétaire lui envoie une requête de stockage de données. Un propriétaire pourra choisir les contributeurs en fonction de la confiance qu'il leur accorde. Par exemple, il pourra choisir en répondant à la question « est-ce que ce contributeur s'est déjà comporté correctement par le passé? ». En pratique, les requêtes comprendront des blocs de données de petite taille qui seront plus adaptées au fait que les rencontres de contributeurs sont imprévisibles et potentiellement de courte durée. Les données seront généralement chiffrées. Enfin, un contributeur pourra lui-même choisir d'accepter ou non une requête en fonction des critères de son choix, tel que sa capacité de stockage disponible ou son niveau d'énergie.

Dans la plupart des scénarios, il serait irréaliste de compter sur une rencontre entre le propriétaire et ses contributeurs pour la restauration de ses données. Par conséquent, nous supposons que les contributeurs transfèrent les données que leurs ont fourni les propriétaires vers un support de stockage accessible sur Internet. Les propriétaires contactent ensuite ce support de stockage pour restaurer leurs données. Un tel support de stockage pourrait être mis en œuvre de différentes façons : un simple serveur FTP commun à tous les participants, un réseau pair-à-pair, la boîte aux lettres électronique du propriétaire, etc. Nous ne nous focalisons pas ici sur cette mise en œuvre

Les contributeurs pourraient aussi *faire suivre* les données qu'ils ont collectées vers d'autres contributeurs. Cependant, cette approche n'apporte aucun avantage du point de vue de la disponibilité des données en l'absence d'informations supplémentaires sur les contributeurs (par exemple, il pourrait être profitable de transmettre les données à un contributeur qui a souvent accès à Internet, mais cette information n'est pas forcément connue des autres). De même, chaque contributeur pourrait aussi *copier* les données qu'il

a reçues vers d'autres contributeurs. Cette approche est attrayante, car elle augmenterait bien entendu la disponibilité des données. Cependant, elle pose plusieurs problèmes. D'abord, il est improbable qu'un propriétaire fasse confiance à ses contributeurs pour effectuer réellement cette copie, l'intérêt des contributeurs étant plutôt d'économiser leur énergie. Ensuite, sans davantage de coordination, cette approche pourrait rapidement conduire à une inondation du service. Nous aborderons de nouveau ces problèmes dans le chapitre 3.

2.3.2. Processus de restauration

Dans un scénario purement *ad hoc*, les participants pourraient restaurer leurs données à partir des contributeurs présents dans leur voisinage. Dans le cas général, ils récupéreront leur données à partir d'un serveur accessible par Internet, comme nous l'avons mentionné. La plupart du temps, les propriétaires demanderont la dernière sauvegarde disponible. Parfois, il pourra aussi être utile de demander, par exemple, « la sauvegarde datant du 4 avril ».

Étant donné que chaque dispositif accède à Internet à son propre rythme, les sauvegardes peuvent mettre un certain temps à se propager jusqu'au serveur sur Internet. Ainsi, il n'est pas impossible que la sauvegarde disponible à un instant donné sur Internet ne soit pas la dernière effectuée. Parfois, il pourra donc être possible pour un propriétaire de restaurer une version plus récente à condition d'attendre plus longtemps, ce qui illustre un compromis entre la récence et la disponibilité des données.

2.4. Travaux connexes

Dans cette section, nous décrivons les travaux connexes au nôtre, dans le domaine de la tolérance aux fautes, de la réplication, de la sauvegarde coopérative sur Internet, du stockage réparti pour dispositifs mobiles, et des réseaux tolérant les retards (*delay-tolerant networks* ou DTN en anglais).

2.4.1. Points de reprise

Dans cette thèse, nous nous intéressons à la tolérance aux fautes de dispositifs mobiles individuels vus comme des systèmes centralisés fournissant un service (prise de photos, production de documents, etc.) ne dépendant pas nécessairement d'autres dispositifs, par opposition aux systèmes répartis dont l'activité implique plusieurs nœuds.

La tolérance aux fautes de systèmes centralisés est généralement réalisée en stockant la liste des opérations changeant son état, ou en stockant des copies de son état, appelées *points de reprise*, à intervalles réguliers. Lors du recouvrement, on rejoue les opérations ou bien on réinstalle le dernier point de reprise. Pour que le point de reprise soit sûr de fonctionnement, il doit avoir un certain nombre de propriétés, en particulier l'*atomicité*

et la *cohérence*, un sous-ensemble des propriétés trouvées dans les systèmes de gestion de bases de données (SGBD) [Gray 1981]. On parle de « sémantique transactionnelle ».

Les *systèmes d'exploitation persistants* [Liedtke 1993, Shapiro & Adams 2002] et les approches intégrées aux langages de programmation offrent des solutions génériques [Prevayler.Org 2006, Red Hat, Inc. 2007]. Toutefois, les plus souvent, les applications utilisent des solutions *ad hoc*, ayant recours au *système de fichiers* pour sauvegarder leurs données. La création d'exemplaires supplémentaires des points de reprise est généralement reléguée à des outils tiers qui parcourent le système de fichiers [Melski 1999, Rubel 2005, Quinlan & Dorward 2002].

Les systèmes de gestion de version fournissent le plus souvent atomicité et capacité de retour à un état antérieur [Tichy 1985, Hamano 2006], de même que les systèmes de fichiers intégrant des mécanismes de gestion de version [Santry *et al.* 1999, Hitz *et al.* 1994, Peterson & Burns 2003, Cornell *et al.* 2004]. Certains d'entre eux fournissent aux utilisateurs, en plus de l'interface POSIX, une interface leur permettant de demander explicitement le stockage d'une nouvelle version.

2.4.2. Sauvegarde coopérative pair-à-pair

La plupart des travaux sur la sauvegarde coopérative concernent une mise en œuvre sur Internet. Les outils de sauvegarde coopérative sur Internet s'inspirent des réseaux de partage de fichiers, en cherchant à mettre à contribution les ressources de stockage disponibles pour les partager et s'en servir comme support du service de sauvegarde. Les solutions proposées diffèrent principalement dans leur niveau de centralisation, d'une part concernant la *découverte* et d'autre part concernant les échanges de données entre participants [Lillibridge *et al.* 2003, Cox *et al.* 2002, Cox & Noble 2003, Landers *et al.* 2004, Batten *et al.* 2001, Sit *et al.* 2003].

Ces systèmes ont également différentes manières d'aborder les problèmes de sécurité que nous avons mentionnés, notamment les attaques en déni de service. Certains considèrent que les participants se comportent correctement [Cox *et al.* 2002, Batten *et al.* 2001], alors que d'autres fournissent des protocoles permettant de traiter les comportements malveillants [Cox & Noble 2003, Lillibridge *et al.* 2003, Aiyer *et al.* 2005]. Nous aborderons ces aspects plus en détail au chapitre 4.

À notre connaissance, peu de tentatives ont eu lieu pour transposer le modèle de sauvegarde coopérative à l'environnement mobile. On citera toutefois *FlashBack*, un outil de sauvegarde coopérative entre dispositifs au sein d'un *réseau personnel sans fils* (PAN) [Loo *et al.* 2003], et *OmniStore* qui permet la réplication automatique entre dispositifs portables au sein d'un réseau personnel, avec migration des données vers un ordinateur de bureau [Karypidis & Lalis 2006]. Ces approches diffèrent de la nôtre car dans leur contexte, tous les dispositifs participant appartiennent à la même personne, et donc se font mutuellement confiance. En particulier, les attaques contre la confidentialité et l'intégrité, de même que les attaques en déni de service sont hautement improbables dans ce cadre. Une autre

réseau personnel sans fils

différence est que les dispositifs interagissant au sein d'un PAN sont souvent à portée l'un de l'autre.

2.4.3. Stockage mobile réparti

Beaucoup de travaux récents ont porté sur la conception de systèmes de stockage répartis pour dispositifs mobiles. Leurs objectifs sont généralement le partage d'informations (où les informations sont donc disponibles en *lecture seule* [Sözer *et al.* 2004, Zöls *et al.* 2005, Papadopouli & Schulzrinne 2000, Tolia *et al.* 2004, Flinn *et al.* 2003]), ou la mise en place d'un support de stockage réparti réinscriptible pouvant être accédé par différents dispositifs. Bien qu'ils ne s'agisse pas de sauvegarde, les solutions proposées sont parfois pertinentes dans notre cadre.

Les systèmes de stockage inscriptibles ont été proposés d'abord comme une extension des systèmes de fichiers utilisés sur les ordinateurs fixes [Demers *et al.* 1994, Lee *et al.* 1999], puis de manière davantage décentralisée [Karypidis & Lalis 2006, Preguica *et al.* 2005, Nightingale & Flinn 2004, Boulkenafed & Issarny 2003, Barreto & Ferreira 2004]. Ces systèmes doivent répondre à des problème de *cohérence des caches* maintenus par chacun des participants. Pour ce faire, ils font souvent appel à des techniques utilisées dans le domaine de la gestion de version, notamment pour la fusion d'exemplaires divergents d'un même fichier. Cette problématique ne se retrouve pas dans le cadre d'un service de sauvegarde où les données ne sont pas modifiées de manière destructive. En outre, seuls certains prennent en comptent la coopération entre dispositifs ne se faisant pas mutuellement confiance.

2.4.4. Réseaux tolérant les retards

Les réseaux tolérant les retards (DTN) traitent le problème de variabilité de la connectivité : manque d'accès à une infrastructure, connectivité intermittente, partitionnement du réseau dû à la mobilité. Notre approche de la sauvegarde coopérative peut être vue sous cette approche réseau : les données transmises de propriétaires à contributeurs puis vers un support de stockage sur Internet peuvent être vues comme des paquets envoyés des propriétaires vers le support de stockage sur Internet, les contributeurs servant de simples *relais*. Le processus de sauvegarde peut être vu comme un canal de communication à forte latence où transitent les sauvegardes.

Ces réseaux ont été largement étudiés au cours des dernières années [Zhang 2006]. Leurs applications incluent l'exploration interplanétaire, les situations de crise à la suite de désastres, ou encore l'observation de la vie sauvage [Fall 2003, Harras *et al.* 2007, Juang *et al.* 2002]. La contribution de ces travaux porte principalement sur deux points : la définition de protocoles d'échanges de données [Scott & Burleigh 2007, Cerf *et al.* 2007] et la conception et l'évaluation d'algorithmes de routage [Zhang 2006, Spyropoulos *et al.* 2007]. Ce dernier aspect est à rapprocher des stratégies de réplication que nous abordons notamment au chapitre 3.

Toutefois, les DTN diffèrent de notre approche de plusieurs façons. D'abord, le plus souvent, ils considèrent que la destination d'un paquet est une nœud physique spécifique, alors que dans notre cas la destination est un support de stockage sur Internet, donc accessible depuis une multitude de points d'accès. Ensuite, les DTN ne rendent pas compte de la dimension apportée par la gestion de versions : dans notre cas, une version même ancienne des données a une valeur intrinsèque, d'un point de vue tolérance aux fautes, alors que l'approche purement réseau des DTN ignore cette sémantique. Enfin, d'un point de vue sécurité, peu de travaux ont été menés concernant la coopération entre participants ne se faisant pas confiance dans le cadre des DTN [Fall 2003, Farrell & Cahill 2006]. Nos travaux sont une contribution dans ce domaine, comme nous le verrons dans le chapitre 5.

2.5. Résumé

Les contributions de ce chapitre peuvent être résumées comme suit :

- Nous avons illustré les besoins pour de nouveaux mécanismes de sauvegarde de données pour dispositifs mobiles.
- Nous nous sommes fixé des objectifs de *sûreté de fonctionnement*.

En outre, nous avons exposé une solution basée sur la *sauvegarde coopérative* pour atteindre ces objectifs :

- Cette approche tire parti des *interactions spontanées* entre dispositifs.
- Les dispositifs participant partagent leurs espaces de stockage.
- Les dispositifs ayant collecté des données d'autres participants les envoient ensuite à un support de stockage sur Internet.
- En case de besoin, les données sont récupérées en contactant ce support de stockage sur Internet.

Enfin, nous avons présenté les travaux connexes dans différents domaines.

Chapitre 3. Évaluation analytique du système proposé

Ce chapitre propose une évaluation analytique de la sûreté de fonctionnement du service de sauvegarde coopératif proposé. Une partie des résultats présentés ici a été décrite dans [Hamouda 2006] et dans [Courtès *et al.* 2007a].

