PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Réputation de composants logiciels dans les réseaux mobiles ad-hoc
Auteur: Author:	Hinnoutondji Kpodjedo
Date:	2007
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Kpodjedo, H. (2007). Réputation de composants logiciels dans les réseaux mobiles ad-hoc [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8053/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8053/
Directeurs de recherche: Advisors:	Samuel Pierre, & Alejandro Quintero
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

RÉPUTATION DE COMPOSANTS LOGICIELS DANS LES RÉSEAUX MOBILES AD-HOC

HINNOUTONDJI KPODJEDO
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2007

© Hinnoutondji Kpodjedo, 2007.

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-35685-2
Our file Notre référence
ISBN: 978-0-494-35685-2

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

RÉPUTATION DE COMPOSANTS LOGICIELS DANS LES RÉSEAUX MOBILES AD-HOC

présenté par : KPODJEDO Hinnoutondji
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. GAGNON Michel, Ph.D., président
M. PIERRE Samuel, Ph.D., membre et directeur de recherche
M. QUINTERO Alejandro, Doct., membre et codirecteur
M. FERNANDEZ José-Manuel, Ph. D, membre

REMERCIEMENTS

Je remercie M. Makan Pourzandi, Ph.D à Ericsson pour sa grande disponibilité et sa productive implication tout au long de ce travail.

Je remercie également mon directeur de recherche, M. Samuel Pierre, professeur à Polytechnique et directeur du LARIM, pour son soutien effectif, ses encouragements tout au long de ce travail et ses précieux conseils pour la rédaction et la correction de ce mémoire.

Je remercie aussi M. Alejandro Quintero mon codirecteur de recherche pour sa disponibilité dans la correction de ce mémoire.

Je remercie enfin les membres du LARIM pour leur collaboration et leur soutien.

RÉSUMÉ

Les réseaux mobiles ad hoc, aussi appelés MANET (Mobile Ad-Hoc Networks), sont restés longtemps un champ de recherche prometteur mais dont les applications concrètes se limitaient jusque là à des expérimentations assez confidentielles (armée, zones sinistrées etc.). La donne devrait changer considérablement avec la mise en œuvre prochaine des VANET (Vehicle Ad-Hoc Networks). Ces réseaux de véhicule à véhicule ouvrent la voie à toute une gamme d'applications devant faciliter la vie des automobilistes en leur permettant d'échanger et de relayer toutes sortes d'informations utiles sur leur environnement. Le succès d'une telle technologie est de toute évidence conditionné par un bon niveau de sécurité et de coopération dans de tels réseaux.

La sécurité et la coopération dans un MANET posent la question de la confiance que peuvent s'accorder les nœuds d'un réseau, par nature, décentralisé. En pointe de cette problématique, on retrouve les modèles de réputation qui se fondent sur l'historique des comportements « sociaux » d'un nœud pour lui créditer un niveau de confiance. De même, l'informatique de confiance, proposée par les acteurs majeurs de l'industrie informatique, est appelée à jouer en la matière un rôle grandissant. En ce sens, les rapports fiables d'intégrité des plates-formes, promis par le TCG (Trusted Computing Group), ouvrent des perspectives tout à fait intéressantes. On pourra désormais faire confiance à un nœud pour rapporter sans tricher sa configuration interne.

L'objectif de ce mémoire est de proposer, comme modèle de confiance dans un MANET, un modèle de réputation basé sur les composants logiciels des configurations transmises par les plates-formes. Plus spécifiquement, il s'agit de dégager des expériences bonnes ou mauvaises d'un nœud des informations permettant à terme d'isoler les éléments logiciels responsables d'éventuels désagréments. En effet, les VANET offrent la particularité d'environnements très grands où les nœuds sont très nombreux, avec moins de chances d'expériences communes entre deux nœuds donnés. L'approche proposée apparaît donc particulièrement pertinente dans de tels scénarios.

Pour évaluer la performance du modèle proposé, des simulations multi-agents ont été réalisées sur différents scénarios, avec en sortie deux indices principaux : le pourcentage de détection des nœuds corrompus et le pourcentage de faux positifs. Il en ressort que le modèle génère des résultats très satisfaisants quant à la détection - totale et rapide - des nœuds corrompus. A la différence d'un modèle de réputation pour les nœuds, le modèle proposé identifie non seulement les nœuds corrompus mais aussi et surtout la cause du mal, procurant de fait un avantage certain. Cet avantage peut se révéler être un inconvénient en cas d'erreur puisque de nombreux nœuds seront d'office concernés. Les résultats obtenus montrent que les erreurs sont rares (autour de 1%) et avec un impact très limité sur l'ensemble des nœuds.

Au final, le modèle proposé ouvre une piste de recherche très prometteuse dans des environnements aussi décentralisés et élargis que sont les VANET.

ABSTRACT

Mobile Ad-Hoc Networks (MANET) had been a very active research field but with just a few applications, essentially in military areas or war-zones. Things may change dramatically with the coming of the VANET (Vehicle Ad-Hoc Networks). Those networks from vehicle to vehicle will introduce a new category of softwares designed to help drivers by allowing them to exchange and forward all kinds of useful information about their common environment. The success of such a technology requires a good level of security and cooperation in the network.

Security and cooperation in a MANET introduce the question of the trust among nodes in a decentralized network. Reputation models, standing among the first-class approaches in that area, propose that the level of trust granted to a given node should be function of its past social behaviors in the network. Trusted computing, as major actors of computer science define it, should also play a growing role in that matter. From this point of view, reliable integrity reporting, as considered by the Trusted Computing Group, might be the start point of very interesting prospects. From now on, a node can be trusted to report without cheating on its own internal configuration.

The goal of this thesis is to propose, as a new trust model for MANET, a reputation model based on software components reported by platforms. More specifically, the model will gain, from good or bad experiences of a node, information leading to the isolation of malicious software components. In VANET environments, nodes are numerous and there is little chance of commune experiences between two given nodes. The proposed approach appears to be particularly pertinent in such scenarios.

To evaluate the performance of the proposed model, multi-agents simulations have been realized to measure the percentage of detection of corrupted nodes and the percentage of false positive errors. While there are very good results for the detection, which is fast and eventually perfect, the percentage of false positives, although very low (around 1%), may not be absolutely satisfying.

TABLE DES MATIÈRES

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vii

TABLE DES MATIÈRES viii

LISTE DES FIGURES x

LISTE DES TABLEAUX xii

SIGLES, ABBREVIATIONS, ACRONYMES xiii

CHAPITRE I : INTRODUCTION 1

1.1 Définitions, concepts de base et contexte de recherche 2

1.1.1 Les réseaux ad hoc 2

1.1.2 La technologie V2V / V2I 2

1.2 Éléments de la problématique 4

1.3 Objectifs de recherche 6

1.4 Plan du mémoire 6

CHAPITRE II : SÉCURITÉ ET COOPÉRATION DANS UN MANET 8

2.1 Problématiques de sécurité dans les MANET 8

2.2 Les modèles de réputation 10

2.2.1 Présentation des systèmes de réputation 10

2.2.2 Systèmes de réputation pour réseaux P2P 11

2.2.3 Les spécificités des MANET 14

2.2.4 Quelques systèmes de réputation pour les MANET 15

2.2.5 Synthèse 19

2.3 L'architecture TCG 20

2.3.1 Principales caractéristiques du TPM 23

2.3.2 Le protocole de rapport d'intégrité 25

2.3.3 TPM : l'essentiel 26

2.4 Configuration logicielle d'un nœud 27
2.4.1 Configuration logicielle complète 27
2.4.2 Configuration logicielle : le minimum requis 30
2.4.3 Configuration logicielle avancée 32
2.5 Conclusion 32

CHAPITRE III : RÉPUTATION DE COMPOSANTS LOGICIELS 34
3.1 Préalable 34
3.1.1 TCG et les réseaux ad-hoc 35
3.1.2 TCG et les modèles de réputation 35
3.1.3 La proposition 38
3.2 Hypothèses du modèle 39
Évaluation de l'interaction 39
3.3 Architecture générale 40
3.3.1 Vue d'ensemble 40
3.3.2 Base de données de réputation 43
3.3.3 Module Reporter 48
3.3.4 Module Discovery 48
3.3.5 Module Réputation 49
3.4 Algorithmes 57

CHAPITRE IV : SIMULATION ET RESULTATS 59
4.1 Implémentation 59
4.2 Plan d'expériences 65
4.3 Résultats 73

CHAPITRE V : CONCLUSION 92
5.1 Synthèse du travail accompli 92
5.2 Limites de l'approche proposée 94
5.3 Travaux futurs 95

BIBLIOGRAPHIE 96

LISTE DES FIGURES

Figure 1.1 Scénarios de communication 3
Figure 2.1 DSR : Construction d'une route sur demande 15
Figure 2.2 Hardware Rooted Trust et couches d'abstraction 21
Figure 2.3 Hardware Rooted Trust et Secure Boot avec un TPM 21
Figure 2.4 Plate-forme de confiance (TBB+TPM) 22
Figure 2.5 Architecture interne d'un TPM 22
Figure 2.6 Diagramme des relations entre crédentiels 24
Figure 2.7 Protocole de rapport d'intégrité 26
Figure 2.8 Boot Aggregate dans une liste de mesure d'intégrité 28
Figure 2.9 Rapport des mesures d'intégrité dans le détail 29
Figure 2.10 ProcessExplorer : Fichiers chargés par Eclipse 30
Figure 3.1 Architecture logicielle au niveau de chaque nœud 42
Figure 3.2 Graphe de stockage des composants logiciels rencontrés 44
Figure 3.3 Mécanisme de « Fading » proposé 47
Figure 3.4 Distribution des évaluations et composants logiciels suspects 52
Figure 4.1 Le terrain et les nœuds mobiles 60
Figure 4.2 Construction de la base de données des logiciels légitimes 61
Figure 4.3 Composants logiciels 62
Figure 4.4 Roulettes biaisées pour les répartitions 63
Figure 4.5 RECON : REputation des CONfigurations 67
Figure 4.6 Évolution du C-FN selon le nombre de nœuds 74
Figure 4.7 Évolution du C-FP selon le nombre de nœuds 75
Figure 4.8 Évolution du C-FN selon le pourcentage de nœuds infectés 76
Figure 4.9 Évolution du C-FP selon le pourcentage de nœuds infectés 77
Figure 4.10 Évolution du C-FN selon le I-FN 78
Figure 4.11 Évolution du C-FP selon le I-FN 79
Figure 4.12 Évolution du C-FN selon le I-FP 80
Figure 4.13 Évolution du C-FP selon le I-FP 80

Figure 4.14 Évolution du C-FN selon la vitesse des nœuds 82
Figure 4.15 Évolution du C-FP selon la vitesse des nœuds 82
Figure 4.16 C-FN et C-FP des 75 nœuds sains (Interactions 0-500) 84
Figure 4.17 C-FN et C-FP des 75 nœuds sains (Interactions 1000-1500) 84
Figure 4.18 C-FN et C-FP des 75 nœuds sains (Interactions 2000-2500) 85
Figure 4.19 C-FN et C-FP des 75 nœuds sains (Interactions 4500-5000) 85
Figure 4.20 Vitesse de détection des composants malicieux pour un nœud 86
Figure 4.21 Comparaison avec SORI (C-FN) 88
Figure 4.22 Comparaison avec SORI (C-FP) 88
Figure 4.23 Session 4x : Comparaison avec SORI (C-FN) 90
Figure 4.24 Session 4x : Comparaison avec SORI (C-FP) 90

LISTE DES TABLEAUX

Tableau 2.1 Récapitulatif des crédentiels TPM 25

Tableau 3.1 Illustration d'une configuration logicielle 43

Tableau 4.1 Facteurs primaires et niveaux associés : 70

Tableau 4.2 Sessions du plan d'expérimentation 72

SIGLES, ABBREVIATIONS, ACRONYMES

CA	Certificate Authority
C-FN	Configuration - Faux Négatif (a priori)
C-FP	Configuration - Faux Positif (a priori)
I-FN	Interaction - Faux Négatif (a posteriori)
I-FP	Interaction - Faux Positif (a posteriori)
MANET	Mobile Ad Hoc Network
TCG	Trusted Computing Group
TPM	Trusted Platform Modules
VANET	Vehicle Ad Hoc Network
V2V	Vehicule to Vehicule
V2I	Vehicule to Infrastructure

CHAPITRE I

INTRODUCTION

La confiance est une notion fondamentale dans toute relation impliquant une ou plusieurs entités. Des sociétés humaines aux environnements électroniques, la confiance conditionne toutes sortes d'interactions. Toutefois, comme chacun est censé le savoir, « la confiance n'exclut pas le contrôle ». En vérité, la « bonne » confiance exige du contrôle. Dans les environnements informatiques, la confiance ne peut s'établir sans un minimum de mécanismes de sécurité. En ce sens, les *réseaux ad hoc* posent, du fait de leur nature intrinsèquement décentralisée, des défis plus élevés. Aux transactions entre pairs s'ajoute la question toute spécifique aux réseaux ad hoc de la coopération des nœuds au routage de l'information. Il existe dans la littérature de nombreux *modèles de confiance* se proposant de répondre à ces diverses problématiques. Les modèles de réputation s'inscrivent ici en pointe de ces démarches et se fondent sur l'historique des comportements « sociaux » d'un nœud pour lui créditer un niveau de confiance.

Parallèlement à cela, se développe, depuis de nombreuses années, le concept d'« informatique de confiance ». Dans ce cadre, le *TCG* (Trusted Computing Group) propose depuis peu un nouveau standard : les *TPM* (Trusted Platform Modules) et, au travers de cette puce, de la confiance enracinée dans le matériel. TCG propose ainsi un support robuste des concepts d'identité et d'intégrité des nœuds interagissant dans un réseau, garantissant un rapport fiable des mesures d'intégrité de chaque plate-forme. Une pleine intégration de ces mesures d'intégrité dans la détermination des liens de confiance entre nœuds pourrait rehausser considérablement le niveau de sécurité et de collaboration. C'est ce qui fait l'objet de notre mémoire. Ce chapitre d'introduction explicite les concepts de base ainsi que le contexte de recherche, expose les éléments de la problématique, précise les objectifs de recherche et esquisse le plan du mémoire.

1.1 Définitions, concepts de base et contexte de recherche

Quelques concepts clés sont essentiels pour situer notre travail de recherche. Nous présenterons ici les réseaux ad hoc et la technologie dérivée qui a initié notre réflexion.

1.1.1 Les réseaux ad hoc

Les réseaux ad hoc sont des réseaux sans fil capables de s'organiser sans infrastructure préalablement définie. Dans leur configuration mobile, ils sont connus sous le nom de MANET (pour Mobile Ad-hoc NETwork). Dans un MANET, on peut donc souligner les points suivants :

- aucune administration centralisée n'est requise ;
- la topologie du réseau peut changer rapidement, de façon aléatoire et non prédictible du fait de l'arrivée / départ d'un nœud mobile à tout instant ;
- tous les terminaux mobiles peuvent remplir des fonctions de routage et la portée de communication est limitée ;
- la durée de vie des liaisons est limitée dans le temps, du fait de la mobilité et de la consommation d'énergie. Les plates-formes mobiles impliquent généralement des contraintes de ressources et de calculs.

Les réseaux sans fil sont par nature plus sensibles aux problèmes de sécurité. Les réseaux ad hoc sont des réseaux pair à pair où tous les nœuds sont équivalents et potentiellement nécessaires au fonctionnement du réseau. Les possibilités de s'insérer dans le réseau sont plus grandes et l'absence de centralisation pose un problème de remontée de l'information.

1.1.2 La technologie V2V / V2I

De nos jours, il n'est plus rare de trouver des voitures équipées de capteurs qui leur fournissent toutes sortes d'informations sur leur environnement immédiat. Ainsi, un conducteur pourra être averti du non-respect de la distance de sécurité par rapport à la voiture qui est devant lui, d'un éventuel « nid de poule », etc.

V2V / V2I [16] Vehicule to Vehicule / Vehicule to Infrastructure est une technologie en cours d'élaboration (WAVE, IEEE 1609 [17]) qui permet à des véhicules d'un réseau routier de collaborer fortement en faisant circuler ces informations utiles sur leur environnement commun. Chaque nœud communique des informations personnelles (position, vitesse et direction) mais aussi sa perception de l'environnement (brouillard, accidents etc.). Les informations sont directement échangées entre nœuds (V2V) ou acheminées vers des infrastructures (V2I) qui se chargent d'en faire la synthèse et d'en tirer les conclusions qui s'imposent. Le déploiement et le coût (autant financier qu'en temps de traitement) des infrastructures justifient que l'on s'attache à rendre le V2V le plus autonome possible.

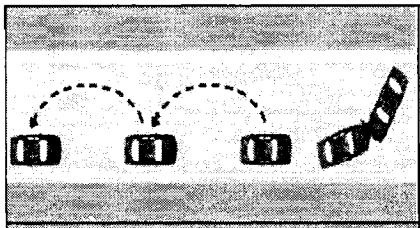
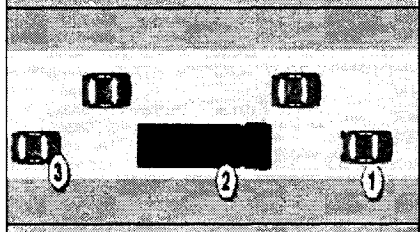
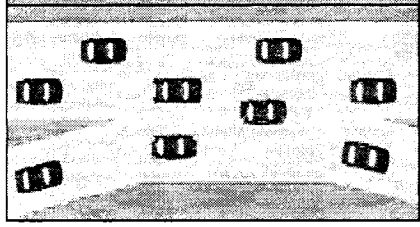
	**Fonctions d'information et d'alertes** Les véhicules transmettent des messages d'alerte quand une situation critique (collision de véhicules, trafic très dense, conditions de routes dangereuses comme du verglas, etc.) est observée.
	**Contrôle longitudinal basé sur la communication** Les véhicules peuvent anticiper des manœuvres de freinage. ex : un véhicule (n. 1) invisible pour un autre (n. 3) freine.
	**Manœuvres de conduite collaborative** En échangeant des informations allant jusqu'à des plans simples de trajectoire, les véhicules peuvent prévoir et résoudre d'eux-mêmes des situations critiques

Figure 1.1 Scénarios de communication

Le cadre de réflexion est donc celui d'un réseau de nœuds très mobiles qui s'échangent de l'information. Dans un réseau routier, les nœuds - les voitures - sont très mobiles et potentiellement très nombreux. Les voitures sont perçues comme étant de nœuds multifonctionnels gérant, entre autres, leurs capteurs. Elles forment ainsi un

« réseau de capteurs » autonomes et intelligents. Les capteurs sont donc des composantes de nœuds plus évolués qui utilisent les informations échangées dans leurs prises de décisions. La Figure 1.1 donne un aperçu des scénarios envisagés, scénarios qui illustrent l'utilité d'une telle technologie. Les perspectives sont extrêmement prometteuses mais une telle technologie demande à l'évidence un niveau certain de fiabilité et de sécurité.