3.1. Introduction

Plusieurs stratégies de réplication peuvent être envisagées. Les blocs de données peuvent être dupliqués simplement, ou bien au moyen de techniques plus sophistiquées utilisant des *codes d'effacement*. Les codes d'effacement permettent d'augmenter la fragmentation puis la dissémination des données à répliquer. L'augmentation du niveau de fragmentation et dissémination des données est bénéfique à la confidentialité [Deswarte *et al.* 1991]. En revanche, son impact sur la disponibilité des données, particulièrement dans les scénarios envisagés, reste à explorer.

codes d'effacement

Dans ce chapitre, nous évaluons l'apport de MoSAIC sur la disponibilité des données stockées sur les dispositifs mobiles en fonction (i) de paramètres environnementaux, et (ii) de la stratégie de réplication choisie. Cette approche a pour but de nous aider à identifier les scénarios dans lesquels MoSAIC procure le plus grand bénéfice et à guider le choix d'une stratégie de réplication.

3.2. Contexte

Les codes d'effacement ont été beaucoup étudiés [Lin *et al.* 2004, Mitzenmacher 2004, Xu *et al.* 1999, Xu 2005, Weatherspoon & Kubiatowicz 2002]. Nous ne nous concentrons pas sur les algorithmes eux-mêmes mais sur leurs propriétés. Ces algorithmes peuvent être définis comme suit :

- pour une donnée d'entrée de k symboles, un code d'effacement produit $n \geq k$ *fragments* ;
- m fragments sont nécessaires et suffisant pour recouvrir la donnée d'origine, avec $k \leq m \leq n$; quand $k = m$, le code d'effacement est dit *optimal* [Xu *et al.* 1999].

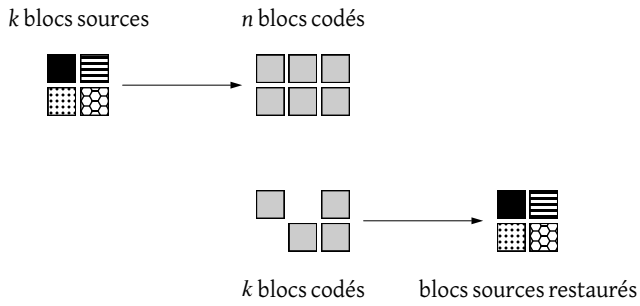

Figure 1. Codage et recouvrement des données avec un code d'effacement optimal, avec $k = 4$ et $n = 6$.

La figure 1 illustre le processus de codage et de recouvrement avec un code d'effacement optimal. Par la suite, nous faisons l'hypothèse que nous utilisons des codes optimaux. On note (n,k) un tel code.

Pour un coût de stockage égal, l'utilisation d'un code d'effacement permet de tolérer davantage de défaillances qu'une approche de « réplication simple », c'est-à-dire une réplication avec $k = 1$. À supposer que les n fragments soient stockées sur des dispositifs différents, on tolère $n - k$ défaillances pour un coût de stockage de $\frac{n}{k}$.

3.3. Méthodologie

On s'intéresse à la sauvegarde d'une donnée en la copiant sur un certain nombre de contributeurs *différents*, en supposant qu'ils la transféreront ultérieurement au support de stockage sur Internet. On fait l'hypothèse que la donnée est « sûre » (ne peut plus être perdue) une fois qu'elle a été envoyée sur Internet. On suppose également que chaque rencontre d'un contributeur donne une occasion de copier la donnée ou un fragment de la donnée si l'on utilise des codes d'effacement.

Afin d'évaluer le risque de perdre la donnée lorsque cette approche est utilisée, nous utilisons une modélisation sous forme de réseau de Petri stochastique généralisé que nous transformons ensuite en chaîne de Markov. Les processus stochastiques modélisés sont :

- un processus de taux α modélisant la rencontre du propriétaire avec un contributeur ;
- un processus modélisant l'accès à Internet d'un dispositif, de taux β_0 pour le propriétaire et β pour les contributeurs;
- un processus représentant la défaillance d'un dispositif, de taux λ_0 pour le propriétaire et λ pour les contributeurs.

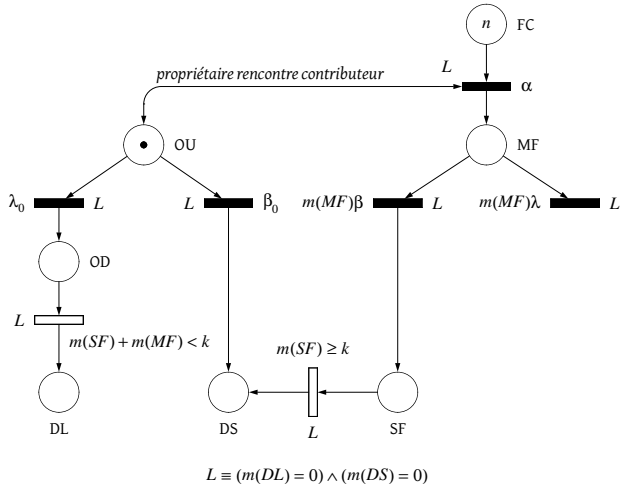

Figure 2. Réseau de Petri du processus de réplication et dissémination d'une donnée pour un code d'effacement (*n*,*k*).

La figure 2 montre le résultat de cette modélisation. Les places OU et OD dénotent les situations où le propriétaire et « vivant » ou « mort » respectivement. Le sous-réseau de droite décrit : (i) le processus de réplication menant à la copie de fragments (place MF), et (ii) le processus menant au stockage des fragments sur Internet (taux β) ou à leur perte (taux λ). Le marquage initial de la place FC correspond au nombre de fragments à créer. La place DS (*data safe*) correspond à l'état absorbant où la donnée est sûre, tandis que la place DL (*data lost*) correspond à l'état absorbant où la donnée est définitivement perdue. Enfin, L est le « prédicat de vivacité » du réseau qui est vrai si et seulement si $m(DS) = m(DL) = 0$: dès que DL ou DL contient un jeton, aucune transition ne peut plus être tirée.

probabilité de perte de donnée

Nous évaluons la *probabilité de perte de donnée*, notée PL, sur les chaînes de Markov générées à partir du réseau de Petri pour différents couples (*n*,*k*) [Kemeny & Snell 1960]. Pour mesurer l'amélioration apportée par MoSAIC, nous comparons PL avec PL_{ref}, la probabilité de perte de donnée dans un scénario sans MoSAIC où :

- le propriétaire ne coopère pas avec d'autres dispositifs;
- le propriétaire peut défaillir avec un taux λ_0;
- le propriétaire peut accéder à Internet et y sauvegarder ses données avec un taux β_0.

Dans ce scénario de référence, la probabilité de perte de la donnée est : $PL_{ref} = \frac{\lambda_0}{\lambda_0+\beta_0}$. On mesure le *facteur d'amélioration* apportée par MoSAIC, noté LRF (pour *loss reduction factor*), et qui vaut $LRF = PL_{ref}/PL$.

facteur d'amélioration

La probabilité de perte de la donnée dépend d'un certain nombre de paramètres (n, k, α, β, λ, β_0, and λ_0). Plutôt que de considérer la valeur absolue de chaque paramètre, nous considérons des rapports correspondants aux processus en concurrence. Par exemple, nous étudions *LRF* en fonction du taux de connectivité $\frac{\alpha}{\beta}$ et en fonction du rapport entre le taux d'accès à Internet et le taux de défaillance $\frac{\beta}{\lambda}$.

3.4. Résultats

Nous nous contentons ici de résumer les principaux résultats obtenus. Nous invitons le lecteur à se référer à la deuxième partie de cette thèse (en anglais) pour davantage de détails.

Dans un premier temps, nous faisons l'hypothèse que contributeurs et propriétaire se comportent de la même manière, c'est-à-dire que $\beta_0 = \beta$ et $\lambda_0 = \lambda$. La figure 3 montre le facteur d'amélioration *LRF* en fonction de deux rapports pour un code d'effacement (2,1) (c'est-à-dire réplication simple où la donnée est envoyée à 2 contributeurs différents). Trois observations peuvent être faites :

1. comme on pouvait s'y attendre, MoSAIC n'apporte aucune amélioration en termes de disponibilité des données lorsque $\frac{\alpha}{\beta} = 1$, c'est-à-dire lorsque la rencontre d'un contributeur n'est pas plus fréquente que l'accès à Internet ;
2. *LRF* atteint une asymptote après un certain seuil de $\frac{\beta}{\lambda}$;
3. l'amélioration est d'abord proportionnelle à $\frac{\alpha}{\beta}$ puis, à partir d'un certain seuil, atteint une asymptote.

Après une étude plus approfondie du comportement asymptotique de *LRF*, nous concluons que d'un point de vue disponibilité des données, la réplication simple ($k = 1$) est toujours préférable aux stratégies de réplication basées sur les codes d'effacement ($k > 1$) au delà d'un seuil $\frac{\beta}{\lambda}$. Pour des valeurs plus faibles de $\frac{\beta}{\lambda}$, on observe que les codes d'effacement ne fournissent un facteur d'amélioration plus élevé que la réplication simple que dans des scénarios très restreints.

Nous nous sommes également intéressés à l'effet de l'*efficacité* des participants sur *LRF*. L'efficacité d'un contributeur est caractérisée par le rapport $\frac{\beta}{\lambda}$ alors que celle du propriétaire est caractérisée par $\frac{\beta_0}{\lambda_0}$. Nous avons donc tracé *LRF* en fonction de $\frac{\beta/\lambda}{\beta_0/\lambda_0}$ d'une part, et du taux de connectivité du propriétaire $\frac{\alpha}{\beta_0}$ de l'autre. Nous observons que *LRF* devient inférieur à 10 lorsque les contributeurs deviennent 100 fois moins efficaces que le propriétaire.

Enfin, nous avons évalué des stratégies de réplication *hybrides*. Dans les stratégies évaluées précédemment, seul un fragment était distribué à chaque contributeur. Avec ces stratégies hybrides, nous étudions l'effet de la distribution d'un nombre de fragments compris entre 1 et k à chaque contributeur. On observe que ces stratégies ont un effet intermédiaire sur *LRF*, entre les stratégies à réplication simple et les stratégies à codes d'effacement étudiées précédemment.

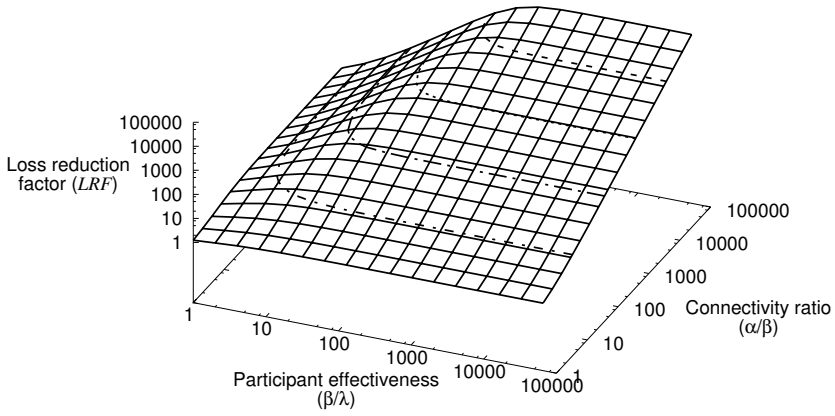

Figure 3. Facteur d'amélioration *LRF* pour un code d'effacement (2,1).

3.5. Travaux connexes

Les codes d'effacement ont été largement utilisés dans le domaine du stockage (réparti) [Lillibridge *et al.* 2003, Aiyer *et al.* 2005, Kubiatowicz *et al.* 2000, Goldberg & Yianilos 1998]. Quelques travaux se sont intéressés à l'analyse de leur impact sur la disponibilité des données. Par exemple, le temps moyen avant défaillance (MTTF) a été évalué analytiquement dans le cadre de stockage pair-à-pair sur Internet avec un processus de réparation [Kubiatowicz *et al.* 2000, Weatherspoon & Kubiatowicz 2002]. Les auteurs concluent que la réplication au moyen de codes d'effacement procure une meilleure disponibilité des données, en supposant toutefois que 90% des contributeurs stockant une donnée sont atteignables en n'importe quel instant.

Dans [Vernois & Utard 2004], une comparaison similaire entre codes d'effacement et réplication simple est menée et conclue que les codes d'effacement sont inappropriés (d'un point de vue disponibilité) dans le cas où les participants pris individuellement ont une faible disponibilité. En revanche, ces études supposent l'existence d'un processus de *réparation* des données qui est inexistant dans notre cas. Enfin, une comparaison est également proposée dans [Lin *et al.* 2004], montrant que l'espace des scénarios est séparé entre une partie où les codes d'effacement procurent de meilleurs résultats et une partie où c'est au contraire la réplication simple. Ces résultats sont cohérents avec les nôtres.

Les codes d'effacement ont également été considérés dans le cadre des réseaux tolérant les retards (DTN) [Zhang 2006, Wang *et al.* 2005, Jain *et al.* 2005, Liao *et al.* 2006]. À notre connaissance, aucune évaluation analytique n'a été menée dans ce cadre. Une

analyse basée sur la simulation de DTN est proposée dans [Jain *et al.* 2005] et conduit à des résultats comparables à ceux de [Lin *et al.* 2004].