1.2 Éléments de la problématique

Le V2V, aussi appelé VANET (pour Vehicle Ad hoc NETworks) est un type particulier de MANET. Dans un environnement aussi ouvert que celui évoqué, la coopération effective entre nœuds du réseau ne peut être tenue pour acquise. L'éventail des problèmes est assez large, du nœud défaillant (en panne ou mal configuré) au nœud malfaisant (cherchant délibérément à nuire), sans oublier le nœud égoïste (voulant se soustraire à ses obligations envers la communauté des nœuds). Ce dernier point fait la spécificité des réseaux pair à pair. Il n'est pas acquis que toutes les voitures coopèrent de bon gré au routage de l'information. Les nœuds mobiles ont la particularité d'avoir généralement des réserves d'énergie limitées. Dans le cas des voitures, tout conducteur avisé est plutôt attentif à ménager ses batteries. La pleine participation à un tel réseau peut impliquer un rythme soutenu dans le relais des paquets et entrainer des consommations non négligeables d'énergie. Il n'est pas exclu que s'opèrent alors, plus ou moins marginalement, des bidouillages, d'un genre nouveau, destinés à la mise en œuvre de comportements égoïstes.

Le problème ici est de garantir l'acheminement et la validité des informations qui transitent dans un VANET. L'efficacité d'un tel réseau repose en bonne partie sur la confiance qui s'établit entre ses nœuds. La littérature scientifique fournit, dans ce cadre, différentes familles de modèles de confiance.

En première ligne, les solutions cryptographiques asymétriques telles que les *PKI (Public Key Infrastructure)* offrent des garanties essentielles pour des communications électroniques point à point. On assure ainsi :

- *l'authentification* : lors de l'envoi d'un message, on identifie de manière sûre l'émetteur ;
- *la confidentialité* : seul le destinataire légitime d'un message pourra en avoir une vision intelligible ;
- *l'intégrité* : le message expédié ne peut être altéré ;
- *la non répudiation* : une entité ne peut renier une transaction douteuse.

Bien que destinées principalement à des environnements centralisés, les PKI peuvent être utilisées, sous réserve des bonnes hypothèses, dans des environnements ad hoc. Mais dans tous les cas, les garanties offertes sont insuffisantes. Un nœud parfaitement authentifié peut être corrompu ou tout simplement volé. La carence essentielle ici, c'est l'absence de contrôle continu.

Les *modèles de réputation* jouent là un rôle inaliénable. Pour autant que les comportements délictueux soient observables, les systèmes de réputation [2] sont très utiles pour les gérer et les décourager. L'hypothèse de base est que le comportement antérieur d'une entité est un bon indicateur de son comportement futur. Partant de là, un nœud se construira une bonne ou une mauvaise réputation auprès des autres selon ses agissements passés. Cette réputation conditionne ses interactions avec les autres nœuds, lui valant droits et privilèges ou isolation et rejet. Les modèles de réputation souffrent cependant de quelques sérieuses carences. En effet, avant de conclure qu'un nœud se comporte mal, il faut, et c'est là un mécanisme tout à fait pertinent, un certain nombre d'observations concordantes sur une certaine période. Le problème, c'est qu'un nœud malfaisant ou égoïste pourra en jouer pour ne pas se faire épingler. De plus, le même logiciel responsable des comportements délictueux pourra se retrouver au niveau de plusieurs nœuds qui bénéficieront pourtant de la même prudence dans leur classification.

C'est dans ce cadre que se retrouve la dernière famille de modèles de confiance : l'informatique de confiance selon le *TCG* (pour Trusted Computing Group). Là, le concept phare de confiance enracinée dans le matériel (« hardware rooted trust ») voit sa concrétisation dans la spécification d'une puce : les Trusted Platform Modules (TPM) [19]. Microcontrôleur fixé sur une plate-forme, le TPM permet notamment la mise en

place d'une procédure de démarrage sécurisé qui pré-conditionne les chargements et les exécutions des différents programmes et modules aux mesures d'intégrité effectuées sur la plate-forme. Dans un réseau, une plate-forme équipée d'un TPM mettra en œuvre un double concept d'identité-intégrité. Des mécanismes de mesures, de stockage et de rapport d'intégrité sont couplés avec un système d'attestations (sur le modèle PKI) pour permettre la diffusion d'informations exactes sur l'identité et l'intégrité de chaque plate-forme. Une plate-forme disposant d'un TPM valide devrait ainsi rapporter sans falsification sa configuration logicielle aux autres plates-formes. Ceci ouvre des perspectives nouvelles mais une question reste en suspens : quelles peuvent être ces mesures et surtout comment les interpréter et les utiliser ?

1.3 Objectifs de recherche

L'objectif de ce mémoire est de proposer un modèle de réputation efficace basé sur la réputation des composants logiciels actifs dans un nœud. De manière plus spécifique, ce mémoire vise à

- concevoir un nouveau modèle de système de réputation pour les MANET en s'inspirant des modèles existants et en s'appuyant sur les technologies TCG ;
- implémenter le modèle de réputation proposé de manière à obtenir une simulation suffisamment paramétrable permettant la génération de scénarios très divers qui puissent approcher la réalité ;
- évaluer la performance du modèle de réputation proposé.

1.4 Plan du mémoire

Faisant suite au chapitre d'introduction, le chapitre II est une revue de littérature qui approfondit la problématique, présente brièvement les différents systèmes de réputation proposés dans la littérature en s'attachant à en extraire les considérations essentielles. Les spécifications techniques essentielles des technologies TCG retenues y seront également exposées et discutées. Le chapitre III ouvre une réflexion sur l'utilisation des standards TCG dans des MANET, décrit l'architecture du modèle de

réputation proposé, et présente les algorithmes développés. Le chapitre IV présente le travail d'implémentation, l'environnement de simulation et les résultats qui seront analysés et discutés. Ce mémoire s'achève sur une synthèse des travaux effectués et des directions de recherche future.

CHAPITRE II

SÉCURITÉ ET COOPÉRATION DANS UN MANET

Sécurité et coopération sont des défis majeurs pour les MANET. La sécurité dans un MANET est essentiellement perçue par rapport à son impact sur le bon fonctionnement du réseau ; bon fonctionnement qui se traduit par une coopération effective et sans malice des nœuds. Beaucoup de travaux ont été conduits sur ces problématiques et il convient de s'y intéresser dans les détails. De la même façon, les mécanismes proposés par le TCG demandent certains éclaircissements. Dans ce chapitre, nous commencerons par un état des problématiques de sécurité dans les MANET. Suivra une présentation des modèles de réputation, dans leurs principes généraux et dans leur application aux MANET. Nous aborderons ensuite les spécifications TCG relatives aux TPM pour finir sur des précisions quant à la nature des mesures d'intégrité d'une plate-forme.

2.1 Problématiques de sécurité dans les MANET

La sécurité dans les MANET est un champ très actif de la recherche. La nature décentralisée et la topologie très dynamique de ces réseaux posent des défis uniques en matière de sécurité. Les nombreux travaux qui s'y rapportent s'articulent autour de plusieurs problématiques qui se recoupent.

La sécurisation des protocoles de routage ad hoc

Dans un réseau ad hoc, étant donnée l'absence de toute centralisation, la question du routage devient complexe. Chaque nœud est potentiellement un routeur et peut servir de relais pour acheminer l'information. Les protocoles de routage se chargent donc de maintenir à jour des tables de routage raisonnablement réduites, de choisir les « meilleures » routes pour joindre une destination donnée etc. On y distingue généralement les protocoles proactifs (les nœuds rafraichissent périodiquement leurs tables de routage) tels OLSR, DSDV et les protocoles réactifs (le rafraichissement des

tables se fait sur demande lorsqu'un nœud veut en joindre un autre) tels AODV, DSR. On peut aussi recenser un certain nombre de protocoles hybrides (comme ZRP) qui essaient avec plus ou moins de succès de combiner les avantages des deux principales approches.

La sécurisation de ces protocoles est une question essentielle [7] au regard des vulnérabilités plus prononcées des MANET sur le routage. Des mécanismes de surveillance et de rapport des activités de routage des nœuds s'avèrent indispensables.

Les modèles de confiance

Un MANET est en général un environnement très ouvert : des nœuds rejoignent et quittent le réseau assez fréquemment. L'établissement de liens de confiance entre les nœuds du réseau requiert une attention particulière. Les solutions cryptographiques asymétriques telles que les infrastructures à clés publiques (ICP ou PKI pour Public Key Infrastructure) sont ici en première ligne dans le sens où des éléments de cryptographie constituent des hypothèses de départ pour la plupart des autres modèles de confiance. Les modèles de réputation proposent, quant à eux, l'établissement des liens de confiance sur la base des comportements antérieurs observés ou rapportés. Vient ensuite, du moins chronologiquement, l'informatique de confiance (Trusted Computing) dont la mise en place ouvre, dans ce domaine, des perspectives nouvelles et pour le moins intéressantes.

Les mécanismes renforçant la collaboration

Dans un MANET, il y a principalement deux sortes de nœuds « non grata » : les nœuds délibérément « malfaisants » et les nœuds égoïstes. Les nœuds délibérément malfaisants ont pour objectif assumé de nuire au réseau et essayeront activement d'y arriver. Par contre, les nœuds égoïstes mettront en danger le réseau de manière passive, essentiellement par souci d'économie de leur énergie. Ils mettent en œuvre des comportements égoïstes en tentant de se soustraire à leur rôle de routeur, menaçant ainsi les hypothèses de départ des MANET. D'où la nécessité d'avoir des mécanismes incitatifs pour la collaboration. On a besoin ici, pour parler trivialement, d'une bonne politique du « bâton et de la carotte » pour punir les « égoïstes » et récompenser les

« bons » nœuds. De nombreux systèmes, notamment de réputation [17], se proposent de répondre à ces problèmes.

Ce tour d'horizon rapide des problématiques de sécurité dans les MANET nous indique que la confiance dans un MANET implique souvent un contrôle continu qui ne peut se réduire à la présentation de bons certificats (numériques). Du fait de la structure décentralisée des MANET, les nœuds se surveillent entre eux et devraient pouvoir faire appliquer les règles de bon fonctionnement du réseau. Analysons ici une spécificité importante des MANET : les « nœuds égoïstes ». A elle seule, cette expression, qui ne détonnerait pas en sociologie ou quelque autre science humaine, est une indication sérieuse sur une approche adéquate du sujet. Les nœuds dans un MANET sont des agents autonomes dont la participation effective et bien intentionnée au bon fonctionnement du réseau n'est pas acquise. On est ici dans un milieu très hétérogène où les interactions des nœuds entre eux et avec leur environnement ne sont pas dictées par une autorité centrale. Toutefois, les interactions d'un nœud donné peuvent être observées, rapportées pour provoquer des réponses appropriées de la part des autres nœuds. C'est ce créneau essentiel qu'occupent les modèles de réputation.

2.2 Les modèles de réputation

Les modèles de réputation sont très utilisés dans les réseaux « pair à pair » (P2P). Les MANET se retrouvent dans ce cadre mais présentent quelques spécificités essentielles. Nous allons donc d'abord présenter les systèmes de réputation et leurs principes généraux dans tout réseau P2P. Relevant ensuite les spécificités des MANET, nous présenterons quelques systèmes de réputation dédiés aux réseaux mobiles ad hoc. Enfin, nous ferons la synthèse des modèles de réputation dans ce qui en constitue la substance, les points forts et les points faibles.

2.2.1 Présentation des systèmes de réputation

Comme nous l'avons déjà exposé au chapitre d'introduction, les comportements des entités impliquées dans une communauté sont le fondement des modèles de

réputation. Les modèles de réputation sont en fait des modèles sociaux dont les principes sous-jacents sont directement inspirés du concept de « réputation » tel qu'on le connait dans toutes les sociétés humaines.

En théorie, voici ce qu'on devrait avoir : deux ou plusieurs humains interagissent, et chacun d'eux ressort de cette interaction avec une opinion plus ou moins positive, plus ou moins juste sur chacun des autres participants. Ces opinions seront éventuellement rapportées à d'autres personnes, construisant progressivement la réputation d'une personne. Cette réputation pourra être rattachée à un contexte particulier : un « bon » étudiant mais un « mauvais » orateur etc.

En pratique, les choses sont un peu plus complexes ; non seulement les opinions sur une interaction donnée peuvent être biaisées, délibérément ou en toute bonne foi, mais il ne sera pas rare que des opinions soient construites sur la base d'interactions fictives ou tout au moins falsifiées. On pourra également observer, plus rarement, des usurpations d'identité qui fausseront les données de réputation. La réputation associée à un individu ou à une entité, n'est donc pas forcément le reflet d'une réalité objective. Les principes de base des modèles de réputation souffrent dans la pratique de tares importantes.

Dans des réseaux informatiques, les MANET en particulier, toutes ces difficultés se retrouvent renforcées, compte tenu de certaines spécificités des contextes informatiques (notamment sur les questions d'identité). Les possibilités de mystification organisée sont démultipliées. En tant que réseau P2P, les MANET sont concernés par la littérature traitant des systèmes de réputation dans les P2P. Présentons-en sommairement quelques points cruciaux.

2.2.2 Systèmes de réputation pour réseaux P2P

Nous sommes ici essentiellement dans un contexte de transactions vendeur/acheteur ou fournisseur/client. Le client requiert les services - téléchargement d'une ressource, obtention d'une information etc. - d'un fournisseur (éventuellement préalablement sélectionné). Une fois la transaction effectuée, le client marque son

niveau de satisfaction en la notant. Un feedback peut être envoyé au fournisseur, qui pourra le présenter à ses futurs clients, charge à ces derniers d'aller éventuellement le confirmer auprès des précédents clients. On retient ici que les cotes seront souvent attribuées directement par les clients sans mise en œuvre de mécanismes particulièrement complexes.

Bien évidemment, se créer une bonne réputation ou contribuer à la mauvaise réputation du concurrent est une démarche intéressante pour beaucoup. La construction de la réputation d'une entité doit faire face à un certain panel d'attaques.

Les principales attaques

Selon [3], la tricherie consiste, d*u côté des clients*, à l'attribution de cotes incorrectes (surestimées ou sous-estimées) ne reflétant pas la réelle qualité du service qui leur a été fournie. *Du côté des fournisseurs*, on peut relever de la discrimination (positive ou négative). La technique qui facilite, voire conditionne la capacité de nuisance de tous ces comportements est celle connue sous le nom de « Sybil Attack », d'après le célèbre cas de schizophrénie relaté dans le roman « Sybil » de Flora Schreiber. Une attaque Sybil consiste pour un attaquant à détourner un système de réputation en se créant un très grand nombre de pseudonymes, pour ensuite les utiliser dans le but de se créer une influence disproportionnée. La prise en compte de telles attaques est cruciale dans la modélisation d'un système de réputation.

Modélisation

La modélisation d'un système de réputation demande la considération de certains points importants. Certains auteurs [11] identifient le stockage et l'intégrité de l'information de réputation, les métriques de réputation et le changement d'identité.

1. Le **stockage** de l'information de réputation

Il devrait être distribué (l'information de réputation d'un nœud donné est répartie entre de nombreux nœuds) mais avec une grande disponibilité qui tienne compte du dynamisme des liaisons dans un P2P.

2. L'**intégrité** de l'information de réputation

C'est un point critique qui détermine l'utilité du système de réputation et dont la garantie est plus difficile dans un environnement décentralisé.

3. Les **métriques** de réputation

La complexité de leur calcul détermine les performances de tout le système. Certains auteurs [15] introduisent dans la gestion des rapports de réputation des autres nœuds, la prise en compte des contextes de transaction.

4. Le **changement d'identité**

Parfois très aisé dans un P2P et généralement de coût nul à la différence du monde réel, il doit être découragé par tout bon système de réputation selon [4]. Ainsi, une certaine forme de « bizutage », de « périple initiatique » pourra être sainement pratiquée à l'encontre des nouveaux venus. La philosophie ici est de contrer les stratégies malicieuses qui consisteraient pour un nœud corrompu à effacer son passif en se créant une nouvelle identité.

A ces considérations, s'ajoutent certains points complémentaires tels que le maintien de l'anonymat, l'impact de la surcouche de réputation, la robustesse du système, les éventuels rôles et privilèges etc. Abordons les ici brièvement :

- Le *maintien de l'anonymat* : La réputation d'une entité sera de préférence associée à un identifiant opaque (nom d'utilisateur dans la communauté considérée) plutôt qu'à une identité plus retraçable (comme une adresse IP). C'est un point qui interdit une solution simpliste à la question du changement d'identité.
- La *minimisation en termes de calcul, d'infrastructure, de stockage et de complexité* des messages de la surcouche « réputation » appliquée.
- La *robustesse du système aux attaques de collectifs d'entités malicieuses.*
- Les *notions de rôles ou de privilèges,* liées à la gestion du changement d'identité.
- Les *contextes des métriques de réputation* amenant une classification des transactions.
- L'utilisation (ou non) de tierces parties de confiance (« *trusted-third-party* ») pour aider à l'authentification.

2.2.3 Les spécificités des MANET

Les transactions telles qu'elles sont effectuées dans des réseaux P2P concernent également les MANET. Mais les MANET se distinguent des réseaux P2P classiques par les problématiques du routage (trouver le chemin pour un paquet) et de la retransmission (relayer un paquet pour les autres nœuds). Toute une gamme de comportements délictueux entre ici en ligne de compte et il faut pouvoir les détecter.

Les mauvais comportements

Du fait de la distribution de la fonction de routage dans les MANET, tout comportement déviant de cette obligation de collaboration entre les nœuds mérite d'être dénoncé. En dehors des attaques délibérées de nœuds malfaisants, l'égoïsme des nœuds est en effet un problème essentiel des MANET. Des mesures de rétorsion sont donc indispensables en cas de mauvais comportements. Faute de quoi, échanges réduits, dénis de services et partitionnements du réseau sont à redouter. Au final, lorsqu'une application des MANET - telle que le V2V - est proposée, des niveaux aussi bas de fiabilité n'exerceront qu'une séduction pour le moins modérée sur la plupart des utilisateurs. Une méfiance généralisée et la crainte de l'absence de réciprocité ne favorisent pas l'adhésion à un système qui se veut collaboratif. Selon [17], les modèles de réputation sont les plus complets pour résoudre ces problématiques. La détection des comportements déviants devient dans ce cas un préalable évident.

Le « Monitoring »

Il s'agit de rassembler de l'information de première main sur les comportements des nœuds dans le réseau. Dans les MANET, cette information de première main n'est pas « gratuite ». Plusieurs mécanismes [2] peuvent être mis en œuvre. Lorsqu'un nœud source A envoie un paquet a un nœud destination B, un accusé de réception (ACK) de B pourra confirmer auprès de A la bonne coopération de tous les nœuds présents sur la route. A l'aide d'antennes omnidirectionnelles, des accusés de réception passifs (PACK) pourront aussi être obtenus par un nœud source en pratiquant une écoute sur les transmissions du prochain nœud relais s'il est à sa portée. Ces divers mécanismes de détection alimentent de manière égale divers modèles de réputation.

2.2.4 Quelques systèmes de réputation pour les MANET

Nous nous limiterons ici à la présentation de quelques systèmes de réputation conçus comme des ajouts sécuritaires au protocole de routage réactif DSR, très prisé dans la recherche sur les modèles de réputation dans les MANET.

Le protocole DSR

Nous présentons à la Figure 2.1 les principes de fonctionnement du protocole DSR Dynamic Source Routing [6].

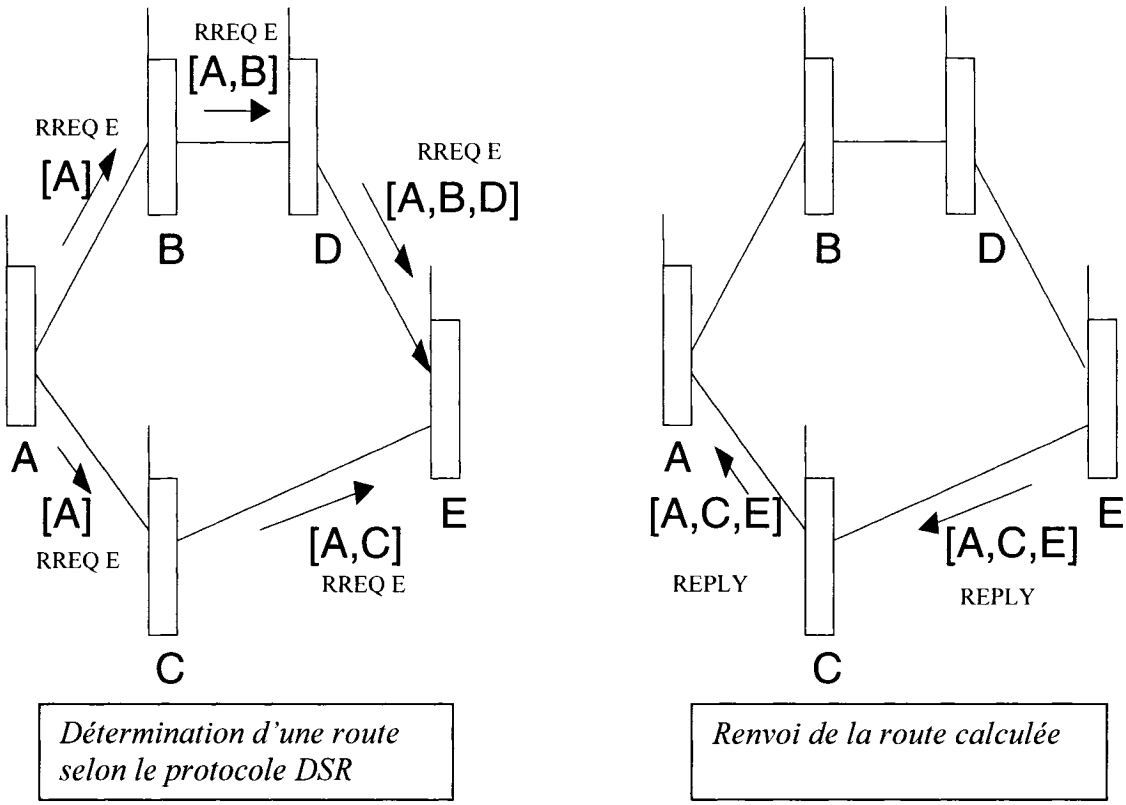


Figure 2.1 DSR : Construction d'une route sur demande

Lorsqu'un nœud désire en joindre un autre, il inonde le réseau de messages ROUTE REQUEST (RREQ) en spécifiant la destination à atteindre. Tous les nœuds qui reçoivent ces messages les retransmettent en y rajoutant leur propre adresse. Si un nœud recevant le message est la destination ou connait la route vers la destination, il renvoie un REPLY au nœud source avec la route complétée.

Le nœud source, après avoir reçu un ou plusieurs REPLY, choisit la meilleure route (typiquement la plus courte), l'enregistre et commence à envoyer des messages sur cette route. En cas de problème (rupture de liaison), le nœud qui détecte le problème envoie un ROUTE ERROR (RERR) au nœud source qui doit choisir ou chercher une autre route.

Quelques systèmes de réputation pour les MANET

Il s'agit pour nous de donner un bref aperçu des mécanismes mis en œuvre dans un système de réputation et de leur agencement dans des systèmes concrets. Nous présenterons dans ce qui suit Watchdog / PathRater, CONFIDANT, CORE, Context-Aware Detection, SORI.

Watchdog / Pathrater

Marti et al. (2000) [9] ont proposé un mécanisme qui tient souvent lieu de référence dans la littérature des systèmes de réputation pour MANET. Le mauvais comportement visé est la non retransmission des paquets. Deux mécanismes sont mis en œuvre. Le Watchdog permet à un nœud source de surveiller (par écoute) les retransmissions des nœuds situés sur la route qu'emprunte son message vers le nœud destination. Avec le PathRater, chaque nœud maintient des informations de réputation (cotes) sur tous les nœuds du réseau et met à jour périodiquement les informations (cotes) relatives aux nœuds qu'il utilise le plus. La route sélectionnée est alors celle qui présente la plus haute moyenne (des cotes) sur l'ensemble des nœuds qui la constituent. Le but ici est d'éviter les nœuds qui présentent des mauvais comportements. A la différence de la quasi-totalité des systèmes de réputation pour MANET, aucun mécanisme de rétorsion n'est mis en œuvre à l'encontre des nœuds peu coopératifs ; leurs messages continuent à être relayés. Au final, on peut déplorer que les nœuds peu coopératifs soient tout simplement déchargés de leur devoir de coopération.

CONFIDANT

Buchegger et Le Boudec (2004) [1] ont proposé CONFIDANT (Cooperation Of Nodes, Fairness in Dynamic Ad Hoc Networks). Ce système propose un système

adaptif bayésien de confiance et de réputation. Les nœuds surveillent leur voisinage et détectent plusieurs types de mauvais comportements grâce à un mécanisme PACK (Passive ACK) amélioré. Des informations de réputation de seconde main sont aussi collectées depuis les autres nœuds. Par estimations bayésiennes, chaque nœud classifie les autres comme normaux ou non coopératifs. Un nœud qui est jugé coupable de mauvais comportements est tout simplement ignoré et de fait isolé du réseau.

CORE

Le système CORE (COllaborative REputation), proposé par Michiardi et Molva (2002) [10], utilise une composante Watchdog pour la surveillance des comportements. Le mécanisme de réputation complémentaire distingue la réputation subjective (les observations directes), la réputation indirecte (seules des informations de réputation positive sont échangées) et la réputation fonctionnelle (comportements par rapport à une tâche spécifique). Ces trois réputations sont pondérées pour donner une valeur de réputation qui est utilisée pour la prise de décisions et l'isolation graduelle d'un nœud. Les valeurs de réputation sont obtenues en plaçant les nœuds dans une optique client/fournisseur et en comparant le résultat espéré d'une interaction avec le résultat obtenu.

SORI

Le système SORI (Secure and Objective Reputation-based Incentive), proposé par He, Wu et Khosla (2004) [5], veut décourager les mauvais comportements liés au défaut de retransmission des paquets. Le mécanisme de surveillance utilisé est similaire au Watchdog. La cote d'un nœud est calculée en comparant le nombre de paquets qu'il lui a été demandé de relayer au nombre de paquets qu'il a effectivement retransmis. Les informations de réputation sont échangées localement. Ces informations de seconde main sont pondérées par la crédibilité (la cote) des nœuds qui les propagent. Les nœuds qui présentent un mauvais comportement augmentent la probabilité de voir leurs paquets se faire jeter.

Soit N un nœud du réseau à qui un autre nœud X demande un service (relais de ses paquets). N doit décider selon la réputation de X si X mérite ce service.

➢ *Création d'une opinion de N sur X*

N a sa propre opinion sur X qui se calcule comme suit :

$RF_N(X)$: nombre de paquets de N envoyés pour relais à X

$HF_N(X)$: nombre de paquets de N effectivement relayés par X

$LER_N(X)$: opinion locale de N sur X, est une structure à deux champs.

(évaluation) $G_N(X) = HF_N(X) / RF_N(X)$

(niveau de confiance) $C_N(X) = RF_N(X)$

Le premier champ $G_N(X)$ est un réel entre 0 et 1 qui rend compte du pourcentage de paquets de N effectivement relayés par X tandis que le second, $C_N(X)$, témoigne de l'importance accordée à $G_N(X)$. On pose ici que plus il y a eu d'expériences, mieux $G_N(X)$ reflète le comportement de X.

➢ *Collecte des opinions auprès des voisins*

La collecte des informations de seconde main se fait auprès des voisins. Les opinions collectées auprès des voisins demandent à être pondérées par le crédit que leur accorde N. Dans SORI, l'opinion sur X, $LER_i(X)=\{C_i(X), G_i(X)\}$, rapportée par un voisin i est pondérée par $G_N(i)$ telle que décrite ci-dessus.

➢ *Raisonnement appliqué*

Il s'agit de déterminer la probabilité p selon laquelle sera rendu le service à X. On calcule d'abord la réputation de X comme suit. On désigne par NL la liste des voisins de N, N inclus.

$$OER_N(X) = \frac{\sum_{i \in NL, i \neq X} G_N(i) * C_i(X) * G_i(X)}{\sum_{i \in NL, i \neq X} G_N(i) * C_i(X)}$$

Cette réputation représente, pour N et son voisinage, le rapport du nombre de paquets effectivement relayés par X sur le nombre de paquets envoyés (pour relais) à X. Les opinions des voisins sont pondérées comme décrit ci-dessus. On a ici $G_N(N) = 1$ et pour tout voisin i tel que $C_N(i) = 0$, on a $G_N(i) = 0$

Soit $q = 1 - OER_N(X)$ qui représente le taux d'échec de X dans la retransmission des paquets et préfigure de la probabilité de refus de relais des paquets de X par N. Un paramètre supplémentaire δ est introduit et représente la tolérance du système. En dessous de δ, on aurait un taux d'échec acceptable et donc une probabilité nulle de rejet des paquets de X.

Au final, on calcule comme suit p :

$$p = \begin{cases} q - \delta & si\ q > \delta \\ 0 & sinon \end{cases}, \delta\ tolérance\ du\ système$$

2.2.5 Synthèse

Un système de réputation pose comme hypothèse de base que le comportement antérieur d'une entité est un bon indicateur de son comportement futur. Un mécanisme plus ou moins complexe, selon les environnements considérés, est indispensable pour observer et noter les comportements des autres entités. Si, dans les communautés virtuelles, l'appréciation d'une interaction est laissée au seul soin de l'utilisateur, les MANET devront mettre en œuvre des routines de surveillance.

Une fois ce point réglé, un système de réputation peut être décomposé en trois étapes :

1) la création et le contenu d'une « opinion » ;
2) la collecte d'autres « opinions » ;
3) l'interprétation et le raisonnement appliqués à l'information collectée.

La création d'une opinion implique la synthèse de l'expérience personnelle d'un nœud en une forme exploitable par ses voisins. Elle peut inclure uniquement des expériences négatives ou des expériences positives ou les deux, être une moyenne des observations passées, une position hiérarchique etc. Elle peut intégrer un mécanisme priorisant les observations les plus récentes. Éventuellement, l'« opinion » peut même donner un niveau de certitude sur sa propre estimation.

La collecte d'autres opinions peut être active - et très sélective - ou passive, le nœud se contentant d'attendre des rapports d'opinions des nœuds à sa portée. Les

opinions collectées peuvent être pondérées suivant la confiance accordée aux entités les rapportant.

Ces opinions collectées demandent à être combinées avec l'expérience locale pour aboutir à une décision sur la confiance à accorder à l'entité concernée. La gamme de mécanismes déployés pour y arriver est très étendue. Dans certains environnements, la peur des représailles en cas d'erreur pourra générer plus de souplesse dans les décisions.

Les systèmes de réputation sont très utiles dans les communautés virtuelles où les utilisateurs ont fréquemment l'opportunité d'interagir avec d'autres utilisateurs qu'ils ne connaissent pas au préalable. Les cotes sur les comportements antérieurs jouent là un rôle inaliénable. Dans certains environnements très dynamiques et peu maîtrisés où les critères d'admissibilité sont souples, les entités ne peuvent être jugées que sur la base de leurs agissements antérieurs.

La fiabilité des systèmes de réputation a ses limites. Même en maintenant une réputation fidèle sur chaque entité, on ne peut garantir à 100% son futur. Ainsi, quelque soit son niveau de perfection, un système de réputation ne pourra pas éviter qu'une entité jusque là parfaite puisse commettre ponctuellement d'importants « délits ».

2.3 L'architecture TCG

Les plates-formes basées sur les spécifications TCG [19] sont censées satisfaire des standards de fonctionnalité et de fiabilité permettant un niveau de confiance rehaussé. TCG développe l'idée des « roots of trust », racines de confiance à partir desquelles se construit la confiance accordée à une plate-forme. Le concept de confiance enracinée dans le matériel (« hardware-rooted trust ») est central au TCG et est concrétisé par la conception du TPM (Trusted Platform Module). Le TPM est une puce qui est appelée à être installée sur des plates-formes pour en constituer la racine de confiance. Le TCG définit ainsi dans un contexte multi-plates-formes les couches d'abstraction indiquées à la Figure 2.2.

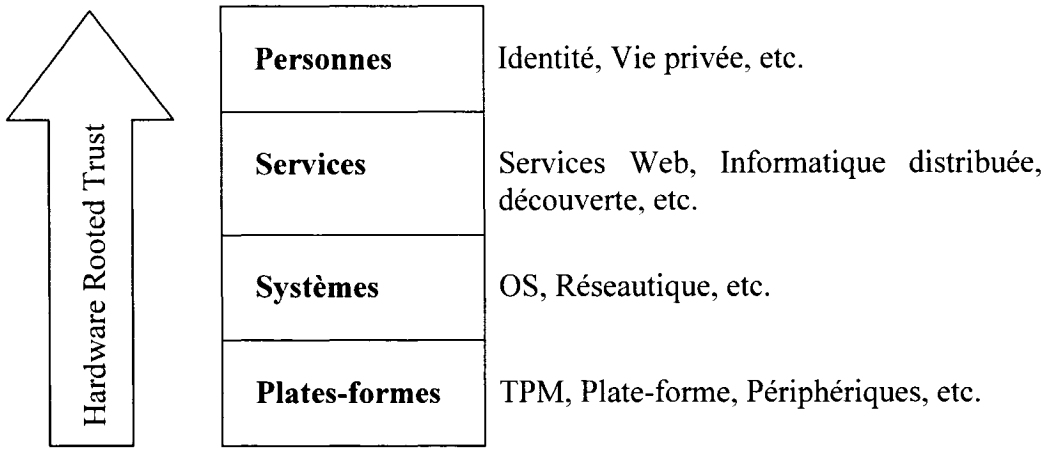


Figure 2.2 Hardware Rooted Trust et couches d'abstraction

Le TPM se retrouve à la racine d'une chaîne transitive de confiance qui peut conditionner jusqu'au démarrage d'une plate-forme comme le montre la Figure 2.3. Au premier niveau de cette chaine de confiance, se retrouvent, à côté du TPM, les TBB (Trusted Building Blocks). Ces TBB représentent les composants de la plate-forme à qui on ne peut que faire confiance (connections physiques du TPM à la carte mère de la plate-forme, bus etc.). Ils sont encadrés en gras à la Figure 2.4.

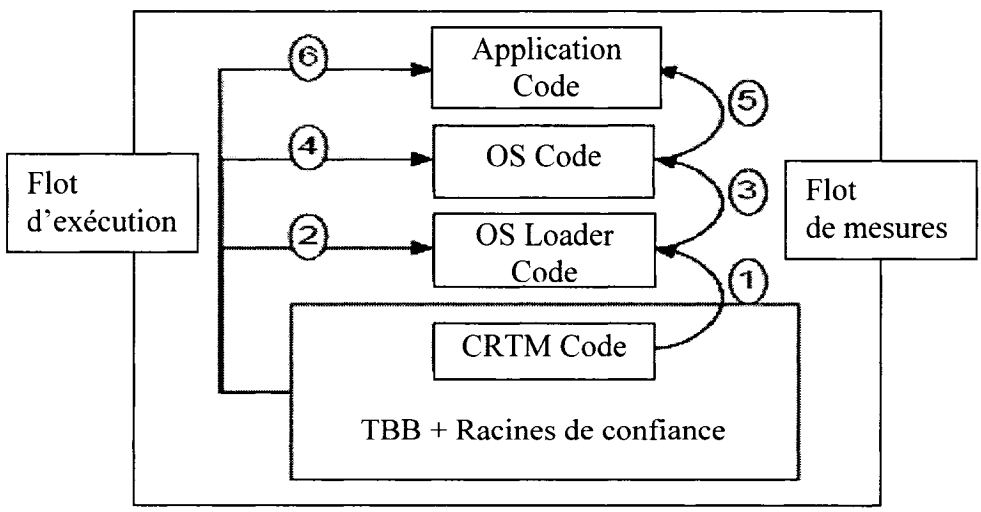


Figure 2.3 Hardware Rooted Trust et Secure Boot avec un TPM

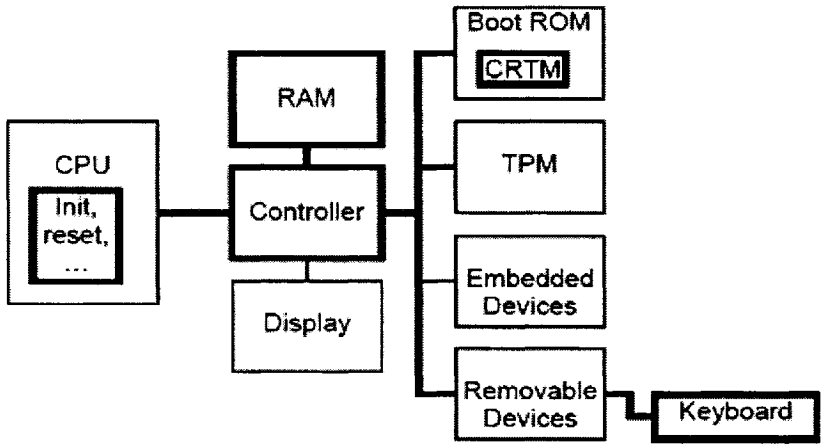


Figure 2.4 Plate-forme de confiance (TBB+TPM)

La Figure 2.5 donne un aperçu de l'architecture interne d'un TPM. On peut y remarquer une capacité de stockage non volatile, des moteurs de cryptage / décryptage RSA et SHA-1, un générateur de nombres aléatoires, un module de génération de clés, des registres rendant compte de la configuration d'une plate-forme, etc. Un module Opt-In permet un choix sur l'activation du TPM.