3.6. Résumé

Dans ce chapitre, nous avons abordé les points suivants :

- un modèle du processus de sauvegarde coopérative a été proposé, utilisant les réseaux de Petri et les chaînes de Markov, ainsi qu'une méthodologie pour l'évaluer d'un point de vue sûreté de fonctionnement ;
- l'évaluation a permis d'identifier des scénarios où l'approche coopérative est bénéfique ; en particulier, nous avons vu que MoSAIC était bénéfique (d'au moins un ordre de grandeur par rapport à une sauvegarde sans MoSAIC) quand $\frac{\beta}{\lambda} > 2$ et $\frac{\alpha}{\beta} > 10$;
- nous avons montré que MoSAIC diminuait la *probabilité de perte des données* par un facteur inférieur ou égal au rapport entre le taux de rencontre de contributeurs et le taux de connexion à Internet ;
- l'approche coopérative n'améliore plus la disponibilité des données lorsque les contributeurs sont 100 fois moins efficaces que le propriétaire ;
- les stratégies de réplication utilisant des codes d'effacement n'améliorent la disponibilité des données que dans des scénarios très restreints.

Ces résultats peuvent guider le choix de stratégies de réplication en fonction du contexte.

Chapitre 4. Techniques de stockage réparti

Nous nous intéressons à présent aux mécanismes de stockage de notre service de sauvegarde coopérative. Nous identifions d'abord les exigences du service. Nous présentons ensuite différentes possibilités pour mettre en œuvre un support de stockage satisfaisant ces propriétés. Notre mise en œuvre de la couche de stockage est brièvement décrite, et nous donnons les résultats d'une évaluation expérimentale de différents mécanismes.

Une partie des résultats présentés dans ce chapitre a été publiée dans [Courtès *et al.* 2006].

4.1. Propriétés attendues de la couche de stockage

Au chapitre 2, nous avons défini des objectifs de sûreté de fonctionnement ainsi que des exigences de plus bas niveau requis par le processus de sauvegarde coopérative par réseau sans fils. Ces objectifs de sûreté de fonctionnement étaient de traiter les menaces à la confidentialité, au respect de la vie privée, à l'intégrité et authenticité, et à la disponibilité. Les exigences de bas niveau étaient de fournir des moyens de gérer la fragmentation des données et des techniques de sauvegarde efficaces en termes de consommation énergétique et de bande passante. Nous détaillons ici ces objectifs :

- **Efficacité du stockage.** Afin de tirer le meilleur parti des rencontres avec les contributeurs, il convient d'exploiter le mieux possible l'espace de stockage qu'ils mettent à disposition. Également, afin de maîtriser les coûts énergétiques, il est préférable de réduire la quantité de données à transférer [Stemm *et al.* 1997]. Dans les deux cas, il s'agira d'utiliser des méthodes de *compression* des données.
- **Blocs de données de petite taille.** Les rencontres avec des contributeurs étant imprévisibles et potentiellement de courte durée, les transferts de données devront se limiter à des petites tailles afin de maximiser leurs chances de réussite.
- **Atomicité de la sauvegarde.** Il sera pratiquement impossible de stocker, par exemple, un fichier ou un ensemble de fichier sur un seul contributeur, ce qui mène à une fragmentation et dissémination des données. Cependant, il est important que la sauvegarde reste dans un état *cohérent* (au sens des propriétés

ACID des bases de données [Gray 1981]) : la sauvegarde doit être soit réalisée *et* recouvrable, soit non-recouvrable.

- **Détection d'erreurs.** Les modifications des données aussi bien accidentelles que malveillantes doivent pouvoir être détectées.
- **Chiffrement.** La couche de stockage doit avoir recours au chiffrement des données pour garantir leur confidentialité.
- **Redondance.** Bien entendu, les sauvegardes devront être suffisamment redondantes, d'autant plus qu'on ne fait pas nécessairement confiance aux contributeurs.

4.2. Techniques de stockage satisfaisant nos exigences

Pour chacune des exigences que nous avons identifiées, nous passons en revue des techniques de stockage pouvant être satisfaisantes.

4.2.1. Efficacité du stockage

stockage à instance unique

Dans les services de stockage coopératifs à grand échelle, on économise souvent l'espace de stockage en ne stockant qu'une seule fois chaque donnée élémentaire. Cette propriété est connue sous le nom de *stockage à instance unique* [Bolosky *et al.* 2000]. Il a été montré que cette approche réduit sensiblement la quantité de données à stocker dans le cadre des systèmes d'archivage [Quinlan & Dorward 2002, You *et al.* 2005], du partage de fichiers pair-à-pair [Bennett *et al.* 2002], de la sauvegarde pair-à-pair [Cox *et al.* 2002, Landers *et al.* 2004], des systèmes de fichiers réseau [Muthitacharoen *et al.* 2001] et des outils de synchronisation des données à distance [Tridgell & Mackerras 1996].

compression différentielle

La *compression différentielle* a été proposée pour des systèmes d'archivage multi-versions [You & Karamanolis 2004, You *et al.* 2005, Kulkarni *et al.* 2004]. Toutefois, cette approche apparaît inadaptée à notre contexte car (i) elle requière l'accès à tous les fichiers déjà stockés, (ii) elle demande beaucoup de capacité de calcul et d'espace mémoire et (iii) les *chaînes de différences* résultantes peuvent diminuer la disponibilité des données [You *et al.* 2005].

compression sans pertes

La *compression sans pertes* classique permet d'éliminer la redondance au sein d'unités de stockage (par exemple, de fichiers) individuelles. En tant que tel, cette technique est un complément au stockage à instance unique.

4.2.2. Blocs de données de petite taille

Nous nous intéressons aux techniques et algorithmes permettant de (1) « découper » un flux de données en blocs et (2) de créer des méta-données décrivant comment ces blocs doivent être assemblés pour produire le flux d'origine.

Le découpage en blocs de taille fixe est l'approche la plus simple. Combinée au stockage à instance unique, elle permet d'améliorer légèrement la compression entre fichiers ou versions de fichiers [Quinlan & Dorward 2002]. Une autre possibilité est le *découpage fonction du contenu* [Manber 1994]. Avec cette approche, les bornes des blocs sont définies en fonction du contenu du flux d'entrée, ce qui permet de détecter des ressemblances entre différents flux d'entrée. Combiné avec le stockage à instance unique, cette technique améliore donc la compression [Cox *et al.* 2002, You *et al.* 2005, Muthitacharoen *et al.* 2001].

découpage fonction du contenu

Une fois les blocs créés, il s'agit de créer des méta-données indiquant comment reconstituer la donnée d'origine. Pour ce faire, il doit d'abord être possible de *nommer* les blocs[1]. Le *schéma de désignation des blocs* utilisé doit éviter les collisions entre noms de blocs (un nom de bloc ne doit pas être ré-attribué, même après une perte des données), et il doit être de préférence indépendant du contributeur stockant le bloc, ce qui permet de pré-calculer les noms de bloc. La section 4.2.4 discutera d'un schéma de désignation répandu.

schéma de désignation des blocs

Une structure de donnée souvent utilisée pour décrire l'assemblage des blocs de données et un arbre tel que celui de la figure 4 : les feuilles de l'arbre sont les blocs de données, les nœuds intermédiaires I_k contiennent des méta-données et les racines R_1 et R_0 pointent sur le premier niveau de nœuds intermédiaires. On observe sur cette figure que les données (feuilles) communes aux deux arbres ont pu être partagées, illustrant la compression fournie par le stockage à instance unique, par exemple entre deux version successives d'un fichier [Quinlan & Dorward 2002].

Avec ces éléments, les contributeurs n'ont pas besoin de savoir comment les propriétaires découpent leurs données, ni quelle structure de méta-données ils utilisent. Le contributeurs peuvent donc se contenter de fournir deux primitives :

- `put (clef, donnée)` qui stocke `donnée` et l'associe à `clef`, un identifiant de bloc;
- `get (clef)` qui retourne la donnée associée à `clef`.

Cette approche a été suivie par différents systèmes de stockage [Quinlan & Dorward 2002, Bennett *et al.* 2002, Cox *et al.* 2002].

4.2.3. Atomicité

Une manière simple de garantir l'atomicité des sauvegardes est de suivre une approche *ajout-seulement* des données [Cox *et al.* 2002, Quinlan & Dorward 2002, Santry *et al.* 1999, Shapiro & Vanderburgh 2002] : les données sont toujours ajoutées à l'espace de stockage et ja-

ajout-seulement

[1] Par la suite, on dira aussi bien « nom de bloc », qu'« identifiant » ou « clef ».

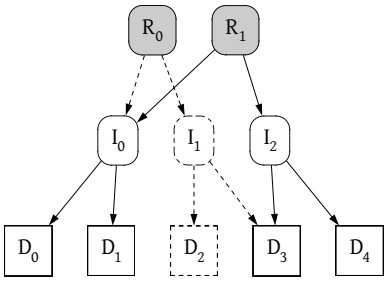

Figure 4. Arbre décrivant l'assemblage de blocs.

mais modifiées sur place. Par conséquent, l'insertion de contenu dans l'espace de stockage est atomique. En outre, il a été montré que le coût de cette approche en termes de stockage est faible dans beaucoup de scénarios, à condition qu'une forme de stockage à instance unique soit utilisée [Santry *et al.* 1999, Gibson & Miller 1998, Quinlan & Dorward 2002].

4.2.4. Détection d'erreurs

Des codes détecteurs d'erreurs doivent être calculés, soit au niveau du flux d'entrée, soit au niveau des blocs individuels. Comme à la fois des modifications accidentelles et malveillantes doivent pouvoir être détectées, il nous faudra utiliser des *fonctions de hachage cryptographiques* [NESSIE Consortium 2003a, NESSIE Consortium 2003b]. Outre l'intégrité des données, leur authenticité doit aussi pouvoir être vérifiée. À cette fin, une partie des méta-données pourra être signée avec une signature électronique.

fonctions de hachage cryptographiques

Les fonctions de hachage cryptographiques ont souvent été proposées comme moyen de désignation de blocs de données : il s'agit de désigner les blocs par le résultat d'une fonction de hachage cryptographique appliquée à leur contenu. En outre, deux blocs dont le contenu est identique auront le même nom, ce qui procure un moyen simple et efficace pour mettre en œuvre le stockage à instance unique [Quinlan & Dorward 2002, Bennett *et al.* 2002, Muthitacharoen *et al.* 2001, Tridgell & Mackerras 1996]. En effet, les collisions accidentelles sont statistiquement extrêmement improbables, et la création de collisions (par un participant malveillant, par exemple) demande des temps de calcul la mettant hors de portée [NESSIE Consortium 2003a, Black 2006].

4.2.5. Chiffrement

Le chiffrement des données peut être appliqué soit au flux d'entrée, soit aux blocs. Toutefois, si un chiffrement asymétrique est utilisé, cette approche empêche de mettre en œuvre du stockage à instance unique entre les blocs provenant de différents propriétaires. Le *chiffrement convergent* [Cox *et al.* 2002] permet de résoudre ce problème, en utilisant un algorithme de chiffrement symétrique avec pour clef le condensé de la donnée en clair

chiffrement convergent

(condensé fourni par une fonction de hachage cryptographique). Le chapitre 5 reviendra plus en détails sur ces aspects.

4.2.6. Redondance

Le chapitre 3 a traité des différentes approches (redondance simple, codes d'effacement) permettant de stocker les blocs de données de manière redondantes. Il est cependant possible d'appliquer les codes d'effacement directement au niveau du flux d'entrée au lieu des blocs résultant du découpage d'un flux. [Jain *et al.* 2005] montre, dans le cadres des réseaux tolérant les retards, que cette approche est préférable d'un point de vue disponibilité des données. Toutefois, dans notre cas, un tel choix nous empêcherait d'utiliser d'autres optimisations telles que le stockage à instance unique des blocs issus d'un flux de données.

4.3. Mise en œuvre

Nous avons développé une bibliothèque C, nommée *libchop*, qui permet de combiner une partie des techniques de stockage mentionnées précédemment. Le flux des données d'entrée jusqu'à leur stockage sous forme de blocs est schématisé par la figure 5 : chaque boîte représente un composant de *libchop*, et chaque flèche représente le flux de données entre composants.

Le composant `stream` représente les flux d'entrée (tels que le contenu d'un fichier) et le composant `block_store` représente le support de stockage des blocs, c'est-à-dire essentiellement les primitives `put` et `get` discutées précédemment. Les composants de type `chopper` découpent les flux d'entrée en blocs. Les composants `block_indexer` mettent en œuvre la désignation des blocs de données tandis que les `stream_indexer` produisent les méta-données décrivant l'assemblage des blocs.

Enfin, les composants de type `filter` effectuent des traitements supplémentaires sur les données. En particulier, nous avons mis en œuvre des « filtres » de compression et décompression utilisant *zlib* [Deutsch & Gailly 1996], *bzip2* [Seward 2007] et LZO [Oberhumer 2005]. LZO met en œuvre un algorithme de compression sans perte utilisant beaucoup moins de ressources (calcul et mémoire) que les deux autres, mais permettant d'atteindre des taux de compression plus faibles.

4.4. Évaluation préliminaire

Nous avons utilisé *libchop* pour comparer (i) le temps de calcul et (ii) le taux de compression atteint avec différentes combinaisons des techniques de stockage dont nous avons parlé. Pour ce faire, nous avons choisi des ensembles de fichiers (données d'entrée) pouvant être représentatifs du *type* de données traitées sur un dispositif mobile : différente versions du code source d'un logiciel, c'est-à-dire ensemble de fichiers textes comportant beaucoup

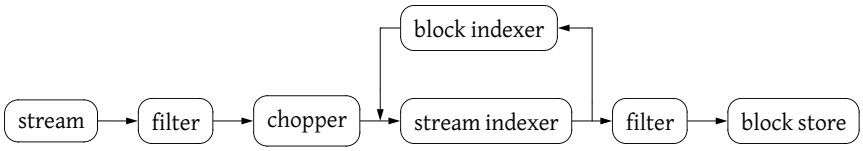

Figure 5. Flux des données à stocker au travers des composants de *libchop*.

de similarités, fichiers audio, et boîte aux lettres électroniques (fichier texte unique de grande taille).

Cette évaluation nous a permis de tirer plusieurs conclusions :

- la mise en œuvre du stockage à instance unique au moyen de fonctions de hachages cryptographiques est peu coûteuse en temps de calcul ;
- le stockage à instance unique est surtout profitable dans le cas des versions successives de fichiers textes ;
- la taille des méta-données en utilisant des condensés (SHA-1) de 20 octets pour désigner les blocs est négligeable, même avec des tailles de bloc assez faibles ;
- le découpage fonction du contenu [Manber 1994] ne procure une amélioration du taux de compression que dans le cas du stockage de version successives des fichiers textes, et ce au coût de temps de calculs élevés ;
- la combinaison de compression sans perte du flux d'entrée et de découpage en blocs de taille fixe offre le meilleur compromis pour les trois types de données d'entrée considérés.

Cette évaluation nous donne donc de précieux indices sur le choix des mécanismes de stockage les plus adaptés à notre contexte.

4.5. Travaux connexes

Des évaluations similaires de mécanismes de stockage ont déjà été menées dans le cadre de systèmes d'archivages de données, avec des résultats comparables aux nôtres quant aux taux de compression [You *et al.* 2005, You & Karamanolis 2004, Kulkarni *et al.* 2004]. Cependant, ces études ne considèrent pas les temps de calcul nécessaires. Notre évaluation ajoute, en outre, des combinaisons impliquant davantage d'algorithmes de compression.

En préparant notre évaluation, nous avons supposé que le temps de calcul et la quantité d'énergie consommée par le processeur étaient proportionnels, et que les coûts de transmission de données par réseau sans fils étaient également proportionnels à la quantité de données transmises. Une étude plus précise de ces coûts a été proposée, se basant sur une instrumentation du matériel [Barr & Asanovic 2006]. Les auteurs ont conclu que la consommation énergétique du processeur est effectivement proportionnelle au temps de calcul.

Ils notent également que, bien que les interfaces Wi-Fi soient très gourmandes en énergie, l'énergie consommée pour compresser des données dépasse souvent l'énergie pour les transmettre, *zlib* avec un faible taux de compression et LZO étant des exceptions notables.

4.6. Résumé

Les contributions de ce chapitre peuvent être résumées comme suit :

- nous avons identifié six critères devant être remplis par la couche de stockage du service de sauvegarde coopérative;
- nous avons passé en revue différentes techniques de stockage;
- un aperçu de notre mise en œuvre de la couche de stockage a été donné;
- une évaluation de plusieurs combinaisons des techniques de stockage a été menée.

Les principales contributions de cette évaluation sont :

- la mise en œuvre du stockage à instance unique au moyen de fonctions de hachages cryptographiques est peu coûteuse en temps de calcul et profitable en termes de taux de compression dans certains cas;
- le stockage à instance unique est surtout profitable dans le cas des versions successives de fichiers textes;
- la combinaison de compression sans perte du flux d'entrée et de découpage en blocs de taille fixe offre le meilleur compromis pour les trois types de données d'entrée considérés.

Ce chapitre nous donne donc les bases de la couche de stockage de notre service de sauvegarde coopérative.

Chapitre 5. Coopération sécurisée

Le chapitre 2, section 2.2 a décrit des menaces à la confidentialité, l'intégrité et la disponibilité dans le cadre du service coopérative de sauvegarde. Dans les chapitres précédents, nous nous sommes concentrés principalement sur les menaces à la disponibilité des données résultant de défaillances *accidentelles.* Dans ce chapitre, nous nous intéressons aux menaces à la disponibilité des données ou du service dues à la malveillance de participants. Une partie de ces travaux a été publiée dans [Courtès *et al.* 2007b].

5.1. Introduction

En section 2.2, nous listions les menaces suivantes dans le cadre de la coopération entre participants ne se faisant pas confiance : menaces contre la confidentialité et au respect de la vie privée, menaces contre l'intégrité et l'authenticité des données, menaces contre la disponibilité. Certaines de ces menaces ont déjà été traitées au chapitre 4. Ici, nous abordons la protection contre les *attaques en déni de service* contre le service ou ses utilisateurs :

- la *rétention de données*, où un contributeur refuse de donner les données qu'il est supposé avoir stocker à leur propriétaire ;
- l'*inondation*, où un participant cherche volontairement à épuiser les ressources (l'espace de stockage) offertes par le service ;
- l'*égoïsme*, où des participants profitent du service sans y contribuer en retour.

Ces attaques sont bien connues dans le domaine du partage de fichiers ou de la sauvegarde pair-à-pair sur Internet [Bennett *et al.* 2002, Lillibridge *et al.* 2003, Cox *et al.* 2002], et sont aussi en parti traitées dans le domaine des protocoles de routage pour réseaux *ad hoc* [Michiardi & Molva 2002, Buttyán & Hubaux 2003]. Il s'agit là de *menaces contre la coopération.*

Dans ce chapitre, nous identifions les exigences qui doivent être remplies pour pouvoir limiter l'impact de ces attaques. Nous proposons des mécanismes primitifs décentralisés et *autogérés*, n'imposant aucune politique particulière de coopération.

autogérés

5.2. Aperçu de la couche de stockage

Nous résumons les principaux aspects de la couche de stockage présentée au chapitre 4 sous un angle sécurité.

Tout d'abord, les données d'entrée sont découpées en blocs par le propriétaire qui crée également les méta-données nécessaires pour pouvoir les assembler. Chaque bloc se voit attribuer un identifiant par le propriétaire, suivant le schéma de son choix. Lorsqu'un contributeur est rencontré, des blocs de données lui sont envoyés en utilisant la primitive `put`. Le contributeur doit conserver l'association entre un bloc de données et son nom ; en outre, comme le choix des noms est spécifique à chaque propriétaire, il doit maintenir les blocs de chaque propriétaire dans des espaces séparés.

Généralement, les propriétaires vont également chiffrer leurs données et méta-données. Pour être capables de vérifier l'authenticité des données qu'ils reçoivent lors d'une restauration, les propriétaires devront également signer, au moyen d'une signature électronique, tout ou partie des méta-données produites.

Dans cette approche, seul le protocole de stockage (les primitives `put` et `get`) est imposé, les autres choix étant laissés à la discrétion de chaque propriétaire.

5.3. Tirer parti de la coopération

Dans cette section, nous décrivons notre approche pour traiter les problèmes de sécurité que nous avons évoqués.

5.3.1. Démarche de conception

Deux approches sont répandues pour traiter ce les attaques en déni de service dans les réseaux mobiles *ad hoc* et réseaux pair-à-pair : au moyen d'un *domaine d'autorité singulier* ou au moyen d'*auto-organisation*, ne dépendant d'aucune autorité unique à aucun moment [Capkun *et al.* 2003]. Nous considérons que les systèmes dépendant d'une autorité responsable d'appliquer des sanctions externes, comme c'est le cas avec BAR-B [Aiyer *et al.* 2005], s'apparente à la première catégorie. De même, l'utilisation de « modules de sécurité résistant aux altérations » (ou *tamper-resistant security modules*) assurant l'application de règles et de protocoles au niveau de chaque participant, comme dans [Buttyán & Hubaux 2000] fait également partie de cette catégorie.

auto-organisation

Nous sommes de l'avis que dépendre d'une autorité centrale peut, selon certaines politiques de sécurité, être considéré comme une menace de sécurité : pourquoi un utilisateur ferait-il confiance à une entité externe juste parce qu'on lui a dit qu'elle était « de confiance » ? En outre, une telle autorité constitue un *point singulier de confiance* : sa défaillance ou compromission peut entraîner l'indisponibilité du service qui en dépend. Pour ces raisons, nous focalisons sur des solutions auto-organisées. Une telle approche auto-

organisée correspond également bien aux réseaux mobiles *ad hoc* qui sont eux-même auto-organisés.

Par conséquent, les participants peuvent utiliser une politique de coopération de leur choix, et il nous paraît important de laisser cette possibilité. Une observation clef, toutefois, est que le point commun de toute politique de coopération est qu'elle requiert des mécanismes d'*imputabilité*, c'est-à-dire des moyens permettant d'imputer les actions qui sont faites à leur auteur [Dingledine *et al.* 2001]. imputabilité

5.3.2. Identifiants de dispositifs uniques, autogérés et vérifiables

Les dispositifs participant doivent pouvoir se désigner les uns les autres pour pouvoir (i) mettre en œuvre les espaces de stockage par propriétaire (section 5.2), et (ii) permettre l'imputabilité.

Le mécanisme de désignation des dispositifs utilisé à cette fin doit satisfaire plusieurs critères. Il doit être possible de créer son propre identifiant sans avoir recours à une autorité externe, de sorte à ce que le service soit auto-organisé. Les noms des dispositifs doivent également être (statistiquement) *uniques* et *indépendants du contexte*. Enfin, il doit être possible d'*authentifier* le lien entre un nom et un dispositif ou, en d'autres termes, de vérifier qu'un dispositif est bien propriétaire du nom qu'il prétend avoir. Ce dernier point est indispensable pour permettre l'imputabilité.

Un certain nombre de schémas de désignation classiques, telles que des adresses IP, ne répondent pas à tous ces critères. Les adresses « statistiquement uniques et cryptographiquement vérifiables » (SUCV) proposées pour *Mobile IPv6*, cependant, répondent à ces attentes [Montenegro & Castelluccia 2002]. Nous proposons de nous baser sur la cryptographie asymétrique pour produire un tel mécanisme de désignation, utilisant les clefs publiques des dispositifs (ou leurs condensés) pour nommer les dispositifs. Les dispositifs doivent en outre s'authentifier mutuellement avant d'interagir pour vérifier la véracité du lien entre le nom qu'ils utilisent et eux-mêmes.

5.3.3. Assurer l'intégrité des communications

Puisqu'il s'agit de permettre l'imputabilité des actes de chaque participant, il est primordial de pouvoir garantir l'intégrité des communications entre participants, de sorte qu'un utilisateur malveillant ne puisse pas, par exemple, altérer le contenu des messages échangés entre deux participants (des requêtes `put` et `get`) sans que cela soit détecté. Des protocoles cryptographiques bien connus apportent une solution à ce problème, comme nous le verrons en section 5.4.

5.3.4. Limiter l'impact des attaques sybillines

attaques sybillines

Les identifiants des dispositifs participants étant générés de manière auto-organisée par chacun des participants, notre service est sujet aux *attaques sybillines* [Douceur 2002, Marti & Garcia-Molina 2003] : les participants peuvent changer d'identifiants comme ils le veulent, ce qui peut leur permettre de ne pas se faire imputer leurs actions passées, y compris leurs comportements malveillants.

Le mécanisme de désignation proposé ne peut pas par lui-même résoudre ce problème. De plus, dans un système où les identifiants sont produits de manière auto-organisée, il est impossible d'*empêcher* complètement ce type d'attaques. C'est aux politiques de coopération des participants de rendre se type d'attaque *moins attractif* . Heureusement, il a été montré que certaines politiques étaient assez efficaces contre ces attaques [Marti & Garcia-Molina 2003, Michiardi & Molva 2002, Buchegger & Boudec 2003]. La plupart des politiques envisageables auront pour dénominateur commun de dédier peu de ressources aux inconnus et d'être en revanche davantage coopératives avec les dispositifs qui ont déjà coopéré par le passé.