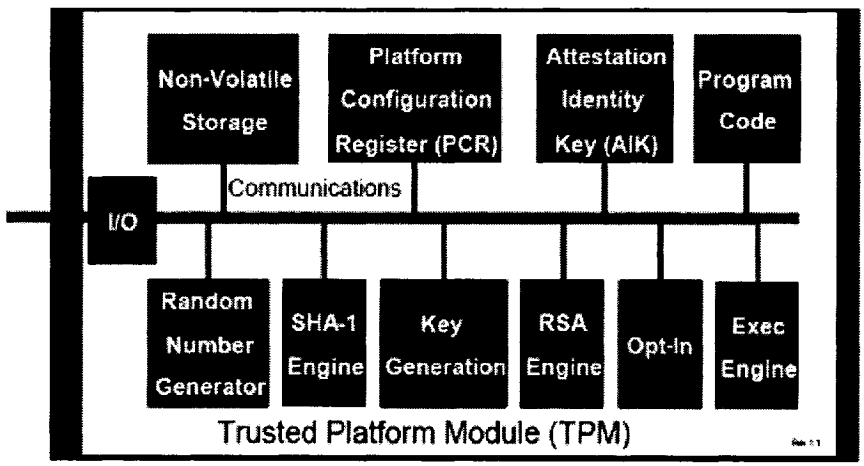


Figure 2.5 Architecture interne d'un TPM

Dans les sous-sections qui vont suivre, nous allons présenter les caractéristiques fondamentales du TPM, notamment le système d'attestations, avant de nous intéresser plus en détail au protocole TCG de rapport d'intégrité. Nous finirons sur ce qu'il faut retenir à propos des plates-formes équipées de TPM.

2.3.1 Principales caractéristiques du TPM

Le TPM se distingue par trois caractéristiques fondamentales qui sont : les droits protégés, le rapport d'intégrité et les attestations.

Droits protégés

Les droits protégés sont un ensemble de commandes avec permission exclusive d'accès à des zones bouclées. Les zones bouclées sont des zones (mémoire, registres etc..) où les opérations sur les données sensibles peuvent s'effectuer sans risque. Le TPM implémente des droits protégés et des zones bouclées qu'il utilise pour protéger et rapporter les mesures d'intégrité.

Intégrité : Mesures, Stockage et Rapports

La mesure d'intégrité est le processus d'obtention des métriques des caractéristiques de la plate-forme qui peuvent affecter son intégrité.

Le stockage de l'intégrité assure le stockage intégral des métriques d'intégrité dans des SML (Store Measurement Log) complets et celui de leurs hashs dans des PCR.

Le rapport d'intégrité est le processus qui atteste auprès d'une entité extérieure du contenu des unités de stockage de l'intégrité.

Ce qu'il faut retenir, c'est qu'*une plate-forme peut éventuellement entrer dans n'importe quel état, même indésirable, mais ne doit pas pouvoir mentir à ce propos.* Un processus indépendant pourra évaluer cet état et en déterminer la réponse appropriée.

Les attestations

Une attestation est un procédé qui consiste à donner une garantie sur l'exactitude de l'information. Toutes les formes d'attestations requièrent des preuves fiables de l'entité qui les fournit. Ces preuves sont ici appelées « credentials », terme anglais que nous traduirons par le néologisme « crédentiels ».

Définitions

Certificat – Un certificat est une instanciation d'un crédentiel selon les standards industriels en vigueur. La génération d'un certificat consiste à assembler les valeurs pour les champs du crédentiel et à signer ces champs complétés

Crédentiel – On peut ainsi définir le crédentiel comme la preuve abstraite qui doit être instanciée en tant que certificat avant de pouvoir être échangée entre entités. On distingue :

• Les crédentiels utilisés pour la gestion de l'identité des plates-formes : ils contiennent la clé publique de la paire clé publique / clé privée qui est détenue à l'intérieur du TPM. C'est le cas des crédentiels EK (Endorsement Key) et AIK (Attestation Identity Key)

• Les crédentiels utilisés pour la gestion de l'intégrité des plates-formes : les crédentiels plate-forme qui représentent les TBB (Trusted Building Blocks)

La Figure 2.6 et le Tableau 2.1 illustrent le déploiement des TPM. Les puces TPM sont identifiées par un Endorsement Key unique et sont attachées ensuite à des plates-formes, elles-mêmes identifiées dans le réseau par leur(s) alias.

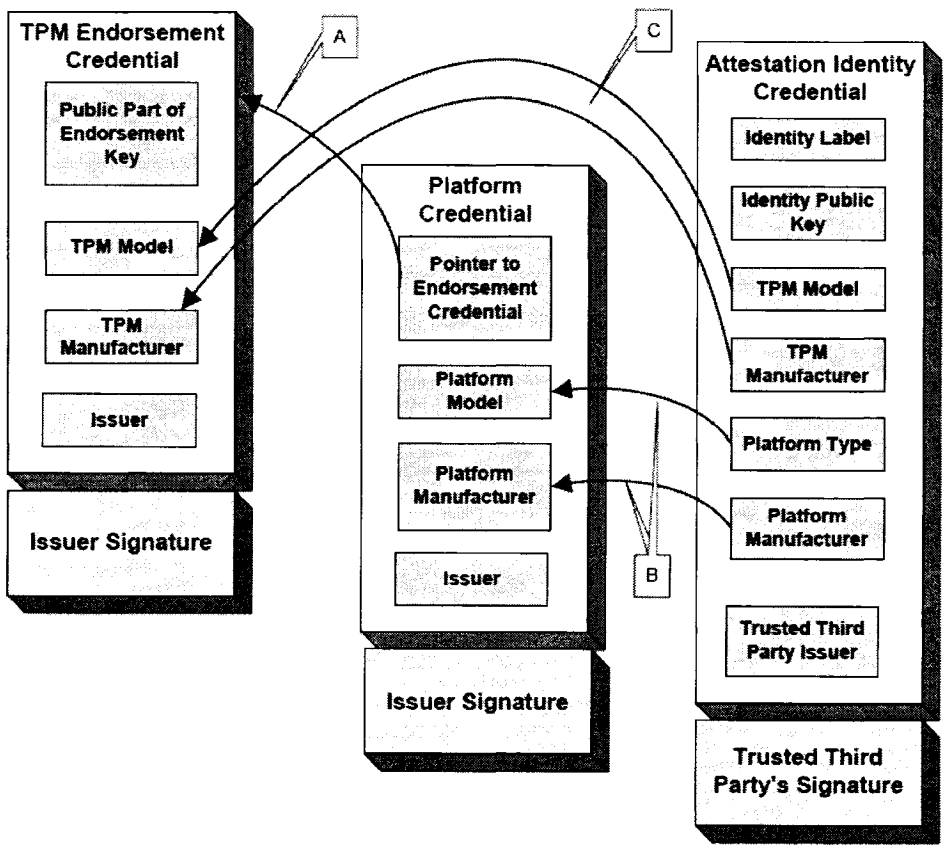


Figure 2.6 Diagramme des relations entre crédentiels

Tableau 2.1 Récapitulatif des crédentiels TPM

	Crédentiel EK	**Crédentiel Plate-forme**	**Crédentiel AIK**
Déploiement	L'Endorsement Key identifie de manière unique le TPM.	Le TPM est attaché à une plate-forme.	La plate-forme obtient (possiblement auprès d'un CA) 1 ou plusieurs alias d'identité AIK à des fins de protection de vie privée.
Délivreur	Fabricant TPM	Fabricant plate-forme	Un Privacy-CA, *sur requête d'un propriétaire de TPM et si cette requête satisfait les requis de sécurité et la politique de ce CA*
Utilisé	Par un Privacy CA	Par un Privacy CA *Pour vérifier que la plate-forme contient un TPM unique*	Dans un processus d'authentification (d'identification)
Atteste que	Le détenteur de la clé privée EK correspondante est un TPM conforme aux spécifications TCG	La plate-forme contient un unique TPM et des TBB intégres	Le AIK, est bien l'identité d'une plate-forme contenant un TPM conforme et unique satisfaisant les normes d'admission du CA
Révocation	Selon la politique du Privacy CA	-	En cas de compromission de la clé privée AIK ou EK

2.3.2 Le protocole de rapport d'intégrité

Il consiste en plusieurs étapes comme illustré à la Figure 2.7:

1. Un challenger requiert une ou plusieurs valeurs PCR d'une plate-forme
2. Un agent sur la plate-forme contenant un TPM collecte les entrées SML
3. L'agent plate-forme reçoit les valeurs PCR du TPM
4. Le TPM signe les valeurs PCR en utilisant un AIK
5. L'agent plate-forme collecte les crédentiels garants du TPM. Les valeurs PCR signées, les entrées SML et les crédentiels sont retournés au Challenger

6. Le Challenger vérifie la requête. Le digest de la mesure est calculé et comparé avec la valeur PCR. Les crédentiels de la plate-forme sont évalués et les signatures contrôlées.

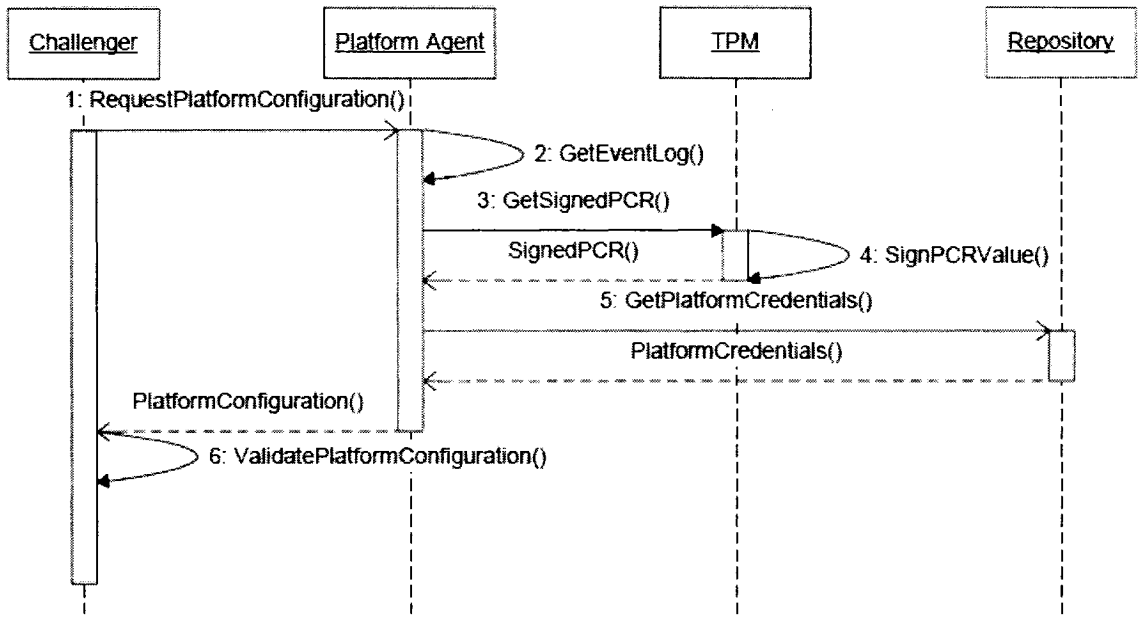


Figure 2.7 Protocole de rapport d'intégrité

2.3.3 TPM : l'essentiel

Ce qu'il faut retenir, c'est la démarche pour mettre en place un standard qui, s'il tient ses promesses, constituerait une avancée considérable pour la sécurité informatique. A ce niveau, il convient de relever et de souligner l'importance de la controverse qui anime la communauté scientifique quant à l'efficacité des propositions du TCG. Il existe un certain scepticisme quant aux travaux actuels du TCG mais on ne peut préjuger de l'avenir. Le concept du « hardware rooted trust » nous paraît suffisamment sensé pour qu'on puisse tabler sur des propositions suffisamment robustes à court ou moyen terme. Les autres points d'inquiétude s'étendent des considérations sur la protection des informations personnelles des utilisateurs aux risques de marginalisation des logiciels issus de la communauté du libre.

Revenons aux propositions du TCG pour mettre en lumière deux éléments :

On a pour chaque plate-forme un Endorsement Key (EK) faisant office d'identifiant absolu. La possibilité d'utiliser des alias est offerte à travers les AIK (Attestation Identity Key) mais le lien entre ces AIK et un EK valide devra être prouvé. Le recours à un CA constitue l'option la plus probable.

L'autre point important, c'est la notion de mesures et de rapport d'intégrité. Une plate-forme ne doit pas pouvoir mentir sur ses états passés (depuis le démarrage) ou présents. Elle est censée « se confesser » honnêtement quelque soit son état. La confiance accordée à une plate-forme sera donc fonction de ces mesures. Les politiques d'accès dans un réseau donné seront écrites pour prendre des décisions sur de telles mesures. Il convient de s'intéresser plus en détail et plus concrètement à ces mesures d'intégrité.

2.4 Configuration logicielle d'un nœud

Avec TCG et le rapport des mesures d'intégrité, les nœuds d'un réseau reprennent leur dimension de système plus ou moins complexe. Que faut-il alors rapporter comme mesures d'intégrité ?

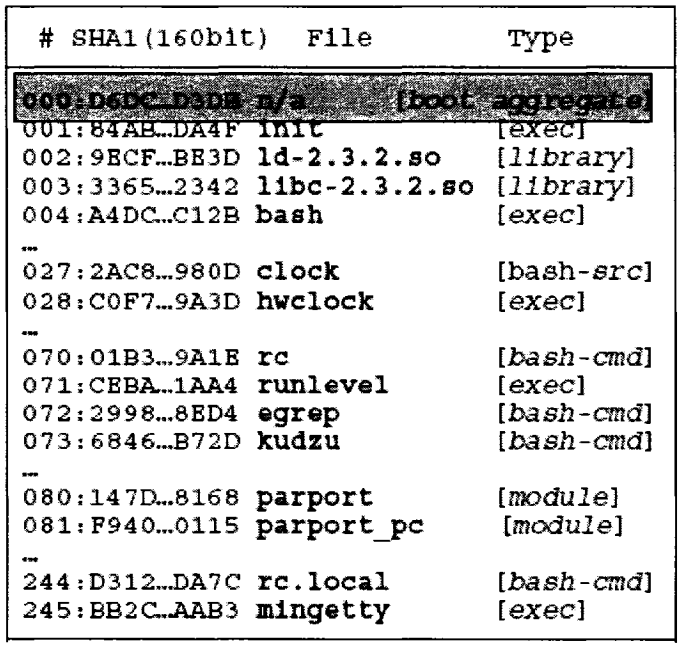
2.4.1 Configuration logicielle complète

Analysons ici la substance de la littérature traitant des mesures d'intégrité [13, 14] du TPM. La perception logicielle d'un nœud inclut les éléments suivants :

- *Modules du noyau du système d'exploitation*

Leur analyse est un préalable incontournable puisqu'ils affectent potentiellement l'architecture de mesure dans le noyau. Le noyau du nœud n'est pas censé subir des modifications régulières et non-répertoriées. Le domaine de validité de ces modules représentés à la Figure 2.8 (tirée de [14]) est donc bien déterminé puisque très statique.

Les moindres déviances feront d'un nœud suspect sur ce point un paria dans le réseau étant donné que ses mesures d'intégrité rapportées ne seraient plus dignes de confiance. Notre approche se veut la plus souple possible mais à ce niveau, nous préconisons la politique la plus stricte.



#	SHA1(160bit)	File	Type
000:	D6DC…D3DB	n/a	[boot aggregate]
001:	84AB…DA4F	init	[exec]
002:	9ECF…BE3D	ld-2.3.2.so	[library]
003:	3365…2342	libc-2.3.2.so	[library]
004:	A4DC…C12B	bash	[exec]
…			
027:	2AC8…980D	clock	[bash-src]
028:	C0F7…9A3D	hwclock	[exec]
…			
070:	01B3…9A1E	rc	[bash-cmd]
071:	CEBA…1AA4	runlevel	[exec]
072:	2998…8ED4	egrep	[bash-cmd]
073:	6846…B72D	kudzu	[bash-cmd]
…			
080:	147D…8168	parport	[module]
081:	F940…0115	parport_pc	[module]
…			
244:	D312…DA7C	rc.local	[bash-cmd]
245:	BB2C…AAB3	mingetty	[exec]

Figure 2.8 Boot Aggregate dans une liste de mesure d'intégrité

- *Exécutables et librairies partagées*

Ces éléments ne changent pas souvent et peuvent être reliés autant à leurs fonctionnalités qu'à leurs vulnérabilités connues. Il est fait mention dans les travaux TCG [14] à des bases de données centralisées qui détiennent les informations relatives aux vulnérabilités connues des logiciels mesurés. Cette approche cadre mal avec un contexte ad hoc et génère des délais supplémentaires qui seront prohibitifs dans un certain nombre d'environnements hautement dynamiques.

- *Fichiers de configuration*

Ils ne changent pas souvent une fois que le système est correctement configuré et peuvent être décisifs pour déterminer la fiabilité du programme qui les utilise.

- *Autres*

On retrouve ici les fichiers d'entrée importants qui peuvent affecter la fiabilité de la configuration : fichiers de commande bash, Java Servlets, librairies Java…

Les données d'entrée dynamique telles que les saisies d'utilisateur, les requêtes web et les commandes à distance ne sont pas prises en compte d'une part en raison de

leur peu d'intérêt et d'autre part parce que les éventuelles corruptions qui pourraient en découler sont supposées prises en charge par le noyau déjà précédemment mesuré.

La Figure 2.9 tirée de [14] illustre ce qui précède avec un système Linux. Les mesures sont des calculs de hash (SHA-1 sur la figure 2.8) des fichiers considérés . Elles constituent les empreintes digitales des logiciels en cours d'exécution. Le nœud envoie la totalité de ses mesures d'intégrité. Cela représente un volume de données potentiellement énorme. On a ici toute l'information utile mais il est plus que probable qu'un bon pourcentage de cette information ne soit pas pertinent dans un cadre de communication entre logiciels. Le protocole de rapport d'intégrité de TCG prévoit qu'un challenger demande des valeurs PCR spécifiques à l'entité avec laquelle elle s'apprête à interagir. Dans le cadre MANET qui nous intéresse, on peut considérer qu'il n'y ait pas besoin d'envoyer des métriques sur l'ensemble des logiciels qui tournent.

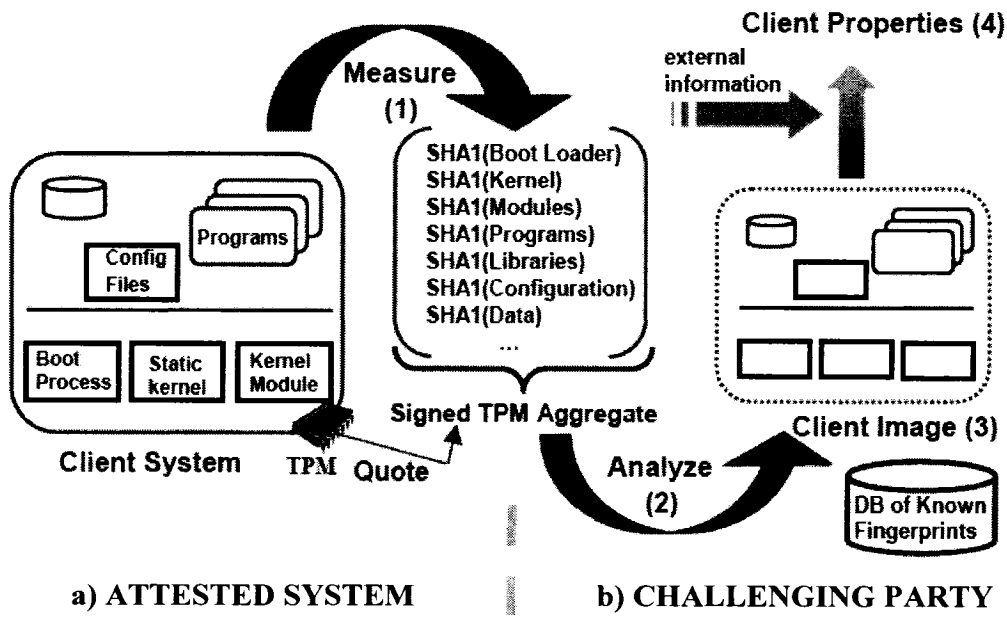


Figure 2.9 Rapport des mesures d'intégrité dans le détail

2.4.2 Configuration logicielle : le minimum requis

Un nœud envoie donc la liste des mesures d'intégrité qui révèlent sa configuration logicielle courante. Chaque nœud devra posséder les informations nécessaires à la validation ou non des mesures d'intégrité du démarrage. Une fois ce contrôle effectué, les mesures reportées seront jugées fiables. Reste à définir leur contenu qui devrait être réduit le plus possible à l'information utile.

L'exemple illustré à la Figure 2.10 avec ProcessExplorer est très intéressant.

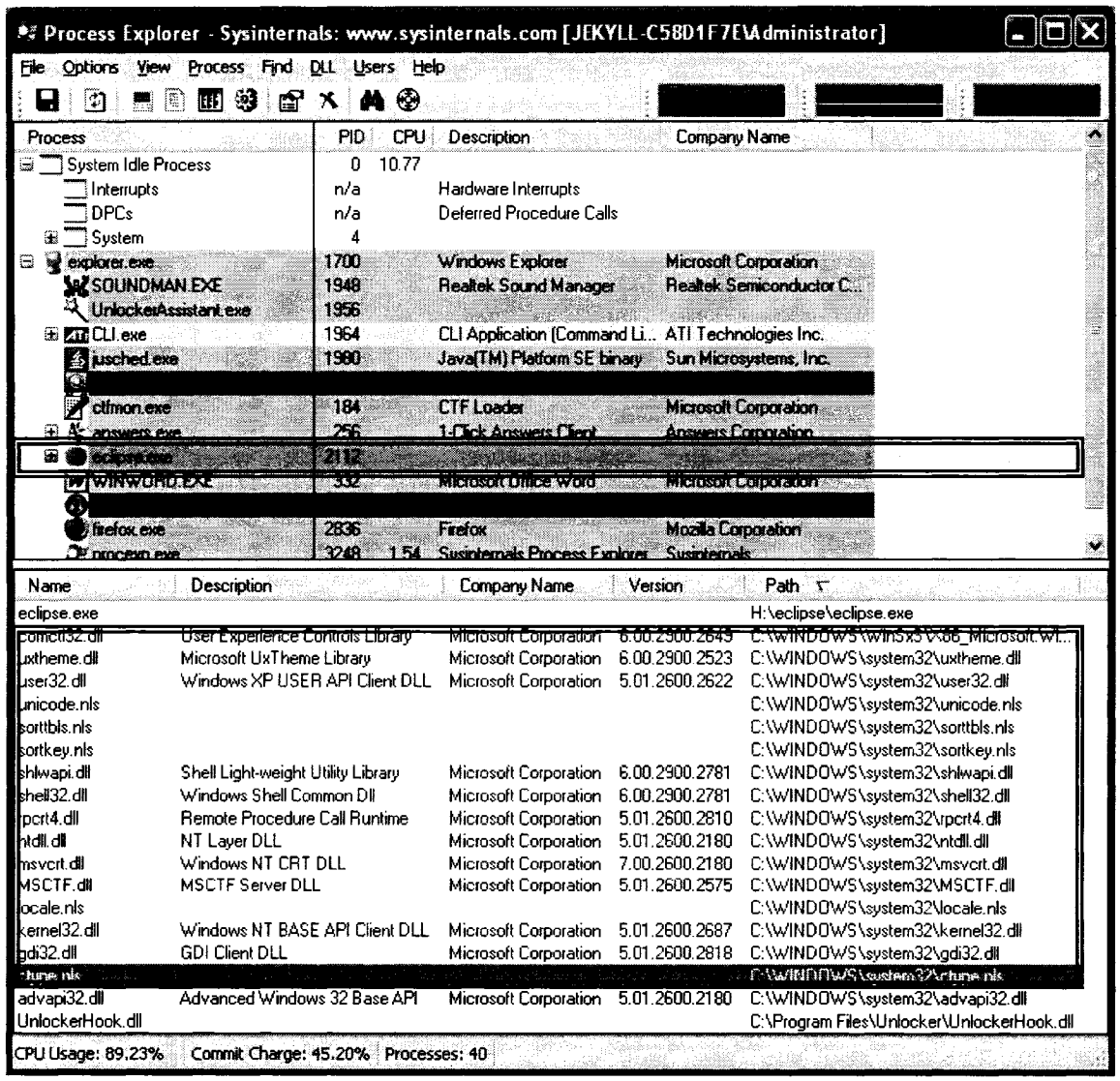


Figure 2.10 ProcessExplorer : Fichiers chargés par Eclipse

Process Explorer est un logiciel de SysInternals développé pour Windows nous permet de visualiser la configuration logicielle courante de notre système. De plus, chaque logiciel se voit associer la liste des fichiers qu'il a chargés.

Dans notre exemple, intéressons-nous aux fichiers chargés par Eclipse. Nous avons :

- le fichier exécutable principal eclipse.exe ;
- un fichier système comctl32.dll ;
- 15 fichiers systèmes de « \system32 », la librairie principale du système ;
- un fichier UnlockerHook.dll qui est une librairie du logiciel Unlocker [1]

Le cas d'Unlocker.dll est très intéressant. C'est une librairie dynamique appartenant à un logiciel UnlockerAssistant (présent dans la liste des logiciels). UnlockerAssistant cumule les facteurs de risque : c'est un utilitaire très peu répandu, non signé qui se permet en plus de faire charger une de ses dll par tous les autres logiciels. C'est exactement le genre de logiciels qui se verrait exclure dans un environnement strictement contrôlé. Un logiciel malveillant présentera probablement des caractéristiques similaires, les mauvais comportements en plus.

Si l'on garantit que ces fichiers déterminent le comportement d'Eclipse, on peut considérer qu'il n'y ait pas besoin d'informations sur les autres logiciels tournant sur la plate-forme. Un challenger voulant interagir avec Eclipse pourra demander le sous ensemble des mesures d'intégrité liées à Eclipse. ProcessExplorer nous fournit l'exemple idéal de la légère couche applicative qu'il faudrait adjoindre aux spécifications TCG pour permettre des rapports plus pertinents de configuration logicielle.

ProcessExplorer permet également de visualiser les processus lancés par Eclipse. Les fichiers chargés par ces processus sont tout aussi essentiels dans le comportement d'Eclipse et leurs hashs devraient donc être également rapportés.

---

[1] Unlocker est un utilitaire permettant de libérer un fichier du verrou posé par un logiciel. ProcessExplorer nous apprend que pour ce faire, il fait charger une de ses librairies UnlockerHook par tous les logiciels.

2.4.3 Configuration logicielle avancée

Entre la configuration totale et la configuration minimale, on peut réfléchir à un juste milieu. Quelque soit la précision de la configuration minimum, certains cas de figure ne seront pas couverts. En première ligne, les questions de bande passante. Il est notoire en effet qu'un logiciel P2P accapare assez égoïstement la connexion réseau. C'est même pour certains jeux en ligne la principale forme de tricherie. Dans le cadre MANET qui nous intéresse, on peut toujours s'attacher à mettre en œuvre la priorité absolue des capteurs dans toute communication réseau et à en rendre compte dans les mesures d'intégrité. A défaut de cela, il faut se poser la question des mesures d'intégrité complémentaires à la configuration minimum.

Les composantes logicielles d'une configuration n'ont pas forcément la même capacité de nuisance. Un filtrage pourrait donc être opéré sur les applications qui tournent pour obtenir des regroupements pertinents. On peut envisager un regroupement des applications suivant les ressources qui leur sont accessibles. On pourrait donc s'intéresser à toutes les applications ayant accès à certaines zones, pouvant envoyer des informations sur le réseau etc. Les droits protégés que permet le TPM pourraient être mis à contribution.

2.5 Conclusion

Nous faisons ici l'hypothèse, sinon le constat, que dans un MANET l'intrusion de nœuds mal intentionnés et la non-coopération de nœuds « légitimes » sont inéluctables. A défaut de pouvoir empêcher toute occurrence de tels événements, il faut se doter de systèmes pour les détecter et y répondre promptement. Dans ce chapitre, nous avons identifié les modèles de réputation comme étant particulièrement adaptés, en complément d'éléments de cryptographie, pour adresser des problèmes de sécurité et de coopération dans les MANET.

La réputation des nœuds reste toutefois problématique dans des environnements aussi étendus et dynamiques que les VANET. La construction de la réputation d'un nœud devient un processus très aléatoire. Typiquement, d'un carrefour à un autre, un

nœud pourra se refaire une toute nouvelle réputation auprès de ses tout nouveaux voisins. Même à l'échelle d'une ville, un tel mécanisme pourrait ne jamais converger d'autant que des problèmes de changement d'identité subsistent.

Les travaux du TCG apportent leur contribution à deux niveaux. D'abord, le déploiement d'un TPM de sa fabrication à la visibilité de sa plate-forme mère dans un réseau suit un processus d'attestations successives impliquant des infrastructures de gestion de clés qui permettent une gestion stricte des identités. De plus, les rapports des mesures d'intégrité permettent d'envisager la « réputation » sous un angle nouveau. Le prochain chapitre présente notre modèle basé sur la réputation des composants logiciels.

CHAPITRE III

RÉPUTATION DE COMPOSANTS LOGICIELS

Avec l'informatique de confiance de TCG, s'ouvrent de toutes nouvelles perspectives pour les modèles de confiance dans les réseaux ad hoc. Les entités interagissant dans un réseau gagnent en transparence et reprennent leur dimension de systèmes complexes. Cette transparence nouvelle sur leurs composants logiciels actifs permet d'envisager un modèle de confiance basé sur la fiabilité de ces composants logiciels. Dans un environnement ouvert et peu maîtrisé, nous proposons une approche de réputation pour les composants logiciels qui font la configuration d'un nœud. Notre démarche est originale et nous ne prétendons pas en avoir fait le tour. Outre la présentation du modèle résultant des choix que nous avons effectués, nous nous attacherons donc à donner une vision d'ensemble de ce que pourrait être un modèle plus complet. Nous commencerons par présenter les réflexions préalables à la conception de notre modèle. Le modèle et les algorithmes développés seront présentés de manière détaillée.

3.1 Préalable

Tirer le meilleur parti des standards TCG pour résoudre des problèmes de validation de l'information circulant dans des environnements ad hoc : telle était l'optique de départ de notre projet. Dans un environnement maîtrisé, la question est vite réglée : l'ensemble des composants logiciels est connu, restreint et les mises à jour sont centralisées. Pour atteindre des objectifs de souplesse et prendre en compte des systèmes à grande échelle, les modèles de réputation sont apparus très prometteurs en tant que techniques approximatives de bon niveau. Dans l'élaboration, sur cette base, d'un modèle compatible avec nos objectifs, il convenait de répondre à un certain nombre de questions essentielles pour déterminer les hypothèses à retenir. Il s'agit ici de réfléchir sur l'utilisation des standards TCG dans la problématique qui nous occupe, ceci en

termes d'apports et éventuellement d'ajustements nécessaires pour intégrer ces spécifications.

3.1.1 TCG et les réseaux ad-hoc

Les documents de spécification du TCG proposent des mécanismes - notamment d'attestation - plus aisément transposables dans des environnements centralisés. Ceci pose un problème pour l'utilisation des TPM dans des réseaux ad hoc où il n'existe aucune entité préétablie et centrale chargée de valider les informations provenant des nœuds. Lorsqu'on reprend le double concept d'identité-intégrité de TCG, on peut relever un certain nombre de points à éclaircir au niveau des preuves d'identité et d'intégrité dans un contexte décentralisé.

Pour ce qui est de l'identité, TCG propose pour les TPM et les plates-formes les contenant un système de certificats inspiré des PKI. Dans le schéma actuel de TCG, les crédentiels AIK des plates-formes contenant un TPM sont délivrés par un CA. Nous considérons que le nombre de CA est suffisamment réduit pour qu'on puisse stocker au niveau de chaque entité mobile toutes les informations nécessaires (clés publiques de ces CA) à la validation des preuves d'identité fournies par une autre entité.

Le problème des preuves d'intégrité est au cœur de notre réflexion. Le protocole de rapport d'intégrité de TCG permet de garantir, sous réserve d'un TPM valide, la conformité des informations qu'un nœud fournit sur sa configuration courante. Nous avons fait dans la section 2.4 le point sur ces informations à fournir. Toutefois, l'interprétation de ces données reste un problème dans des environnements ouverts. C'est à ce niveau que nous comptons faire intervenir les modèles de réputation.

3.1.2 TCG et les modèles de réputation

Grâce aux TPM, on dispose d'un mécanisme qui garantit qu'un nœud rapportera sans altération sa configuration logicielle courante. Dans l'hypothèse où l'on n'exclut pas d'office les logiciels « non-signés », issus par exemple de la communauté du logiciel libre, on peut raisonnablement supposer qu'un seul nœud ne peut pas connaître dès le départ toutes les valeurs valides des mesures d'intégrité. L'utilisation de modèles

d'approximation plus dynamiques tels que les systèmes de réputation apparaît comme étant une option intéressante. D'autre part, il n'est pas impossible que pour une application donnée, il puisse y avoir occurrence de conflits graves avec d'autres applications parfaitement intègres. Une simple connaissance figée des domaines de validité des applications en cause ne pourrait pas démêler ce genre de situations. Dans ce contexte, l'utilisation des modèles de réputation est pertinente dans le sens où l'information de réputation se construit sur l'expérience et non sur des considérations théoriques potentiellement incomplètes ou incorrectes. L'originalité de notre approche sera ici de s'intéresser à la réputation des configurations logicielles plutôt qu'à celle des nœuds. Ce choix se justifie aisément dans le cadre des capteurs, où il n'y a pas implication des usagers. La configuration logicielle permet donc de prévoir le comportement du nœud. Une telle approche demande qu'on la confronte à certaines considérations essentielles dans la construction d'un modèle de réputation.

Les métriques de réputation

Les configurations logicielles des nœuds seront en fait des combinaisons de différents éléments logiciels. Un même nœud ne gardera certainement pas la même configuration sur une longue période. Les combinaisons d'éléments logiciels sont donc potentiellement infinies et il serait peu efficace de maintenir des informations de réputation sur chacune de ces combinaisons. Les composants logiciels sont en eux-mêmes bien plus intéressants. Après chaque interaction avec un nœud donné N, l'issue - bonne ou mauvaise - sera endossée par chacun des composants logiciels de N. Un nœud se constitue ainsi, avec ses propres expériences, de l'information de première main. Mais cette information demande à être complétée par les opinions des autres nœuds de l'environnement. La manière dont sont diffusées et reçues les opinions des nœuds mérite qu'on s'y attarde. Deux modes de fonctionnement sont envisageables.

- Mode réactif :

Un nœud peut demander à ses « nœuds voisins » leur « opinion » sur la configuration logicielle (ou des éléments constituants) d'un autre nœud. Les nœuds

sollicités renvoient une réponse en même temps que leurs configurations actuelles. Les opinions envoyées sont pondérées suivant la fiabilité des configurations qui les accompagnent. Le nœud client confronte alors sa propre opinion à celles qui sont envoyées. De cette analyse découle alors sa décision de « faire confiance ou non » au nœud considéré.

- Mode proactif :

C'est un mode qui semble plus pertinent dans notre démarche. Les échanges d'information de réputation pourraient se faire automatiquement entre les nœuds selon une certaine périodicité. Un nœud pourra ainsi compléter ses informations sur un composant logiciel donné ou sur l'ensemble de sa base de données de réputation.

Les apports d'informations des autres nœuds de l'environnement devront être compilés avec discernement. La confiance accordée à ces compléments d'information sera d'autant plus grande qu'il y aura une certaine compatibilité entre l'expérience propre du nœud et celles rapportées. On peut également envisager la présence de nœuds spécialisés à très haut niveau de fiabilité qui se chargeront de propager de l'information utile sur les configurations observables.

Stockage et intégrité de l'information de réputation

Stockage et intégrité de l'information de réputation sont deux points fortement liés. Dans un contexte centralisé, le stockage sécurisé de l'information est une problématique hors du champ d'un modèle de réputation puisqu'elle est liée au niveau de sécurité de l'entité centrale. Par contre, dans un contexte décentralisé, le stockage de l'information de réputation pose surtout le problème de la distribution de cette information. Avec une réputation basée sur la configuration logicielle des nœuds, les données de réputation sont, de fait, distribuées puisque la configuration logicielle d'un nœud est faite d'éléments divers potentiellement présents sur d'autres nœuds.

Le changement d'identité

TCG propose l'intégration d'un TPM aux plates-formes et ce TPM contient un Endorsement Key (EK) unique qui permet de l'identifier sans faille. Des problématiques

de respect de la vie privée entrent tout de même en ligne de compte qui font que le recours direct à cet EK est loin d'être certain. Mais même dans le cas où l'on se reporte sur les alias AIK (Attestation Identity Keys) délivrés aux plates-formes, on peut raisonnablement préjuger qu'ils auront un certain coût, de quelque nature qu'il soit - typiquement monétaire. On n'est donc plus dans un cadre de « cheap pseudonyms » selon l'expression de Friedman [4].