Enfin, le contexte mobile réduit sensiblement la portée de ce type d'attaques puisqu'il est nécessaire d'être à *proximité physique* des participants pour leur « extorquer » de l'espace de stockage.

5.3.5. Permettre une large gamme de politiques de coopération

rapports sociaux

observations comportementales

Les politiques de coopération des utilisateurs définissent l'ensemble de règles déterminant sous quelles conditions leur dispositif coopère. On peut imaginer principalement deux classes de politiques de coopération : celles basées sur les *rapports sociaux* sous-jacents entre utilisateurs, et celles basées sur les *observations comportementales* des dispositifs [Grothoff 2003, Lai *et al.* 2003, Michiardi & Molva 2002, Buchegger & Boudec 2003]. Notre objectif est de permettre la mise en œuvre de ces deux types de politiques de coopération.

La première catégorie de politiques se base sur les liens sociaux entre individus, par exemple en ne coopérant qu'avec ses proches, ou bien, de manière plus sophistiquée, en tirant parti du phénomène de « petit monde » qui caractérise les rapports humains [Milgram 1967, Capkun *et al.* 2002]. Les politiques de coopération de la deuxième catégorie pourront enregistrer les observations du comportement des dispositifs rencontrés et ensuite s'en resservir pour guider leurs choix de coopération [Grothoff 2003, Lai *et al.* 2003]. Ces informations comportementales pourront aussi être échangées entre dispositifs pour faciliter le passage à l'échelle ; on parle alors de mécanisme de *réputation* [Lai *et al.* 2003, Buchegger & Boudec 2003, Michiardi & Molva 2002]. Les travaux cités ont montré que ce type de politique de coopération réduit l'impact des attaques sybillines.

réputation

5.4. Considérations pratiques

Nous avons étudié les protocoles réseau permettant de répondre à nos attentes. Notre choix s'est porté sur le protocole TLS [Dierks *et al.* 2006] avec l'extension permettant l'authentification utilisant des *certificats OpenPGP* [Mavrogiannopoulos 2007]. Cette dernière permet de directement utiliser une paire de clefs OpenPGP générée pour chaque dispositif pour l'authentification mutuelle. Il est donc ensuite possible d'utiliser par exemple l'*empreinte* de la clef publique d'un participant pour le désigner [Callas *et al.* 1998].

certificats OpenPGP

Enfin, l'intégrité et l'authenticité des messages échangés dans le cadre d'une session TLS est garantie grâce à l'utilisation par le protocole de codes d'authentification des messages HMAC [Dierks *et al.* 2006].

5.5. Travaux connexes

Les attaques en déni de service contre les services coopératifs ont fait l'objet de nombreux travaux. Ils ont notamment été étudiés dans le cadre des services de sauvegarde pair-à-pair [Lillibridge *et al.* 2003, Cox & Noble 2003, Grothoff 2003, Oualha *et al.* 2007a]. Un excellent état de l'art des méthodes existantes pour traiter ce type d'attaques dans les services pair-à-pair est [Dingledine *et al.* 2001]. Relativement peu de propositions ont été faites dans le cadre des réseaux tolérant les retards (DTN) [Farrell & Cahill 2006, Harras *et al.* 2007]; une proposition a été faite dans [Fall 2003] mais elle repose sur une approche centralisée.

Beaucoup de politiques d'*incitation à la coopération* ont été proposées. Une solution est d'imposer des échanges symétriques (« œil pour œil, dent pour dent ») [Lillibridge *et al.* 2003] ou l'utilisation de droits de stockage transférables [Cox & Noble 2003]. Ensuite, il est aussi nécessaire d'*évaluer le service rendu* par un contributeur. Des solutions ont été proposées dans le cadre de services de sauvegarde sur Internet mais elles demandent que les contributeurs soient accessibles par le réseaux, ce qui est inadapté aux réseaux mobiles [Lillibridge *et al.* 2003, Cox *et al.* 2002, Cox & Noble 2003, Aiyer *et al.* 2005]. Une approche novatrice a été proposée pour permettre la délégation à d'autres participants de cette capacité d'évaluation, mais la manière dont les résultats de l'évaluation pourraient être pris en compte par les propriétaires n'est pas encore clairement définie [Oualha *et al.* 2007b].

incitation à la coopération

La note fournie par l'évaluation du service rendu, c'est-à-dire une trace du comportement d'un contributeur, peut être ensuite utilisée pour guider les choix de coopération. On a alors affaire à des politiques se basant sur des niveaux de confiance [Grothoff 2003] ou de réputation [Lai *et al.* 2003, Michiardi & Molva 2002, Buchegger & Boudec 2003].

Les problèmes de *désignation* en environnement décentralisé ont été étudiés dans le cadre de la programmation répartie [Miller 2006] et aussi dans le contexte des infrastructures à clef publique [Ellison *et al.* 1999, Ellison 1996]. Au niveau réseau, une des préoccupations de *Mobile IPv6* a été de donner aux nœuds réseau la possibilité de vérifier la « propriété » d'une adresse, c'est-à-dire d'authentifier le lien entre une adresse réseau et une entité; ces travaux ont mené à la proposition d'identifiants « statistiquement uniques

et cryptographiquement vérifiables », une solution similaire à la nôtre [Montenegro & Castelluccia 2002].

Enfin, les attaques sybillines ont été décrites dans [Douceur 2002]. Dans [Marti & Garcia-Molina 2003], les auteurs montrent qu'un système de réputation permet de réduire leur impact.

5.6. Résumé

Les contributions de ce chapitre peuvent être résumées comme suit :

- nous avons identifié des menaces à la sécurité d'un service de sauvegarde coopérative auto-organisé;
- nous avons proposé des primitives cryptographiques en montrant qu'elles remplissaient nos exigences; ces primitives ne mettent en œuvre aucune politique de coopération particulière;
- les systèmes utilisant des identifiants autogérés sont sujets aux attaques sybillines; nous avons discuté leur impact dans notre contexte et montré que des politiques de coopération pouvaient être mises en œuvre pour le réduire;
- des aspects pratiques de mise en œuvre ont été abordés et nous avons proposé l'utilisation du protocole TLS avec des certificats OpenPGP.

Le chapitre suivant décrit la mise en œuvre du service et l'utilisation des mécanismes proposés ici.

Chapitre 6. Mise en œuvre

Jusqu'à présent, nous avons abordé différents aspects du service de sauvegarde coopérative pour dispositifs mobiles. Dans ce chapitre, nous nous attachons à décrire notre prototype se basant sur les résultats et propositions des chapitres précédents.

6.1. Aperçu

Notre prototype est écrit en langage Scheme [Kelsey *et al.* 1998] et utilise GNU Guile [Jaffer *et al.* 1996]. Il utilise la bibliothèque *libchop* présentée au chapitre 4. Il comprend un *démon de sauvegarde* et un ensemble d'outils client permettant à l'utilisateur d'interagir avec lui. Le choix du langage de programmation était motivé, outre la préférence de l'auteur, par sa clarté lorsqu'un style purement fonctionnel est adopté, et par le besoin de pouvoir paramétrer les différents algorithmes du démon, ce qui est rendu possible par l'utilisation de fonctions d'ordre supérieur.

démon de sauvegarde

L'utilisateur doit fournir une paire de clefs OpenPGP qui sera ensuite utilisée par le démon lorsqu'il interagit avec d'autres instances, comme nous l'avons vu au chapitre précédent.

6.2. Compléments à la couche de stockage

La couche de stockage présentée au chapitre 4 ne se préoccupe pas de certains aspects de haut niveau liés au stockage. En particulier, elle ne produit pas de méta-données relatives aux fichiers (noms de fichiers, dates, etc.) ni aux versions. Notre prototype apporte donc un complément dans ces domaines. D'abord, des *répertoires* sont stockés. Un répertoire contient une liste d'associations entre un nom de fichier et l'identifiant permettant de le restaurer. Ensuite, une *chaîne de révisions* est produite. Chaque élément de la chaîne pointe vers son répertoire courant ainsi que vers la révision précédente; chaque révision contient aussi une estampille temporelle et de manière optionnelle d'autres champs fournis par l'utilisateur.

répertoires

chaîne de révisions

La figure 6 illustre le lien entre ces structures de donnés. On retrouve, à titre d'exemple, l'arbre décrivant l'assemblage de blocs qui représentent le contenu d'un fichier. On voit que deux versions du fichier `/src/chbouib.c` ont des données en commun, ce qui est le produit de l'utilisation de stockage à instance unique.

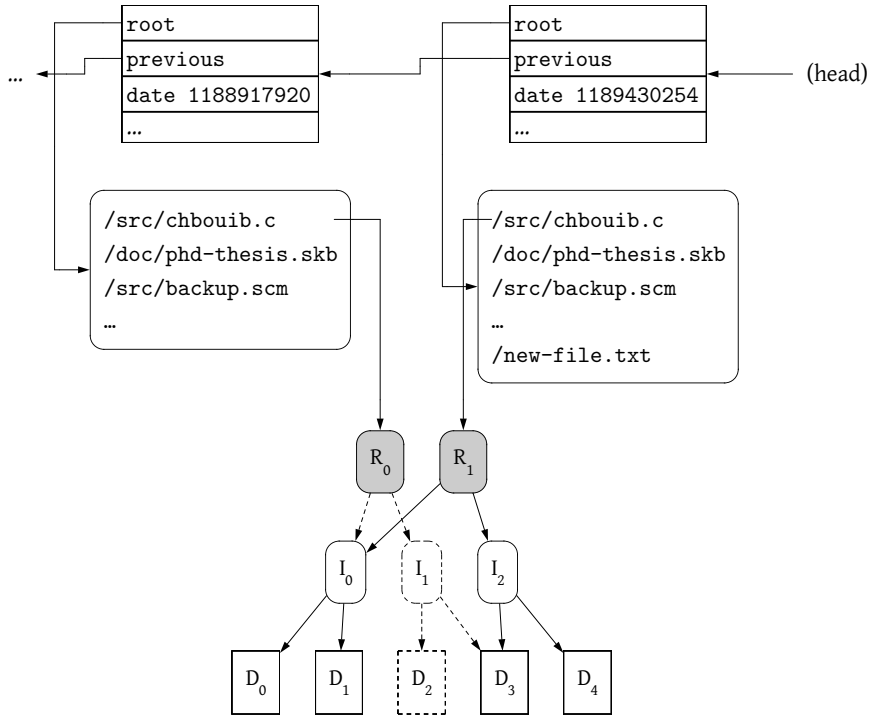

Figure 6. Révisions, répertoires, et contenus de fichiers.

6.3. Activités de sauvegarde coopérative

Nous décrivons ainsi le déroulement des principales activités du démon de sauvegarde.

6.3.1. Stockage des données et file de blocs

Lorsqu'un fichier de l'utilisateur est modifié, le démon de sauvegarde le découpe son contenu en blocs et crée les méta-données nécessaires à son assemblage, suivant le processus de stockage de *libchop* (chapitre 4). Il crée également une nouvelle version du répertoire et une nouvelle révision. En outre, le démon rajoute les blocs ainsi créés dans une *file de blocs à répliquer*. À chaque bloc de la file est associé la liste des contributeurs qui disposent d'un exemplaire du bloc. Cette file est ensuite utilisée lorsqu'une occasion de répliquer le données se présente.

file de blocs à répliquer

6.3.2. Réplication opportuniste

Avec la file de blocs à répliquer, le démon n'a plus qu'à la parcourir et à sélectionner les blocs à répliquer lorsqu'une occasion de réplication se présente. Notre prototype permet

de spécifier une politique de réplication en lui fournissant des prédicats indiquant si un bloc doit être répliqué.

Un premier prédicat permet de dire si, d'une manière générale, le bloc qui lui a été passé doit encore être dupliqué. Un exemple de tel prédicat est une fonction qui renvoie « vrai » lorsque la liste des contributeurs disposant d'un exemplaire du bloc comporte moins de deux éléments. Le deuxième prédicat doit répondre à la question « ce bloc doit-il être dupliqué chez *ce* contributeur particulier? » Un exemple pour ce prédicat serait de ne répondre par l'affirmative que si le contributeur en question est connu.