Reste une inquiétude légitime sur les composants logiciels. Il convient de décourager là aussi les « changements d'identité ». Concrètement, il s'agit de gérer la méfiance des nœuds par rapport à un nouveau composant logiciel. Au déploiement du système, il faut bien que les nœuds puissent laisser le bénéfice du doute aux nouveaux éléments puisqu'ils seront majoritaires. Par contre, une certaine méfiance devra s'installer et s'accroitre au fur et à mesure que le système évolue. Les nœuds contenant de « nouveaux » composants devront prouver leur coopération sans faille avant de pouvoir bénéficier des services des autres.

3.1.3 La proposition

Nous proposons d'implémenter au niveau des nœuds du réseau un modèle de confiance, basé sur la réputation des configurations rapportées, qui opère par mesures statistiques sur les composants logiciels des nœuds rencontrés. Notre modèle suppose l'impossibilité pratique d'avoir au niveau de chaque nœud les mesures d'intégrité de l'ensemble des logiciels. Notre approche est celle d'un modèle de réputation ; elle implique des expériences - positives ou négatives - avec les constituants des configurations rencontrées. Comme nous l'exposerons plus loin, on pourra tirer des particularités de l'environnement considéré pour arriver à « trier le bon grain de l'ivraie ». L'idée est de dégager au fur et à mesure de l'évolution du système des éléments stables et sains pour ensuite traiter - et non interdire - les « originalités ». Le nombre de configurations à gérer dans le détail sera donc beaucoup moins élevé qu'une première observation ne laisse envisager.

3.2 Hypothèses du modèle

Nous considérons un environnement ad hoc avec des nœuds nombreux et très mobiles. Les transactions s'effectuent entre logiciels (ou progiciels) potentiellement différents mais qui échangent sur la base d'une ontologie commune des informations sur leur environnement. Les communications directes sont établies avec des nœuds se trouvant dans une certaine aire ou un certain domaine. Les nœuds s'échangent également des informations fiables sur leur configuration interne. Nous avons retenu les travaux du TCG pour assurer ce volet mais toute autre technologie pouvant nous garantir cette assertion conviendrait. Chaque nœud aura une base de données initiale le renseignant sur un domaine réduit de l'ensemble des composants logiciels auxquels il sera confronté. Ces connaissances initiales contiennent au minimum les données d'intégrité de chacun des systèmes d'exploitation de l'environnement. Ces mesures affectent la crédibilité des mesures reportées. Ceci est donc une condition essentielle à la validité de notre modèle de réputation. Tout nœud qui ne pourra prouver qu'il possède un système de report d'intégrité non corrompu sera tout simplement ignoré.

Généralement, dans les MANET, la configuration logicielle des nœuds (tout au moins ceux qui sont sains) n'est pas très dynamique. Cela n'exclut pas de possibles infections par des malwares ou des virus ou tout simplement l'installation de logiciels favorisant les comportements égoïstes. Nous supposons que la majorité des nœuds mobiles ne présentent pas de comportements malicieux. Un comportement malicieux pourra être associé à un ou plusieurs composants logiciels (malwares) dans la configuration du nœud mobile.

Évaluation de l'interaction

Chaque interaction entre nœuds devra être évaluée par un mécanisme non inclus dans notre modèle. La notion d'interaction est ici générique et désigne :

- l'échange d'informations sur l'environnement ;
- la coopération à l'acheminement de l'information.

Chaque interaction est évaluée de manière binaire 1 ou 0, interaction satisfaisante ou non. Il n'y a pas de niveaux intermédiaires. Nous supposons que le nœud dispose de mécanismes pour évaluer l'interaction. Pour valider une information sur l'environnement, nous considérons que la diversité des sources d'information et la propre perception du nœud sont des pistes intéressantes pour noter ce type d'interaction. Les mauvais comportements liés au routage sont détectables par de nombreux mécanismes déjà proposés dans la littérature (comme Watchdog [9]). Dans tous les cas, notre modèle pose comme hypothèse que nous disposons de mécanismes pour noter chaque interaction. Ces mécanismes auront un certain coût (temps, ressources, etc...) à minimiser, en plus d'être imparfaits. On aura ainsi des faux négatifs (comportements délictueux non détectés) I-FN et des faux positifs (défaillance temporaire d'un nœud correct ou comportement légitime mal jugé) I-FP. Ces évaluations seront utilisées pour déterminer la réputation des configurations logicielles présentées.

Le module d'évaluation d'une interaction est dans notre modèle un module abstrait représenté par un I-FP et un I-FN et régi par l'algorithme ci-dessous

Si un nœud N est infecté,
selon une probabilité de I-FN,
évaluation (interaction de N) = 1
Sinon
selon une probabilité de I-FP,
évaluation (interaction de N) = 0
Fin si

3.3 Architecture générale

Nous proposons au niveau de chaque nœud une infrastructure modulaire pour gérer les différents aspects de notre modèle. Dans les sous-sections suivantes, nous allons donner une vue d'ensemble de cette architecture avant de nous intéresser en détail à chacun des modules définis.

3.3.1 Vue d'ensemble

Au niveau de chaque nœud, le modèle sera décomposé en quatre modules :

- La Base des données de réputation

Notre base de données contient des statistiques sur les différents composants logiciels rencontrés. Elle stocke les données de réputation brutes sur les expériences personnelles du nœud (à communiquer aux voisins) mais aussi les données de réputation rapportées par les autres nœuds de l'environnement. C'est la « matière première » des prises de décision.

- Un module Reporter

Il se charge de diffuser un condensé des interactions passées du nœud auprès des autres nœuds. Chaque diffusion sera une part significative de la base de données des informations de réputation. Les informations rapportées seront donc celles jugées les plus efficaces à l'évolution du système, soit les pires réputations, les meilleures, les plus à jour etc.

- Un module Discovery

Ce module peut être actif et rechercher les rapports de réputation, ou être plus passif et les attendre. Dans tous les cas, le module Discovery reçoit les rapports de réputation des autres nœuds. Il peut les stocker un certain temps avant de les faire suivre, en temps opportun, au module Réputation qui décidera, selon la réputation des configurations des nœuds rapporteurs, de les intégrer ou non à la base de données. Des mécanismes de pondération pourront également être mis en œuvre pour conserver la priorité à l'expérience « personnelle » du nœud. Une autre option est de maintenir séparées la base de données des transactions effectuées et celle des transactions rapportées.

- Un module Réputation

Il évalue les configurations qui lui sont soumises, qu'elles proviennent de nœuds voulant engager une transaction ou de nœuds voulant rapporter leur expérience. Il retrouve dans la base de données les informations de réputation des composants logiciels envoyés, calcule la réputation de l'ensemble de la configuration et décide de l'issue à donner (interactions rejetées ou non, rapports de réputation acceptés ou non). Dans le cas de l'évaluation des configurations de nœuds rapportant leur expérience, un seuil moins

strict pourra être retenu mais les poids accordés seront fonction de la réputation de la configuration du nœud rapporteur. Il peut être intéressant - mais trop complexe - de prendre également en compte les rapports de configurations reconnues corrompues pour en tirer de l'information.

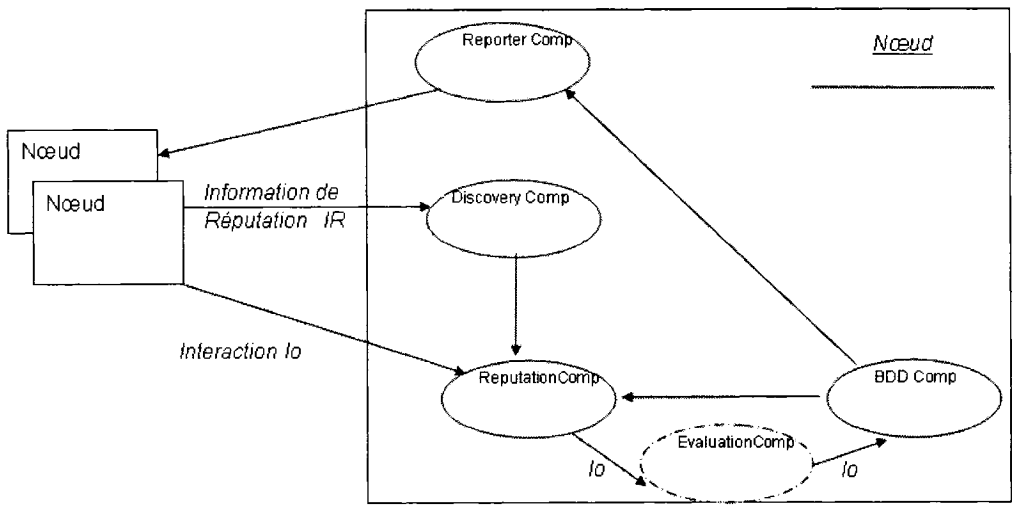


Figure 3.1 Architecture logicielle au niveau de chaque nœud

- Algorithme général au niveau de chaque nœud

Tâches en arrière plan

Diffuser les « plus importantes » informations de réputation aux autres pairs

Ecouter, stocker et intégrer à la base de données (sous réserve de crédibilité) les diffusions des autres pairs

Occurrence d'une interaction avec le pair P

1. Obtenir la Configuration de P ;
2. Si les mesures d'intégrité du noyau sont inacceptables, rejeter l'interaction ;
3. Parcourir la base de données des composants logiciels ;
4. Si Présence de composants logiciels inconnus, appliquer au fil de l'évolution du système, de plus en plus lourdement, des « droits d'entrée » (selon le contexte)
5. Sinon, Evaluer la configuration
   - Si Présence de composants bannis alors abandonner l'interaction sinon l'accepter

6. Traiter et évaluer l'interaction ;
7. Mettre à jour la base de données ;

Dans un contexte multi-sources, les composants logiciels inconnus intégreront la base de données si l'information envoyée depuis leurs nœuds concorde avec celles de configurations déjà reconnues comme dignes de confiance.

3.3.2 Base de données de réputation

Nous considérons la configuration d'un nœud comme étant une table de hachage dont les clés sont les composants logiciels (les fichiers) et les valeurs, les versions (hashs) de ces composants (fichiers).

Tableau 3.1 Illustration d'une configuration logicielle

Win32.dll	Kernel32.dll	Eclipse.exe	...	Unlocker.dll
A511233	VGVG453	VJOJYGD	...	YUKOJOU

Chaque élément représente un composant logiciel, typiquement un fichier représenté par son nom (ou son chemin d'accès). Dans ce qui va suivre, nous utiliserons *composant logiciel* pour désigner indistinctement un fichier et une version spécifique de ce fichier (représentée par le hash de son contenu).

Evaluation d'un composant logiciel (SC_i)

Nous supposons que la configuration contient suffisamment d'information pour y trouver les composants logiciels responsables des mauvais comportements. Dans une configuration, il y a possiblement plusieurs fichiers corrompus et toutes sortes de corrélations (logiciels « sains » mais contenant des trous de sécurité, remplacement de plusieurs fichiers clé etc..). Quoiqu'il en soit, en maintenant des statistiques séparées sur chaque composant logiciel, on pourra isoler les composants corrompus, qui devraient se distinguer par de pires statistiques. Ce que nous argumentons ici, c'est le fait que nous n'avons pas à maintenir des statistiques sur l'ensemble (potentiellement infini) des

combinaisons de versions de fichiers rencontrés dans l'environnement. Les statistiques de chaque composant logiciel suffiront à le classer comme « bon » ou « mauvais ». Toute combinaison contenant un « mauvais » composant logiciel sera considérée comme « mauvaise ». Chaque nœud mobile stocke des statistiques pour chaque composant logiciel dans une base de données. Trois compteurs de base sont utilisés : le nombre de rencontres, le nombre de rejets, le nombre de problèmes.

Stockage des informations de réputation des composants logiciels

Nous stockons les expériences passées d'un nœud en construisant des statistiques sur les composants logiciels rencontrés. Notre structure hiérarchique illustrée à la Figure 3.2 distingue à un *premier* niveau les systèmes d'exploitation des nœuds. Lorsque l'on considère une configuration minimale, on aura là un logiciel de gestion des capteurs.

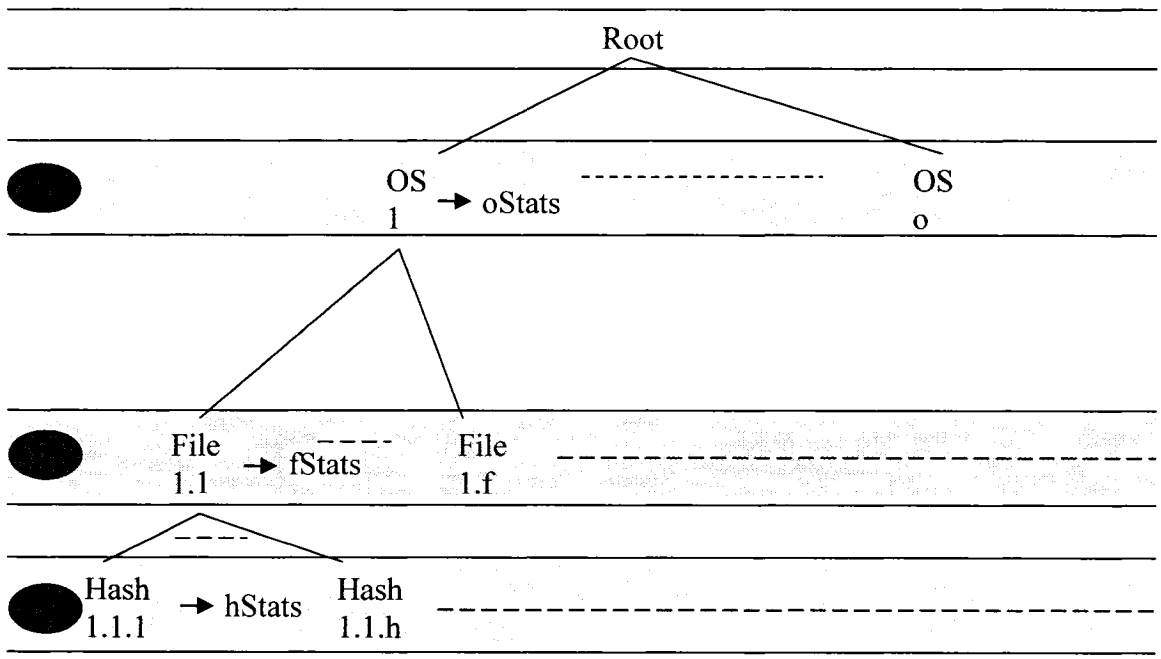


Figure 3.2 Graphe de stockage des composants logiciels rencontrés

Pour chaque niveau de 1 à 3, on stocke les statistiques suivantes :

- nombre de rencontres
- nombre de problèmes détectés
- pourcentage de rejets

Une « rencontre » d'un composant logiciel est une interaction avec un nœud ayant une configuration contenant ce composant logiciel.

Un « problème » pour un composant logiciel est une interaction avec un nœud ayant une configuration contenant ce composant logiciel et dont l'issue a été jugée mauvaise (par le mécanisme de détection des mauvais comportements). On a ici un jugement a posteriori (après l'interaction).

Un « rejet » pour un composant logiciel est une rencontre avec un nœud ayant une configuration jugée corrompue (par notre système) contenant ce composant logiciel. On a ici un jugement a priori (avant l'interaction).

Par la suite, dans l'évaluation d'un composant logiciel, « problèmes » et « rejets » auront la même pondération. Mais nous jugeons bon de les stocker séparément pour d'éventuels raffinements ultérieurs.

Un composant logiciel (SC_i) est donc évalué selon les interactions auxquelles il participe. Supposons un nœud N1 de configuration C1 établissant une interaction avec un nœud N2 de configuration C2.

Trois scénarios sont possibles :

1) C2 est rejetée par notre système de réputation au niveau de N1. Pour chaque composant logiciel (Fichier F, Hash H) de C2, le nombre de rejets est incrémenté de 1 à la fois pour F et H. Même opération pour le nombre de rencontres

F		Rejets (+1)	Problèmes	Rencontres (+1)
	H	Rejets (+1)	Problèmes	Rencontres (+1)
	…	…	…	…

2) C2 n'est pas rejetée par N1. L'interaction a lieu et le comportement de N2 est noté par le mécanisme de détection des mauvais comportements.

2a) Le comportement de N2 est jugé mauvais sur cette interaction

F		Rejets	Problèmes (+1)	Rencontres (+1)
	H	Rejets	Problèmes (+1)	Rencontres (+1)
	…	…	…	…

Pour chaque composant logiciel (Fichier F, Hash H) de C2, le nombre de problèmes est incrémenté de 1 à la fois pour F et H. Même opération pour le nombre de rencontres.

2b) Le comportement de N2 est jugé bon sur cette interaction

F		Rejets	Problèmes	Rencontres (+1)
	H	Rejets	Problèmes	Rencontres (+1)
	…	…	…	…

Pour chaque niveau de 1 à 3, intervient une variable représentant l'évaluation de l'OS, du logiciel, du fichier ou du hash. Pour un composant logiciel SC_i, on a donc

$$Evaluation\ (SC_i) = -\frac{problemes + rejets}{rencontres}$$

on a donc $évaluation(SC_i) \in [0,-1]$

Les niveaux supérieurs 1 et 2 rajoutent à ces compteurs des informations sur les niveaux inférieurs dire.

On rajoute *pour chaque fichier,* des statistiques relatives à ses hashs

- nombre de hashs

On rajoute *pour chaque OS*, des statistiques relatives à ses fichiers et à ses hashs

- nombre de fichiers
- moyenne et déviation standard des évaluations des fichiers
- nombre de hashs
- moyenne et déviation standard des évaluations des hashs

L'évaluation directe d'un composant logiciel est donc un réel entre 0 (aucune « mauvaise » interaction avec une configuration contenant SC_i) et -1 (uniquement des « mauvaises » interactions avec les configurations contenant SC_i).

Le mécanisme de « Fading » proposé

L'évaluation des composants logiciels accorde une priorité aux interactions les plus récentes. Les interactions sont considérées par blocs de N (N entier), le dernier bloc de N interactions comptant pour la moitié de tous les autres blocs de N précédents. Ainsi, pour chaque SC_i, on opère comme illustré à la Figure 3.3 une réduction à chaque 2N sur tous les compteurs des données de réputation.