L'utilisateur a la possibilité de passer à son démon une liste d'adresses IP de contributeurs connus. Le démon essaiera alors de les contacter périodiquement pour y dupliquer ses données. Le démon de sauvegarde peut aussi *découvrir* automatiquement les autres démons. Nous avons mis en œuvre cette fonctionnalité au moyen de la bibliothèque Avahi [Poettering 2006] qui met en œuvre le *protocole de découverte de services* DNS-SD/mDNS [Cheshire & Krochmal 2006a, Cheshire & Krochmal 2006b]. Bien que n'ayant pas été conçu explicitement pour les réseaux sans fils *ad hoc*, ce protocole offre un solution complètement décentralisée.

protocole de découverte de services

Une fois un contributeur découvert, le démon lui envoie (au moyen de requêtes `put`, cf. section 4.1) les blocs satisfaisant les prédicats de réplication fournis par l'utilisateur et met à jour la file de blocs. En revanche, nous n'avons pour le moment pas mis en œuvre la duplication de blocs en provenance de propriétaires vers d'autres contributeurs, et en particulier vers un serveur de stockage sur Internet (section 2.3).

6.3.3. Restauration des données

La restauration des données se fait en envoyant des requêtes `get` aux contributeurs accessibles. Plus précisément, un outil client effectue une requête `get` au démon de sauvegarde qui, à son tour, y répond ou la fait suivre vers les contributeurs accessibles. Lorsqu'un des contributeurs répond avec le bloc demandé, il l'ajoute dans un cache local. Pour améliorer les performances, le démon de sauvegarde garde également un cache des sessions TLS avec les contributeurs qui ont correctement servi des requêtes `get`.

6.3.4. Contribution d'espace de stockage

Les démons de stockage attendent également les connexions entrantes, et servent les requêtes `put` et `get`. L'utilisateur peut fournir un prédicat indiquant si une connexion doit être acceptée ou non, en fonction de l'identifiant du propriétaire demandeur. Cette possibilité permet de mettre en œuvre des politiques de coopération telles que discutées au chapitre 5.

Si la connexion est acceptée, le démon ouvre l'espace de stockage des blocs correspondant à ce propriétaire et s'en sert pour répondre à ses requêtes.

6.4. Mesures expérimentales

Afin d'avoir une première évaluation de notre prototype, nous avons conduit deux expériences. La première consiste à demander à un démon de sauvegarder un certain nombre de fichiers et d'envoyer les blocs correspondants à un autre démon situé sur la même machine. Nous avons mesuré séparément d'une part le temps nécessaire pour découper les fichiers et ajouter les blocs à la file, et d'autre part le temps nécessaire pour envoyer les blocs. Nos mesures ont révélé que cette deuxième étape était la plus longue.

Au final, nous obtenons un débit un ordre de grandeur plus faible qu'une simple copie au moyen de SSH. Nous envisageons principalement une raison à cette différence. Contrairement à SSH, notre démon effectue la copie en deux étapes, les blocs étant d'abord tous insérés dans une liste de blocs (en pratique, une « base de données » TDB [Tridgell *et al.* 1999]), laquelle est ensuite parcourue au moment où les blocs sont envoyés aux contributeurs. Cette approche a un coût qui dépend grandement de la base de données où sont stockés les blocs (TDB en l'occurrence).

La deuxième expérience consiste à répartir les blocs correspondant à un ensemble de fichiers sur différents contributeurs. Nous vérifions ensuite si un démon (le propriétaire supposé des données) est bel et bien capable de restaurer tous les fichiers. Cette opération nous a permis de vérifier le bon fonctionnement du mécanisme de recouvrement. Les débits observés sont toutefois faibles (de deux ordres de grandeur inférieurs à une simple copie avec SSH), ce qui nous a permis d'identifier des goulots d'étranglement probables dans l'implémentation et le protocole de recouvrement. En particulier, il apparaît clair que faire une requête `get` pour chaque bloc à récupérer est trop coûteux car cela entraîne un nombre d'aller-retours trop important entre le démon chargé de la restauration et les contributeurs. Ce problème pourrait être résolu en introduisant une requête `mget` (*multiple get*) permettant de récupérer plusieurs blocs d'un coup.

6.5. Résumé

Dans ce chapitre, nous avons abordé les points suivants :

- les résultats et propositions des chapitres précédents ont été intégrés dans un prototype du service de sauvegarde;
- des structures de données complétant les fonctionnalités de la couche de stockage du chapitre 4 ont été décrites;
- les algorithmes utilisés pour la réplication opportuniste, la restauration de données, et la contribution d'espace de stockage ont été développés;
- nous avons abordé les possibilités de paramétrisation des algorithmes;

- des résultats expérimentaux préliminaires donnent un aperçu des performances de notre prototype, et nous ont permis d'identifier les aspects de l'implémentation ou du protocole à améliorer.

Le résultat de cet effort est un outil de sauvegarde coopérative flexible.

Chapitre 7. Conclusions et perspectives

La sûreté de fonctionnement des dispositifs informatiques mobiles devient un sujet de plus en plus pressant. De nombreux utilisateurs dépendent de ce type d'outil pour une variété d'applications toujours plus grande, créant des quantités de données importantes. Malgré tout, peu de mécanismes de tolérance aux fautes sont disponibles pour garantir la disponibilité de ces données. Les mécanismes existant sont souvent contraignants, nécessitant au moins un accès régulier à une infrastructure réseau. Une conséquence est que les données produites sur ces dispositifs risquent d'être perdues.

Dans cette thèse, nous nous sommes proposés de résoudre ce problème en concevant un *service de sauvegarde coopérative* pour dispositifs mobiles tirant parti des moyens de communications sans fils spontanés et des rencontres d'autres dispositifs. Nous avons cherché à montrer qu'une telle approche pouvait améliorer la disponibilité des données stockées sur des dispositifs mobiles.

Les contributions de cette thèse couvrent plusieurs aspects de la conception et mise en œuvre d'un tel service. Nos objectifs, motivations et notre approche ont été ébauchés au chapitre 2. Nous avons ensuite proposé au chapitre 3 une évaluation analytique de l'amélioration de la disponibilité des données apportée par notre service à l'aide d'une modélisation basée sur les réseaux de Petri et les chaînes de Markov. Cette étude nous a permis de quantifier l'amélioration apportée par le service de sauvegarde coopératif, d'identifier les scénarios où il est bénéfique, et de comparer différentes stratégies de réplication.

Le chapitre 4 a ensuite abordé le choix de techniques de stockage adaptées au service. Nous avons décrit notre mise en œuvre de la couche de stockage du service de sauvegarde. Nous avons présenté les résultats d'une expérimentation visant à comparer différentes combinaisons des techniques étudiées. Le chapitre suivant a considéré les menaces pesant contre la coopération entre des participants ne se faisant pas confiance. Nous avons rappelé les attaques en déni de service auquel est sujet notre service, et proposé des primitives permettant l'imputabilité des actes des participants. Nous avons montré que différentes politiques de coopération pouvaient être mises en œuvre au-dessus de ces mécanismes. Enfin, le chapitre 6 a présenté notre prototype qui se base sur nos résultats et propositions. Il a montré comment intégrer ces différentes contributions. En outre, il a présenté la mise en œuvre des « pièces manquantes » et présenté les algorithmes utilisés pour la réplication opportuniste des données et leur restauration.

Parmi les perspectives, nous envisageons une évaluation plus approfondie de notre prototype et de ses composants. Il serait en particulier intéressant de mettre en œuvre dif-

férentes stratégies de réplication et différentes politiques de coopération et d'étudier leur impact sur la disponibilité des données d'une part, et l'utilisation des ressources partagées de l'autre. Certaines fonctionnalités manquent encore à notre prototype et demandent à être ajoutées. Les limites de ses performances ayant été identifiées, nous aimerions bien sûr y remédier. Enfin, nous pensons qu'un déploiement effectif du logiciel sur des dispositifs mobiles pourraient nous donner un complément d'information intéressant.

Bibliographie

[Aad *et al.* 2006]
I. Aad, C. Castelluccia, J-P. Hubaux. Packet Coding for Strong Anonymity in Ad Hoc Networks. In *Proc. of the nternational Conf. on Security and Privacy in Communication Networks (Securecomm)*, pp. 1–10, August 2006.

[Aiyer *et al.* 2005]
A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J-P. Martin, C. Porth. BAR Fault Tolerance for Cooperative Services. In *Proc. of the ACM Symp. on Operating Systems Principles*, pp. 45–58, October 2005.

[Avizienis *et al.* 2004]
A. Avizienis, J-C. Laprie, B. Randell, C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. In *IEEE Transactions on Dependable and Secure Computing*, (1)pp. 11–33, IEEE CS Press, 2004.

[Barr & Asanovic 2006]
K. Barr, K. Asanovic. Energy Aware Lossless Data Compression. In *ACM Transactions on Computer Systems*, 24(3) , August 2006, pp. 250–291.

[Barreto & Ferreira 2004]
J. Barreto, P. Ferreira. A Replicated File System for Resource Constrained Mobile Devices. In *Proc. of the IADIS Int. Conf. on Applied Computing*, March 2004.

[Batten *et al.* 2001]
C. Batten, K. Barr, A. Saraf, S. Treptin. pStore : A Secure Peer-to-Peer Backup System. Technical Report MIT-LCS-TM-632, MIT Laboratory for Computer Science, December 2001.

[Bennett *et al.* 2002]
K. Bennett, C. Grothoff, T. Horozov, I. Patrascu. Efficient Sharing of Encrypted Data. In *Proc. of the 7th Australasian Conf. on Information Security and Privacy (ACISP 2002)*, Lecture Notes in Computer Science, (2384)pp. 107–120, Springer-Verlag, 2002.

[Bicket *et al.* 2005]
J. Bicket, D. Aguayo, S. Biswas, R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *Proc. of the Int. Conf. on Mobile Computing and Networking (MobiCom)*, pp. 31–42, ACM Press, 2005.

[Black 2006]
J. Black. Compare-by-Hash : A Reasoned Analysis. In *Proc. of the USENIX Annual Technical Conf., Systems and Experience Track*, 2006.

[Bolosky *et al.* 2000]
W. J. Bolosky, J. R. Douceur, D. Ely, M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. In *Proc. of the Int. Conf. on Measurement and Modeling of Computer Systems*, pp. 34–43, 2000.

[Boulkenafed & Issarny 2003]
M. Boulkenafed, V. Issarny. AdHocFS : Sharing Files in WLANs. In *Proc. of the 2nd Int. Symp. on Network Computing and Applications*, April 2003.

[Buchegger & Boudec 2003]
S. Buchegger, J-Y. L. Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks. In *Proc. of WiOpt '03 : Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, March 2003.

[Buttyán & Hubaux 2000]
L. Buttyán, J-P. Hubaux. Enforcing Service Availability in Mobile Ad-Hoc WANs. In *Proc. of the 1st ACM Int. Symp. on Mobile Ad Hoc Networking & Computing*, pp. 87–96, IEEE CS Press, 2000.

[Buttyán & Hubaux 2003]
L. Buttyán, J-P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. In *ACM/Kluwer Mobile Networks and Applications*, 8(5) , October 2003, pp. 579–592.

[Callas *et al.* 1998]
J. Callas, L. Donnerhacke, H. Finney, R. Thayer. OpenPGP Message Format (RFC 2440). Internet Engineering Task Force (IETF), November 1998. *http://tools.ietf.org/html/rfc2440.*

[Capkun *et al.* 2002]
S. Capkun, L. Buttyán, J-P. Hubaux. Small Worlds in Security Systems : an Analysis of the PGP Certificate Graph. In *Proc. of the Workshop on New Security Paradigms*, pp. 28–35, ACM Press, 2002.

[Capkun *et al.* 2003]
S. Capkun, L. Buttyán, J-P. Hubaux. Self-Organized Public-Key Management for Mobile Ad Hoc Networks. In *IEEE Transactions on Mobile Computing*, 2(1) , January 2003, pp. 52–64.

[Cerf *et al.* 2007]
V. G. Cerf, S. C. Burleigh, R. C. Durst, K. Fall, A. J. Hooke, K. L. Scott, L. Torgerson, H. S. Weiss. Delay-Tolerant Networking Architecture (RFC 4838). Google Corporation, VA, USA, April 2007. *http://tools.ietf.org/html/rfc4838.*

[Cheshire & Krochmal 2006a]
S. Cheshire, M. Krochmal. DNS-Based Service Discovery. Apple Computer, Inc., August 2006. *http://www.dns-sd.org/*.

[Cheshire & Krochmal 2006b]
S. Cheshire, M. Krochmal. Multicast DNS. Apple Computer, Inc., August 2006. *http://www.multicastdns.org/*.

[Clarke *et al.* 2001]
I. Clarke, O. Sandberg, B. Wiley, T. W. Hong. Freenet : A Distributed Anonymous Information Storage and Retrieval System. In *Proc. of the Int. Workshop on Designing Privacy Enhancing Technologies*, pp. 46–66, Springer-Verlag, 2001.

[Clausen & Jacquet 2003]
T. H. Clausen, P. Jacquet. Optimized Link State Routing Protocol (RFC 3626). INRIA Rocquencourt, France, October 2003. *http://tools.ietf.org/html/rfc3626*.