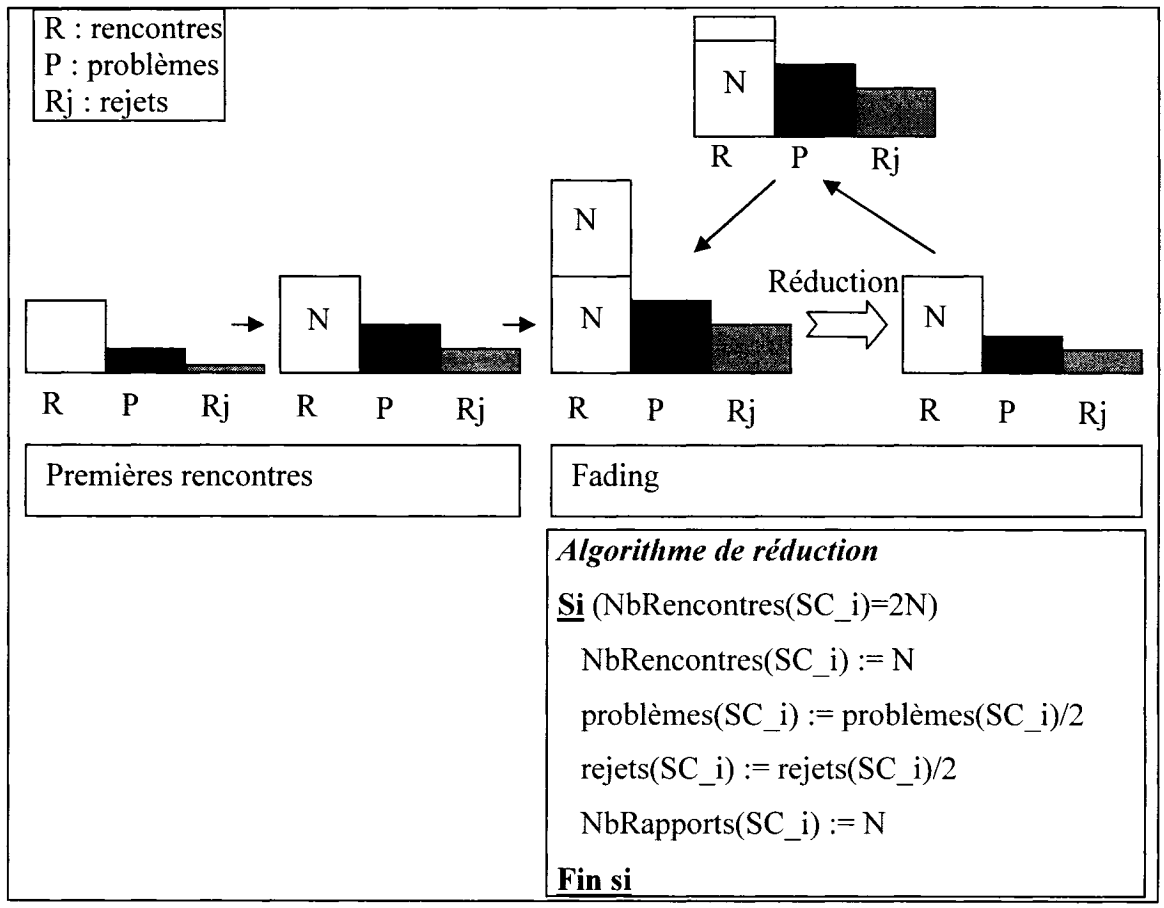


Figure 3.3 Mécanisme de « Fading » proposé

3.3.3 Module Reporter

Le module Reporter se charge de rapporter aux voisins les évaluations des composants logiciels que le nœud a rencontrés. Plusieurs modalités sont possibles :

- rapporter les évaluations des pires composants ;
- rapporter les évaluations des composants récemment rencontrés ;
- rapporter les évaluations des composants les plus rarement rencontrés ;
- etc.

Nous avons fait le choix de rapporter les évaluations des composants rencontrés au cours des R dernières interactions. Ce point fait très certainement partie des travaux futurs, dans le sens où ce choix demande à être validé plus précisément.

3.3.4 Module Discovery

Ce module reçoit les évaluations rapportées par les modules Reporter des autres nœuds. Ce module pourrait être doté d'une mini-structure de stockage pour mettre en attente de validation les rapports d'évaluation. De plus, un tel mécanisme pourra permettre d'éviter la prise en compte dans une courte période de rapports provenant du même nœud.

Dans notre simulation, nous nous sommes contentés de conditionner les échanges d'informations de réputation entre nœuds à une relation de confiance réciproque dans leurs configurations i.e., les nœuds peuvent interagir.

Lorsqu'un nœud reçoit un rapport de réputation sur un composant SC_i, un compteur des rapports d'évaluation est incrémenté et la valeur d'évaluation rapportée est mise à jour. L'évaluation pour un composant logiciel SC_i est la moyenne des réputations rapportées par les nœuds dignes de confiance précédemment rencontrés.

SC_i	Rapports (+1)	Moyenne des Evaluations Rapportées

3.3.5 Module Réputation

En s'appuyant sur les réputations des composants logiciels de la configuration d'un nœud voisin quelconque, ce module construit la réputation de cette configuration. Dans notre modèle, au niveau d'un nœud N, la réputation d'un fichier F peut être différente de celle de sa version H (F=H) s'il y a plusieurs versions de F. Dans le cas où il n'y a pas suffisamment d'information sur une version nouvelle, les informations collectées sur le fichier sont utilisées. Ceci sera surtout déterminant dans le cas où F aura été rangé dans la liste noire des composants logiciels.

Sigma réputation

La réputation pour un composant logiciel est définie comme une somme pondérée de l'évaluation directe et de l'évaluation rapportée de ce composant logiciel.

Introduisons ici ce que nous définissons comme étant la quantité d'informations (*Quantité_Information*) sur un composant logiciel SC_i. Nous calculons cette valeur en utilisant le nombre de rencontres (*NbRencontres*) et le nombre de rapports (*NbRapports*). La formule proposée est la suivante :

$$Quantité_Information(SC_i) = NbRencontres(SC_i) + \alpha * NbRapports(SC_i)$$

Nous considérons que la quantité d'information apportée par un rapport de réputation est α fois égale à celle apportée par une simple rencontre où $\alpha \geq 1$. Nous choisissons cette option puisqu'un rapport est supposé être une opinion basée sur au moins une rencontre directe et probablement plus. Par la suite, nous avons fixé $\alpha = 2$.

De cette formule, nous dérivons les poids des rencontres directes (*d_weight*) et des expériences rapportées (*r_weight*).

$$d_weight(SC_i) = \frac{Nb\ Rencontres(SC_i)}{Quantité_Information(SC_i)}$$

$$r_weight(SC_i) = 1 - d_weight(SC_i)$$

La réputation finale d'un composant logiciel est une combinaison linéaire de ses évaluations directe (*ED*) et rapportée (*ER*). Le poids de l'évaluation rapportée est d'au moins ½ et croît d'autant qu'il y a plus de rapports que de rencontres. Évaluation directe et évaluation rapportée ont initialement le même poids, ½. Mais si le poids *d_weight* passe en dessous de ½, il est alors utilisé avec *r_weight* comme poids courants de leurs évaluations respectives.

Si $(d_weight(\text{SC_i}) < \frac{1}{2})$

$$R(\text{SC_i}) := d_weight(\text{SC_i}) * ED(\text{SC_i}) + r_weight(\text{SC_i}) * ER(\text{SC_i})$$

sinon

$$R(\text{SC_i}) := \frac{ED(SC_i) + ER(SC_i)}{2}$$

fsi

La réputation ainsi calculée est utilisée pour classifier les composants logiciels. Cette valeur est comparée à un seuil fixé ou induit par les données (en utilisant la moyenne et la déviation standard de l'ensemble des réputations des composants logiciels) pour révéler les composants les plus « suspects ».

Seuil d'information

Pour chaque composant logiciel, il faut une quantité d'information minimale nécessaire pour qu'un nœud soit capable de prendre une décision sur la confiance à lui accorder ou non. Grâce à la propagation de réputation entre nœuds qui se font confiance, ce seuil peut être atteint plus vite. Mais le choix de ce seuil est difficile : un seuil trop haut ralentit l'évolution du système (beaucoup de faux négatifs *C-FN* en attendant de pouvoir prendre une décision) tandis qu'un seuil trop bas conduira plus vraisemblablement a des décisions hâtives et erronées (beaucoup de faux positifs *C-FP*).

Supposons que nous choisissions un seuil de 100 pour la quantité d'information (calculée comme présentée précédemment). Au bout de 100, un composant logiciel ayant une mauvaise réputation (mettons inférieure à -2/3) sera banni. Mais atteindre une

quantité d'information de 100 prend du temps. C'est la raison pour laquelle nous avons introduit un système de pénalités.

Le système de pénalités

L'idée ici est d'implémenter un mécanisme simple pour accélérer la détection des mauvaises configurations. Tout le long de ce projet, nous avons essayé de concilier l' « absolu » et le « relatif ». Les compteurs (rencontres, rejets, problèmes) assurent le pendant absolu de notre évaluation. On a au final des évaluations de chaque composant logiciel sans aucune information sur leur situation dans les différentes combinaisons rencontrées. Ceci nous a conduit à ce système de pénalités, le pendant relatif de notre modèle. Pour chaque interaction qui se passe mal, nous attribuons une pénalité au composant logiciel – appartenant à la configuration présentée – classé comme « suspect » et ayant la pire réputation. Nous pouvons ici faire une analogie avec une classe où les étudiants ont une évaluation numérique qui débouche sur une évaluation par lettres : A, B, C etc. La différence est que nous nous intéresserons uniquement au pire étudiant, le dernier des F. Nous procédons comme suit :

- *Composant suspect : Seuil de Réputation*

Pour chaque OS (niveau 1) de l'environnement logiciel d'un nœud, nous calculons incrémentalement au niveau de chaque nœud, la moyenne (*meanRD*) et la déviation standard (*stdRD*) des évaluations directes des composants logiciels de l'OS. Tout composant logiciel d'une configuration C envoyée qui a une réputation inférieure à (*meanRD-stdRD*) peut raisonnablement être considéré comme suspect. A ce seuil dynamique, nous ajoutons un seuil statique de -2/3 car nous estimons qu'indépendamment des circonstances, un composant qu'on retrouve 2 fois sur 3 dans des interactions malignes est un suspect crédible. Lorsqu'une interaction avec un nœud N présentant une configuration C se passe mal, tout composant logiciel suspect pourra subir des pénalités.

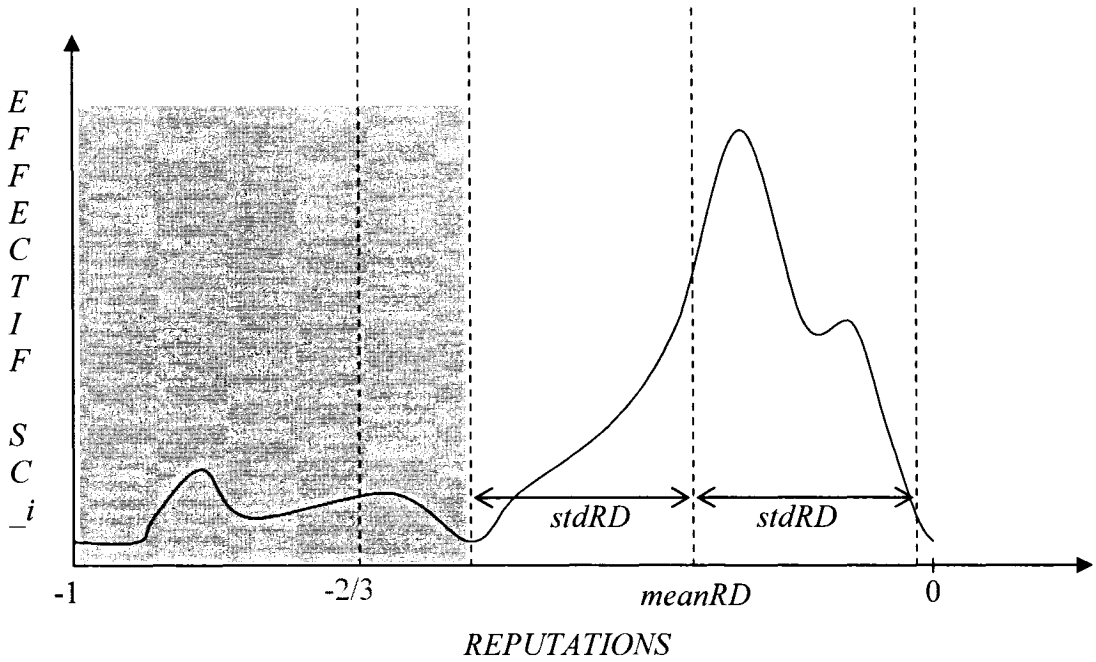


Figure 3.4 Distribution des évaluations et composants logiciels suspects

R(SC_i) < maximum (seuil_statique, meanRD - stdRD)

⇨ *SC est un Composant SUSPECT*

- *Penalités*

Il s'agit ici, à défaut de preuves irréfutables, d'accumuler des indices concordants pour confondre un composant de logiciel malicieux. Qu'un composant logiciel légitime se retrouve dans une configuration infectée sera fréquemment observé. Dès lors que ce composant présente une très mauvaise réputation et que l'interaction se passe mal, il devient plus probable que cet élément « y soit pour quelque chose ». A plus forte raison, si ce composant suspect est celui qui présente la pire réputation, nous lui attribuons une pénalité. A chaque mauvaise interaction, nous utilisons la formule suivante.

$$Penalty = max\left(min_penalty, \frac{max_penalty}{NbSuspects}\right)$$

NbSuspects étant le nombre de composants logiciels suspects dans la configuration.

Un composant logiciel reçoit d'autant plus de pénalités que le nombre de suspects est réduit. En particulier, si ce composant est le seul suspect dans une configuration, on lui applique une pénalité maximale *max_penalty*. A chaque mauvaise interaction, la pénalité attribuée est ajoutée aux éventuelles pénalités déjà infligées à ce composant.

$$Penalty(SC_i) := Penalty(SC_i) + Penalty$$

Au bout d'une certaine valeur de pénalité *thresholdPenalty*, le composant logiciel est banni. La pénalité maximale sera choisie inférieure à *thresholdPenalty* par prudence et pour prendre en compte les faux positifs dans l'évaluation d'une interaction (I-FP). Dans le même ordre d'idées, les pénalités d'un SC_i sont remises à 0 sur la même période que celle utilisée pour le mécanisme de fading. Cette période est suffisamment élevée pour qu'un composant logiciel malsain soit détecté dans cet intervalle. On pourrait craindre que des composants malsains ne cherchent à adapter leur stratégie pour tirer avantage de cette amnistie mais il leur faudrait jouer aux devinettes sur le moment de cette amnistie. Dans les faits, seules les opinions d'un nœud sont transmises. Les autres nœuds n'ont aucun moyen de savoir où un nœud donné en est par rapport à sa période de « fading » pour un composant donné. Si ce composant logiciel se retrouve au niveau de plusieurs nœuds, cela renforce la difficulté pour estimer le nombre de rencontres qu'un nœud donné a pu avoir avec ce composant. On a là un aléa qui dessert sérieusement les logiciels malveillants.

Soulignons également que, pour pouvoir appliquer une pénalité à un composant logiciel dans une configuration, nous devons avoir de l'information même minimale sur tous les composants logiciels (fichiers ou hashs) de cette configuration. Pour appliquer une pénalité à un fichier (hash), nous devons avoir une valeur de réputation pour tous les autres fichiers (hashs) présents dans la configuration.

Classification des composants logiciels

Les composants logiciels sont classifiés comme « sains » ou « malsains ». A cause des faux positifs I-FP et des faux négatifs I-FN du mécanisme de détection, la

prudence est ici de mise. Il y a un bon compromis à trouver entre l'évolution du système et la précision de la classification.

Chaque nœud initialise sa propre liste verte (celle des éléments sains) avec les composants logiciels de sa propre configuration courante. Il reconnaît les fichiers qu'il a chargés comme étant évidemment des composants logiciels sains. Si un nœud A avec une configuration C(A) rencontre un nœud B avec une configuration C(B) tel que C(B) soit un sous ensemble de C(A), A n'a aucune raison de ne pas faire confiance à B. Le mécanisme de surveillance devient à ce moment inutile.

$$C(B) \subset C(A) \Rightarrow A \text{ fait confiance à } B$$

En revanche, à moins qu'en définitive C(A) ne soit égal à C(B), B ne fera confiance à A que si tous les éléments de C(A) – C(B) sont dans sa liste verte.

Classer un SC_i comme « sain »

La classification d'un composant logiciel comme « sain » est une décision extrêmement importante. On peut, au seuil de décision de la quantité d'information , reconnaitre comme « sain » un composant logiciel ayant une très bonne réputation. Le principal avantage d'une liste verte réside dans l'économie de la mise en œuvre du mécanisme de surveillance des interactions. On aurait ainsi un contrôle moins strict lorsqu'un nœud présente une configuration dont tous les composants sont classés comme sains. Ce point est intéressant mais nous limiterons tout de même le classement d'un composant logiciel comme « sain » aux seuls composants ayant été déjà chargés par le nœud.

Classer un SC_i comme « malsain »

Un composant logiciel malsain sera, suivant l'hypothèse où l'environnement est majoritairement sain, moins souvent rencontré. L'obtention de la quantité d'information nécessaire pour conclure à son propos risque d'être un processus long. La manière dont nous calculons la quantité d'information – en incluant les rapports de réputation – sera ici très utile. Mais même lorsqu'on considère que des composants logiciels se verront offrir une (mauvaise) publicité plus active, le seuil de quantité d'information ne sera

probablement pas très vite atteint. C'est pourquoi un composant logiciel sera classé malsain sur les considérations suivantes :

Si *SC_i* SUSPECT ***ET*** *penalty(SC_i)* ≥ *thresholdPenalty*

Alors *SC_i* BANNI

Les composants logiciels marginaux (plus rares) mais sains pourraient être perçus comme malsains si leurs rencontres coïncident souvent avec celles de configurations malsaines. Mais il est moins probable - quoique possible - que ces éléments apparaissent comme étant les pires dans une configuration infectée. L'approche avec les pénalités réduit les risques de voir notre modèle générer un pourcentage élevé de faux positifs C-FP mais accélère également l'identification des composants logiciels malsains puisque la quantité d'information nécessaire avant l'application des pénalités sera très réduite.

Patterns : tirer avantage des propriétés générales d'une configuration

En plus de classifier les composants logiciels, le module Réputation maintient une petite base de données sur les propriétés générales des configurations rencontrées. Le sous-module Patterns est principalement destiné à appliquer une politique adaptative pour gérer l'entrée dans le système de nouveaux composants. Nous adressons ici le cas de logiciels malicieux qui essaieraient de flouer le système en s'auto-répliquant avec des noms de fichiers différents et / ou en abusant du polymorphisme pour générer sans cesse de nouveaux composants logiciels.

Composants logiciels et changement d'identité

Une première approche pour décourager ces comportements est d'instaurer un périple initiatique potentiellement très lourd pour tout nouveau composant logiciel. Ainsi, un nœud ayant un composant logiciel inconnu d'un autre nœud ne pourrait lui demander des « services » (ici des informations sur l'environnement ou une demande de relais) qu'après l'avoir servi un certain temps. Le « bizutage » à mettre en œuvre est ici très lié au contexte considéré et nous ne saurions en préciser plus avant les modalités. De

plus, si une telle méfiance s'instaure dès le départ du système, le risque de statu quo n'est pas nul. C'est pourquoi nous avons pensé à un sous module pour apporter un début de réponse générique à cette question.