[Cornell *et al.* 2004]
B. Cornell, P. Dinda, F. Bustamante. Wayback : A User-level Versioning File System for Linux. In *Proc. of the USENIX Annual Technical Conf., FREENIX Track*, pp. 19–28, 2004.

[Courtès *et al.* 2006]
L. Courtès, M-O. Killijian, D. Powell. Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices. In *Proc. of the 6th European Dependable Computing Conf.*, pp. 129–138, IEEE CS Press, October 2006.

[Courtès *et al.* 2007a]
L. Courtès, O. Hamouda, M. Kaâniche, M-O. Killijian, D. Powell. Dependability Evaluation of Cooperative Backup Strategies for Mobile Devices. In *Proc. of the IEEE Int. Symp. on Pacific Rim Dependable Computing*, December 2007.

[Courtès *et al.* 2007b]
L. Courtès, M-O. Killijian, D. Powell. Security Rationale for a Cooperative Backup Service for Mobile Devices. In *Proc. of the Latin-American Symp. on Dependable Computing*, pp. 212–230, Springer-Verlag, September 2007.

[Cox *et al.* 2002]
L. P. Cox, C. D. Murray, B. D. Noble. Pastiche : Making Backup Cheap and Easy. In *5th USENIX Symp. on Operating Systems Design and Implementation*, pp. 285–298, December 2002.

[Cox & Noble 2003]
L. P. Cox, B. D. Noble. Samsara : Honor Among Thieves in Peer-to-Peer Storage. In *Proc. 19th ACM Symp. on Operating Systems Principles*, pp. 120–132, October 2003.

[Dabek *et al.* 2001]
F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica. Wide-Area Cooperative Storage With CFS. In *Proc. 18th ACM Symp. on Operating Systems Principles*, pp. 202–215, October 2001.

[Demers *et al.* 1994]
A. Demers, K. Petersen, M. Spreitzer, D. Terry, M. Theimer, B. Welch. The Bayou Architecture : Support for Data Sharing Among Mobile Users. In *Proc. of the Workshop on Mobile Computing Systems and Applications*, pp. 2–7, IEEE CS Press, December 1994.

[Deswarte *et al.* 1991]

Y. Deswarte, L. Blain, J-C. Fabre. Intrusion Tolerance in Distributed Computing Systems. In *Proc. of the IEEE Symp. on Research in Security and Privacy*, pp. 110–121, May 1991.

[Deutsch & Gailly 1996]

P. Deutsch, J-L. Gailly. ZLIB Compressed Data Format Specification Version 3.3 (RFC 1950). Internet Engineering Task Force (IETF), May 1996. *http://tools.ietf.org/html/rfc1950.html.*

[Dierks *et al.* 2006]

T. Dierks, E. Rescorla, W. Teerse. The Transport Layer Security (TLS) Protocol, Version 1.1 (RFC 4346). Internet Engineering Task Force (IETF), April 2006. *http://tools.ietf.org/html/rfc4346.*

[Dingledine *et al.* 2001]

R. Dingledine, M. J. Freedman, D. Molnar. Accountability. In *Peer-to-Peer : Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Andy Oram (editor), Chapter 16, pp. 271–341, March 2001.

[Douceur 2002]

J. R. Douceur. The Sybil Attack. In *Revised Papers from the 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pp. 251–260, Springer-Verlag, 2002.

[Ellison 1996]

C. M. Ellison. Establishing Identity Without Certification Authorities. In *Proc. of the 6th USENIX Security Symp.*, pp. 67–76, 1996.

[Ellison *et al.* 1999]

C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. SPKI Certificate Theory (RFC 2693). Internet Engineering Task Force (IETF), September 1999. *http://www.ietf.org/rfc/rfc2693.txt*.

[Fall 2003]

K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 27–34, August 2003.

[Farrell & Cahill 2006]

S. Farrell, V. Cahill. Security Considerations in Space and Delay Tolerant Networks. In *Proc. of the 2nd IEEE Int. Conf. on Space Mission Challenges for Information Technology*, pp. 29–38, IEEE CS Press, 2006.

[Feeney & Nilsson 2001]

L. M. Feeney, M. Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *Proc. of the 20th IEEE Conf. on Computer Communications (IEEE InfoCom)*, pp. 1548–1557, April 2001.

[Flinn *et al.* 2003]
J. Flinn, S. Sinnamohideen, N. Tolia, M. Satyanarayanan. Data Staging on Untrusted Surrogates. In *Proc. of the USENIX Conf. on File and Storage Technologies (FAST)*, March 2003.

[Gibson & Miller 1998]
T. J. Gibson, E. L. Miller. Long-Term File Activity Patterns in a UNIX Workstation Environment. In *Proc. of the 15th IEEE Symp. on Mass Storage Systems*, pp. 355–372, March 1998.

[Goland *et al.* 1999]
Y. Y. Goland, T. Cai, P. Leach, Y. Gu, S. Albright. Simple Service Discovery Protocol/1.0 Operating without an Arbiter. Internet Engineering Task Force (IETF), October 1999. *http://quimby.gnus.org/internet-drafts/draft-cai-ssdp-v1-03.txt*.

[Goldberg & Yianilos 1998]
A. V. Goldberg, P. N. Yianilos. Towards an Archival Intermemory. In *Proc. IEEE Int. Forum on Research and Technology Advances in Digital Libraries (ADL'98)*, pp. 147–156, IEEE Society, April 1998.

[Gray 1981]
J. Gray. The Transaction Concept : Virtues and Limitations. In *Proc. of the Int. Conf. on Very Large Data Bases*, pp. 144–154, IEEE CS Press, September 1981.

[Grothoff 2003]
C. Grothoff. An Excess-Based Economic Model for Resource Allocation in Peer-to-Peer Networks. In *Wirtschaftsinformatik*, 45(3) , June 2003, pp. 285–292.

[Guttman *et al.* 1999]
E. Guttman, C. Perkins, J. Veizades, M. Day. RFC 2608 -- Service Location Protocol, Version 2. Internet Engineering Task Force (IETF), June 1999. *http://tools.ietf.org/html/rfc2608.*

[Hamano 2006]
J. C. Hamano. GIT—A Stupid Content Tracker. In *Proc. of the Linux Symp., Volume One*, pp. 385–393, July 2006.

[Hamouda 2006]
O. Hamouda. Évaluation de la sûreté de fonctionnement d'un système de sauvegarde coopérative pour des dispositifs mobiles (Master de recherche). LAAS-CNRS, Toulouse, France, 2006.

[Harras *et al.* 2007]
K. A. Harras, M. P. Wittie, K. C. Almeroth, E. M. Belding. ParaNets : A Parallel Network Architecture for Challenged Networks. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, February 2007.

[Helmy 2004]
A. Helmy. Efficient Resource Discovery in Wireless Ad Hoc Networks : Contacts Do Help. In *Resource Management in Wireless Networking*, Kluwer Academic Publishers, May 2004.

[Hitz *et al.* 1994]
D. Hitz, J. Lau, M. Malcolm. File System Design for an NFS File Server Appliance. In *Proc. of the USENIX Winter Technical Conf.*, The USENIX Association, January 1994.

[IEEE-SA Standards Board 1999]
IEEE-SA Standards Board. ANSI/IEEE Std 802.11---Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 1999. *http://standards.ieee.org/*.

[Jaffer *et al.* 1996]
A. Jaffer, T. Lord, J. Blandy, M. Vollmer, M. Djurfeldt. GNU Guile : GNU's Ubiquitous Intelligent Language for Extension. GNU Project, 1996. *http://www.gnu.org/software/guile/*.

[Jain & Agrawal 2003]
S. Jain, D. P. Agrawal. Wireless Community Networks. In *IEEE Computer*, 36(8) , August 2003, pp. 90–92.

[Jain *et al.* 2005]
S. Jain, M. Demmer, R. Patra, K. Fall. Using Redundancy to Cope with Failures in a Delay Tolerant Network. In *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 109–120, ACM Press, 2005.

[Juang *et al.* 2002]
P. Juang, H. Oki, Y. Wang, M. Martonosi, L-S. Peh, D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking : Design Tradeoffs and Early Experiences with ZebraNet. In *ACM SIGOPS Operating Systems Review*, 36, 2002, pp. 96–107.

[Karypidis & Lalis 2006]
A. Karypidis, S. Lalis. OmniStore : A System for Ubiquitous Personal Storage Management. In *Proc. of the Annual IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, pp. 136–147, IEEE CS Press, March 2006.

[Kelsey *et al.* 1998]
R. Kelsey, W. Clinger, J. Rees. Revised5 Report on the Algorithmic Language Scheme. In *Higher-Order and Symbolic Computation*, 11(1) , August 1998, pp. 7–105.

[Kemeny & Snell 1960]
J. G. Kemeny, J. L. Snell. Finite Markov Chains. D. Van Nostrand Co., Inc., Princeton, New Jersey, USA, 1960.

[Kim *et al.* 2005]
M. J. Kim, M. Kumar, B. A. Shirazi. Service Discovery using Volunteer Nodes for Pervasive Environments. In *Proc. of the Int. Conf. on Pervasive Services (ICPS)*, pp. 188–197, July 2005.

[Kubiatowicz *et al.* 2000]
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. In *Proc. of the 9th Int. Conf. on Architectural Support*

for Programming Languages and Operating Systems (ASPLOS 2000), pp. 190–201, November 2000.

[Kulkarni *et al.* 2004]

P. Kulkarni, F. Douglis, J. LaVoie, J. M. Tracey. Redundancy Elimination Within Large Collections of Files. In *Proc. of the USENIX Annual Technical Conf.*, 2004.

[Kügler 2003]

D. Kügler. An Analysis of GNUnet and the Implications for Anonymous, Censorship-Resistant Networks. In *Proc. of the Conf. on Privacy Enhancing Technologies*, Lecture Notes in Computer Science, pp. 161–176, Springer, March 2003.

[Lai *et al.* 2003]

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

[Landers *et al.* 2004]

M. Landers, H. Zhang, K-L. Tan. PeerStore : Better Performance by Relaxing in Peer-to-Peer Backup. In *Proc. of the 4th Int. Conf. on Peer-to-Peer Computing*, pp. 72–79, August 2004.

[Lee *et al.* 1999]

Y-W. Lee, K-S. Leung, M. Satyanarayanan. Operation-based Update Propagation in a Mobile File System. In *Proc. of the USENIX Annual Technical Conf.*, pp. 43–56, June 1999.

[Liao *et al.* 2006]

Y. Liao, K. Tan, Z. Zhang, L. Gao. Estimation Based Erasure Coding Routing in Delay Tolerant Networks. In *Proc. of the Int. Conf. on Communications and Mobile Computing*, pp. 557–562, ACM Press, 2006.

[Liedtke 1993]

J. Liedtke. A Persistent System in Real Use — Experiences of the First 13 Years. In *Proc. of the Int. Workshop on Object-Orientation in Operating Systems (I-WOOOS)*, pp. 2–11, December 1993.

[Lillibridge *et al.* 2003]

M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, M. Isard. A Cooperative Internet Backup Scheme. In *Proc. of the USENIX Annual Technical Conf.*, pp. 29–42, June 2003.

[Lin *et al.* 2004]

W. K. Lin, D. M. Chiu, Y. B. Lee. Erasure Code Replication Revisited. In *Proc. of the 4th Int. Conf. on Peer-to-Peer Computing*, pp. 90–97, 2004.

[Loo *et al.* 2003]

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

[Manber 1994]

U. Manber. Finding Similar Files in a Large File System. In *Proc. of the USENIX Winter 1994 Conf.*, pp. 1–10, January 1994.

[Marti & Garcia-Molina 2003]
S. Marti, H. Garcia-Molina. Identity Crisis : Anonymity vs. Reputation in P2P Systems. In *IEEE Conf. on Peer-to-Peer Computing*, pp. 134–141, IEEE CS Press, September 2003.

[Mavrogiannopoulos 2007]
N. Mavrogiannopoulos. Using OpenPGP Keys for Transport Layer Security Authentication (RFC 5081). Internet Engineering Task Force (IETF), November 2007. *http://tools.ietf.org/html/rfc5081.*

[Melski 1999]
E. Melski. Burt : The Backup and Recovery Tool. In *Proc. of the USENIX Conf. on System Administration*, pp. 207–218, USENIX Association, 1999.

[Michiardi & Molva 2002]
P. Michiardi, R. Molva. CORE : A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. In *Proc. of the 6th IFIP TC6/TC11 Joint Conf. on Communications and Multimedia Security*, pp. 107–121, Kluwer Academic Publishers, September 2002.

[Milgram 1967]
S. Milgram. The Small World Problem. In *Psychology Today*, 2, 1967, pp. 60–67.

[Miller 2006]
M. S. Miller. Robust Composition : Towards a Unified Approach to Access Control and Concurrency Control, PhD Thesis, Johns Hopkins University, Baltimore, MA, USA, May 2006.

[Mitzenmacher 2004]
M. Mitzenmacher. Digital Fountains : A Survey and Look Forward. In *Proc. of the IEEE Information Theory Workshop*, pp. 271–276, October 2004.

[Montenegro & Castelluccia 2002]
G. Montenegro, C. Castelluccia. Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses. In *Proc. of the Network and Distributed System Security Symp. (NDSS)*, 2002.

[Muthitacharoen *et al.* 2001]
A. Muthitacharoen, B. Chen, D. Mazières. A Low-Bandwidth Network File System. In *Proc. of the 18th ACM Symp. on Operating Systems Principles*, pp. 174–187, October 2001.

[NESSIE Consortium 2003a]
NESSIE Consortium. NESSIE Security Report. Technical Report NES/DOC/ENS/WP5/D20/2, February 2003.

[NESSIE Consortium 2003b]
NESSIE Consortium. Portfolio of Recommended Cryptographic Primitives. Technical Report , February 2003.

[Nightingale & Flinn 2004]
E. B. Nightingale, J. Flinn. Energy-Efficiency and Storage Flexibility in the Blue File System. In *Proc. of the 6th Symp. on Operating Systems Design and Implementation (OSDI'04)*, December 2004.

[Oberhumer 2005]
M. F.X.J. Oberhumer. LZO Real-Time Data Compression Library. 2005. *http://www.oberhumer.com/opensource/lzo/*.

[One Laptop Per Child Project 2007]
One Laptop Per Child Project. Mesh Networking in OLPC. 2007. *http://wiki.laptop.org/go/Mesh_Network_Details.*

[Open Mobile Alliance 2001]
Open Mobile Alliance. SyncML Sync Protocol, Version 1.0.1. June 2001. *http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html.*

[Oualha *et al.* 2007a]
N. Oualha, P. Michiardi, Y. Roudier. A Game Theoretic Model of a Protocol for Data Possession Verification. In *Proc. of the IEEE Int. Workshop on Trust, Security, and Privacy for Ubiquitous Computing (TSPUC)*, June 2007.

[Oualha *et al.* 2007b]
N. Oualha, S. M. Önen, Y. Roudier. Verifying Self-Organized Storage with Bilinear Pairings. Technical Report Research Report RR-07-201, Eurecom, France, June 2007.

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

[Perkins *et al.* 2003]
C. E. Perkins, E. M. Belding-Royer, S. R. Das. RFC3561 – Ad hoc On-Demand Distance Vector (AODV) Routing. Internet Engineering Task Force, July 2003. *http://www.ietf.org/rfc/rfc3561.txt*.

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

[Poettering 2006]
L. Poettering. The Avahi DNS-SD/mDNS Free Implementation. 2006. *http://avahi.org/*.

[Poettering 2007]
L. Poettering. Diensteverwaltung in Ad-Hoc-Netzwerken (Diplomarbeit). Fakultät für Mathematik, Informatik und Naturwissenschaften, Universität Hamburg, Germany, April 2007. *http://0pointer.de/public/diplom.pdf* .

[Preguica *et al.* 2005]
N. Preguica, C. Baquero, J. L. Martins, M. Shapiro, P. S. Almeida, H. Domingos, V. Fonte, S. Duarte. FEW : File Management for Portable Devices. In *Proc. of the Int. Workshop on Software Support for Portable Storage*, March 2005.

[Prevayler.Org 2006]
Prevayler.Org. Prevayler, a Java Implementation of "Object Prevalence". 2006. *http://www.prevayler.org/*.

[Quinlan & Dorward 2002]
S. Quinlan, S. Dorward. Venti : A New Approach to Archival Storage. In *Proc. of the 1st USENIX Conf. on File and Storage Technologies*, pp. 89–101, 2002.

[Rahman *et al.* 2006]
S. M. M. Rahman, A. Inomata, T. Okamoto, M. Mambo, E. Okamoto. Anonymous Secure Communication in Wireless Mobile Ad-hoc Networks. In *Proc. of the 1st Int. Conf. on Ubiquitous Convergence Technology*, pp. 131–140, Springer-Verlag, December 2006.

[Red Hat, Inc. 2007]
Red Hat, Inc.. Hibernate : Relational Persistence for Java and .NET. 2007. *http://www.hibernate.org/*.

[Rubel 2005]
M. Rubel. Rsnapshot : A Remote Filesystem Snapshot Utility Based on Rsync. 2005. *http://rsnapshot.org/*.

[Sailhan & Issarny 2005]
F. Sailhan, V. Issarny. Scalable Service Discovery for MANET. In *Proc. of the IEEE Int. Conf. on Pervasive Computing and Communication*, March 2005.

[Santry *et al.* 1999]
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. In *Proc. of the 17th ACM Symp. on Operating Systems Principles*, pp. 110–123, December 1999.

[Scott & Burleigh 2007]
K. L. Scott, S. Burleigh. Bundle Protocol Specification (Internet Draft). The MITRE Corporation, VA, USA, April 2007. *http://tools.ietf.org/html/draft-irtf-dtnrg-bundle-spec-09*.

[Seward 2007]
J. Seward. Bzip2 and Libbzip2 Website. 2007. *http://www.bzip.org/*.

[Shapiro & Adams 2002]
J. S. Shapiro, J. Adams. Design Evolution of the EROS Single-Level Store. In *Proc. of the USENIX Annual Technical Conf.*, May 2002.

[Shapiro & Vanderburgh 2002]
J. S. Shapiro, J. Vanderburgh. CPCMS : A Configuration Management System Based on Cryptographic Names. In *Proc. of the USENIX Annual Technical Conf., FREENIX Track*, pp. 207–220, USENIX Association, 2002.

[Sit *et al.* 2003]
E. Sit, J. Cates, R. Cox. A DHT-based Backup System. Technical Report , MIT Laboratory for Computer Science, August 2003.

[Spyropoulos *et al.* 2007]
T. Spyropoulos, K. Psounis, C. Raghavendra. Efficient Routing in Intermittently Connected Mobile Networks : The Single-Copy Case. In *ACM/IEEE Transactions on Networking*, , 2007, .

[Stemm *et al.* 1997]
M. Stemm, P. Gauthier, D. Harada, R. H. Katz. Reducing Power Consumption of Network Interfaces in Hand-Held Devices. In *IEEE Transactions on Communications*, E80-B(8) , August 1997, pp. 1125–1131.

[Sözer *et al.* 2004]
H. Sözer, M. Tekkalmaz, I. Körpeoglu. A Peer-to-Peer File Sharing System for Wireless Ad-Hoc Networks. In *Proc. of the 3rd Annual Mediterranean Ad Hoc Networking Workshop (MedHoc)*, June 2004.

[The UbiSec Project 2005]
The UbiSec Project. Ubiquitous Networks with a Secure Provision of Services, Access, and Content Delivery. 2005. *http://jerry.c-lab.de/ubisec/*.

[Tichy 1985]
W. F. Tichy. RCS—A System for Version Control. In *Software—Practice & Experience*, 15, New York, NY, USA, 1985, pp. 637–654.

[Tolia *et al.* 2004]
N. Tolia, J. Harkes, M. Satyanarayanan. Integrating Portable and Distributed Storage. In *Proc. of the USENIX Conf. on File and Storage Technologies*, pp. 227–238, March 2004.

[Tridgell & Mackerras 1996]
A. Tridgell, P. Mackerras. The Rsync Algorithm. Technical Report TR-CS-96-05, Department of Computer Science, Australian National University Canberra, Australia 1996.

[Tridgell *et al.* 1999]
A. Tridgell, P. Russel, J. Allison. The Trivial Database. 1999. *http://samba.org/*.

[Vernois & Utard 2004]
A. Vernois, G. Utard. Data Durability in Peer to Peer Storage Systems. In *Proc. of the 4th Workshop on Global and Peer to Peer Computing*, pp. 90–97, IEEE CS Press, April 2004.

[Wang *et al.* 2005]
Y. Wang, S. Jain, M. Martonosi, K. Fall. Erasure-Coding Based Routing for Opportunistic Networks. In *Proc. of the ACM SIGCOMM Workshop on Delay-Tolerant Networking*, pp. 229–236, ACM Press, 2005.

[Weatherspoon & Kubiatowicz 2002]
H. Weatherspoon, J. Kubiatowicz. Erasure Coding vs. Replication : A Quantitative Comparison. In *Revised Papers from the 1st Int. Workshop on Peer-to-Peer Systems*, pp. 328–338, Springer-Verlag, 2002.

[Xu *et al.* 1999]
L. Xu, V. Bohossian, J. Bruck, D. G. Wagner. Low Density MDS Codes and Factors of Complete Graphs. In *IEEE Transactions on Information Theory*, 45(1) , November 1999, pp. 1817–1826.

[Xu 2005]
L. Xu. Hydra : A Platform for Survivable and Secure Data Storage Systems. In *Proc. of the ACM Workshop on Storage Security and Survivability*, pp. 108–114, ACM Press, November 2005.

[You & Karamanolis 2004]
L. L. You, C. Karamanolis. Evaluation of Efficient Archival Storage Techniques. In *Proc. of 21st IEEE/12th NASA Goddard Conf. on Mass Storage Systems and Technologies*, pp. 227–232, April 2004.

[You *et al.* 2005]
L. L. You, K. T. Pollack, D. D. E. Long. Deep Store : An Archival Storage System Architecture. In *Proc. of the 21st Int. Conf. on Data Engineering (ICDE 2005)*, pp. 804–815, IEEE CS Press, April 2005.

[Zhang *et al.* 2005]
T. Zhang, S. Madhani, P. Gurung, E. v. d. Berg. Reducing Energy Consumption on Mobile Devices with WiFi Interfaces. In *Proc. of the IEEE Global Telecommunications Conf. (Globecom)*, pp. 561–565, IEEE CS Press, December 2005.

[Zhang 2006]
Z. Zhang. Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks : Overview and Challenges. In *IEEE Communications Surveys & Tutorials*, 8, January 2006, pp. 24–37.

[Zheng & Lee 2004]
J. Zheng, M. J. Lee. Will IEEE 802.15.4 Make Ubiquitous Networking a Reality ? A Discussion on a Potential Low Power, Low Bit Rate Standard. In *IEEE Communications Magazine*, 42, June 2004, pp. 140–146.

[Zöls *et al.* 2005]
S. Zöls, R. Schollmeier, W. Kellerer, A. Tarlano. The Hybrid Chord Protocol : A Peer-to-Peer Lookup Service for Context-Aware Mobile Applications. In *Proc. of the Int. Conf. on Networking*, Lecture Notes in Computer Science, pp. 781–792, Springer Berlin/Heidelberg, 2005.

Sauvegarde coopérative de données pour dispositifs mobiles

Cooperative Data Backup for Mobile Devices

Thèse de Doctorat

Ludovic Courtès

Résumé

Les dispositifs informatiques mobiles tels que les ordinateurs portables, assistants personnels et téléphones portables sont de plus en plus utilisés. Cependant, bien qu'ils soient utilisés dans des contextes où ils sont sujets à des endommagements, à la perte, voire au vol, peu de mécanismes permettent d'éviter la perte des données qui y sont stockées. Dans cette thèse, nous proposons un service de sauvegarde de données coopératif pour répondre à ce problème. Cette approche tire parti de communications spontanées entre de tels dispositifs, chaque dispositif stockant une partie des données des dispositifs rencontrés. Une étude analytique des gains de cette approche en termes de sûreté de fonctionnement est proposée. Nous étudions également des mécanismes de stockage réparti adaptés. Les problèmes de coopération entre individus mutuellement suspicieux sont également abordés. Enfin, nous décrivons notre mise en oeuvre du service de sauvegarde coopérative.

Mots-clef : sûreté de fonctionnement, informatique ubiquiste, sauvegarde de données, systèmes pair-à-pair

Summary

Mobile devices such as laptops, PDAs and cell phones are increasingly relied on but are used in contexts that put them at risk of physical damage, loss or theft. However, few mechanisms are available to reduce the risk of losing the data stored on these devices. In this dissertation, we try to address this concern by designing a cooperative backup service for mobile devices. The service leverages encounters and spontaneous interactions among participating devices, such that each device stores data on behalf of other devices. We first provide an analytical evaluation of the dependability gains of the proposed service. Distributed storage mechanisms are explored and evaluated. Security concerns arising from the cooperation among mutually suspicious principals are identified, and core mechanisms are proposed to allow them to be addressed. Finally, we present our prototype implementation of the cooperative backup service.

Keywords : dependability, ubiquitous computing, data backup, peer-to-peer systems