Patterns

Nous nous intéressons aux composants logiciels de plus bas niveau, celui des versions de fichiers, pour prendre en compte un certain nombre d'éléments. A titre d'illustrations, supposons que nous ne soyons intéressés que par

- l'évaluation directe du fichier ;
- l'évaluation directe du hash.

On a donc là deux (2) variables r1, r2 (ou un mot r1r2) dont nous comptons forcer les valeurs à des entiers de 0 à 5.

***0** : aucune information* ***1** : évaluation ∈ [-1, -0.8 [* ***2** : évaluation ∈ [-0.8, -0.6[*

***3** : évaluation ∈ [-0.6, -0.4 [* ***4** : évaluation ∈ [-0.4, -0.2[* ***5** : évaluation ∈ [-0.2, 0]*

On a donc potentiellement 6^2 soit 36 combinaisons mais nous ne nous intéressons vraiment qu'aux cas où l'une de ces variables est nulle :

- soit un nouveau fichier : 00
- soit un nouveau hash d'un fichier connu : x0 ; $x \in \{1, 2, 3, 4, 5\}$

On fait correspondre à chacune de ces combinaisons une structure *pattern_j*. Cette structure a les compteurs *rencontres, rejets, problèmes* et on calcule son évaluation de la même manière que pour un composant logiciel

$$Evaluation\ (pattern_j) = - \frac{problemes + rejets}{rencontres}$$

A cette formule se rajoute un mécanisme de « fading » identique à celui présenté précédemment. Après un certain seuil de rencontres, cette valeur entre en jeu pour l'acceptation d'une interaction avec un autre nœud N présentant une configuration C avec un ensemble de patterns P. Ainsi, on aura

Probabilité_contactC = minimum (proba_min, 1 + minimum (Evaluation (pattern_j))

avec pattern_j ∈ P et proba_min une borne inférieure de la probabilité de contact

Si l'on utilise cette valeur dans le « bizutage évoqué », le périple initiatique sera d'autant plus long que les nouveaux éléments auront présenté de mauvais comportements. Les composants logiciels légitimes seront généralement moins affectés du fait de leur stabilité et de leur distribution généralement plus large. En cas de mise à jour d'un fichier essentiel par exemple, beaucoup de nœuds seront concernés, pourront communiquer sans problème entre eux en attendant de convaincre les autres nœuds de leur caractère sain.

3.4 Algorithmes

Récapitulons ici les principaux algorithmes utilisés pour notre modèle. Plus que d'algorithmes, il s'agit surtout de formules dont les exécutions se font en temps constants ou linéaires pour les parcours (composants logiciels d'une configuration).

EvaluationDirecte(SC_i) := $-\frac{problemes + rejets}{rencontres} = ED(SC_i)$

EvaluationRapportée(SC_i) := $\frac{\sum EvaluationRapportée}{NbRapports} = ER(SC_i)$

Réputation(SC_i)

$Quantité_Information(SC_i) := NbRencontres(SC_i) + \alpha * NbRapports(SC_i)$

$d_weight(SC_i) := \frac{NbRencontres(SC_i)}{Quantité_Information(SC_i)}$

$r_weight(SC_i) := 1 - d_weight(SC_i)$

Si $(d_weight(SC_i) < \frac{1}{2})$

$R(SC_i) := d_weight(SC_i) * ED(SC_i) + r_weight(SC_i) * ER(SC_i)$

Sinon

$R(SC_i) := \frac{ED(SC_i) + ER(SC_i)}{2}$

```
        Fin Si
        retourner R(SC_i)
Fin
```

Le nœud N avec la configuration C engage une interaction I

```
Accepter ou refuser (Interaction I)
Si (∃ SC_i ∈ C / SC_i est NOUVEAU)
        RésultatInteraction := Evaluer I  OU Interaction annulée (plus forte probabilité)
Sinon si (∃ SC_i ∈ C / SC_i BANNI)
                Interaction annulée => RésultatInteraction := rejet
Sinon Faire Interaction I
                RésultatInteraction := Evaluer I
Fin Si
Mettre à jour la base de données avec RésultatInteraction
```

```
Mise à jour de la base de données (Résultat Interaction)
Pour chaque SC_i de C
        Rencontres(SC_i) := Rencontres(SC_i) + 1;
        Si RésultatInteraction = probleme
                Problemes(SC_i) := Problemes(SC_i) + 1;
        Si RésultatInteraction = rejet
                Rejets(SC_i) := Rejets(SC_i) + 1;
        Si Réputation(SC_i) < thresholdReputation
                Ajouter SC_i à la liste des SUSPECTS
Fin Pour
Si RésultatInteraction = probleme
        Appliquer des pénalités au pire SC_i de la liste des suspects
        Si SC_i.penalty ≥ thresholdPenalty => SC_i BANNI
Fin Si
```

CHAPITRE IV

SIMULATION ET RESULTATS

Afin de mettre en œuvre notre modèle, nous avons entrepris de développer un environnement de simulation suffisamment complet (quoique générique) pour nous permettre de tester notre approche. Notre modèle n'est pas lié à un protocole de routage des MANET mais se situe dans un cadre plus générique où la réputation des composants logiciels émerge des évaluations des interactions dans lesquelles ils sont impliqués. Pour faire notre preuve de concept, nous avons implémenté une simulation multi-agents épurée d'aspects réseau dont la mise en œuvre rigoureuse fait, de toute façon, défaut dans la plupart des simulations MANET [8]. Dans ce chapitre, nous commencerons par présenter les choix d'implémentation de notre simulation. Nous exposerons ensuite notre plan d'expérimentation et nous finirons sur une présentation et une analyse des résultats obtenus.

4.1 Implémentation

En plus de simuler le MANET et ses nœuds mobiles, il faut simuler au niveau de chaque nœud un système d'exploitation avec des logiciels qui tournent. Nous avons donc fait le choix d'une simulation multi-agents générique. Pour ce faire, nous avons utilisé les librairies essentielles de la plate-forme multi-agents Madkit dont nous commençons par présenter la substance. Nous présentons ensuite le modèle de mobilité tel que nous l'avons implémenté. La génération de la base de données des logiciels sains et de celle des logiciels malveillants fera l'objet des sous-sections suivantes. Nous conclurons la présentation de notre implémentation sur le cycle de vie des agents, lequel cycle fait intervenir les éléments précédemment exposés.

4.1.1 Madkit

Madkit [18] est une plate-forme multi-agents modulaire, écrite en Java et développée par l'équipe du professeur Jacques Ferber, dont les travaux font référence

dans le champ du paradigme multi-agents. Dans le cadre de notre simulation, les librairies Madkit auront servi de base à l'instanciation des agents et des classes de messages. Le moteur synchrone de Madkit qui permet de cadencer les agents sur un même pas d'horloge a été retenu. A chaque pas d'horloge, les agents exécutent une tâche donnée (mouvements, messages etc..).

4.1.2 Le modèle de mobilité

Chaque nœud/agent est représenté par un point (x, y) sur une carte de X*Y. Le terrain est organisé en une grille de cellules, en fait des rectangles (X_GRID, Y_GRID). Cela nous permet de définir aisément la portée d'un nœud comme étant limitée aux cellules qui l'entourent. Chaque nœud a sa vitesse propre et une trajectoire aléatoire. Les nœuds se déplacent librement sur le terrain et interagissent avec leurs voisins comme illustré à la Figure 4.1

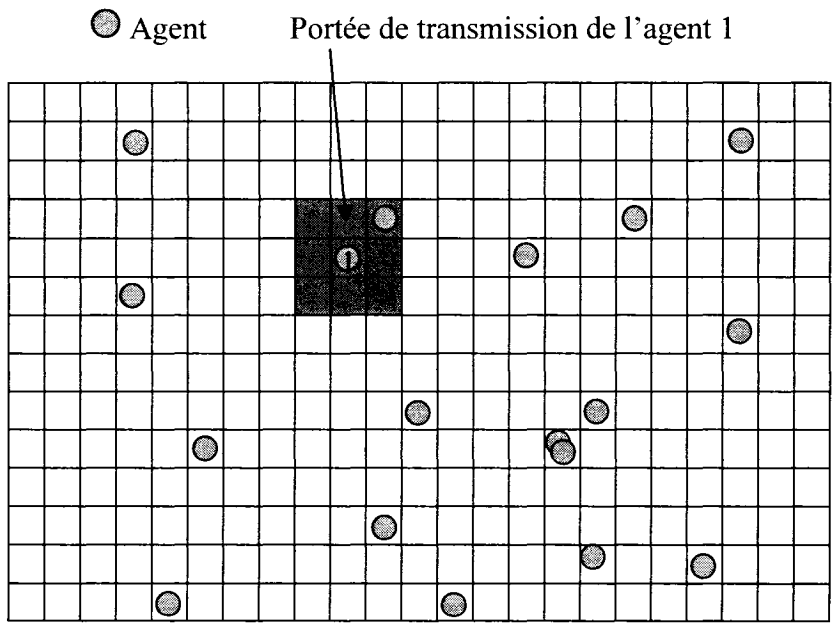


Figure 4.1 Le terrain et les nœuds mobiles

4.1.3 La base de données des composants logiciels légitimes.

Pour recréer à l'intérieur de chaque nœud une configuration logicielle cohérente, nous générons d'abord la base de données complète des composants logiciels légitimes.

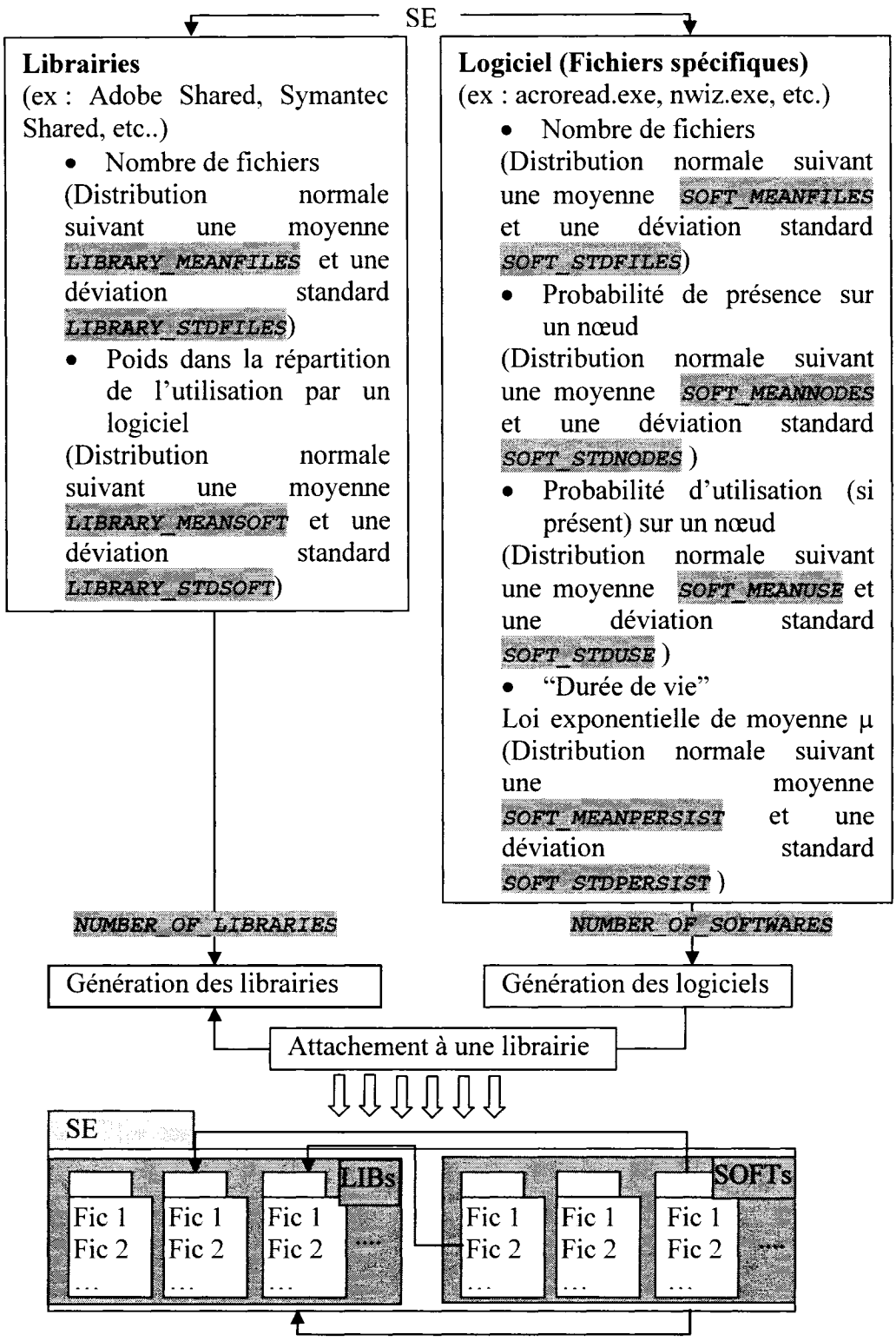


Figure 4.2 Construction de la base de données des logiciels légitimes

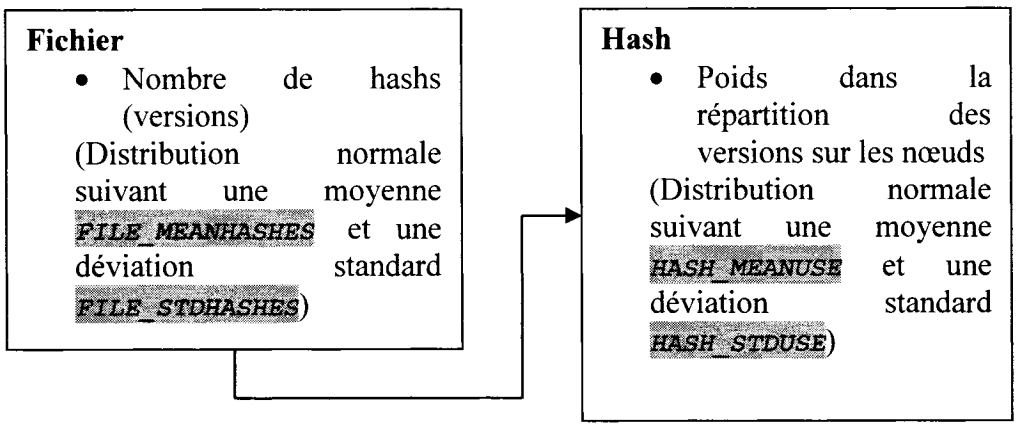


Figure 4.3 Composants logiciels

Pour chaque Système d'exploitation (SE), cette base de données est reconstruite comme illustré à la Figure 4.2 à partir de deux catégories : les groupes de librairies partagées et les fichiers spécifiques des divers logiciels (i.e. exécutables, dll dédiés...).

Nous définissons un groupe de librairies partagées comme l'ensemble des fichiers communs à un groupe de logiciels, typiquement ceux développés par une même compagnie. Par la suite, nous préférerons le terme librairie à « groupe de librairies ». Nous incluons dans cette définition des dossiers essentiels tels « \system32 » pour Windows ou « /bin » pour Linux. On gagne ainsi en cohérence dans la génération et l'installation des logiciels. Lorsqu'un nœud installe un logiciel qui requiert une librairie absente du nœud, celle-ci est installée et disponible pour les autres logiciels qui pourraient l'utiliser.

La seconde catégorie est celle des fichiers spécifiques à un logiciel donné. On y retrouvera typiquement l'exécutable du logiciel et quelques fichiers complémentaires.

En combinant ces deux catégories, nous obtenons une base de données plus réaliste des logiciels légitimes disponibles à l'installation sur un nœud. La configuration logicielle d'un nœud, qui est en définitive la liste des composants logiciels chargés, sera donc restituée à partir de la liste des logiciels qu'il aura précédemment installés. La Figure 4.3 présente les composants logiciels tels qu'ils sont implémentés dans notre simulation.

La génération de cette base de données est paramétrable suivant des lois normales de distribution. L'univers des logiciels légitimes est donc généré comme précédemment expliqué. Les paramètres exprimés comme « poids » expriment en fait des situations de concurrence et sont gérées avec une roulette biaisée.

Par exemple, lorsqu'un logiciel légitime installe un nouveau fichier sur le nœud, ses différentes versions (ici 4) sont en concurrence, chacune avec son poids. Un tirage aléatoire entre 0 et la somme de ces poids est effectué et la zone atteinte détermine le choix (la version installée) comme illustré à la Figure 4.4.

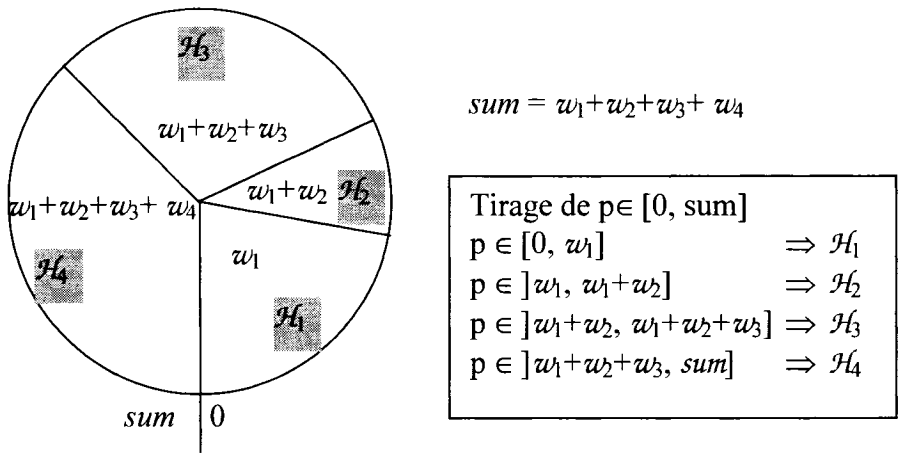


Figure 4.4 Roulettes biaisées pour les répartitions

4.1.4 Génération des logiciels malveillants

Dans notre simulation, un logiciel malveillant (*malware*) est identifié par quelques fichiers spécifiques, généralement un seul. Il peut se répliquer et est également programmé pour infecter des logiciels et/ou librairies cibles, notamment les librairies cœur (ex : /bin, \system32). Les *malwares* sont générés après les logiciels légitimes. Ils « choisissent » - selon les probabilités présentées dans l'encart « Malware » - les librairies et les logiciels à infecter. Une fois que le *malware* s'installe, il remplace les versions légitimes de ses cibles par des copies infectées. Tout comme les fichiers de logiciels sains, chaque fichier de *malware* présente un certain nombre de versions.

Malware

- Nombre de fichiers spécifiques

(Distribution normale suivant une moyenne `MALWARE_MEANFILES` et une déviation standard `MALWARE_STDFILES`)

- Probabilité d'infecter une librairie (*faible*)

`MALWARE_LIBRARYINFECTION`

- Probabilité d'infecter un logiciel (*faible*)

`MALWARE_SOFTWAREINFECTION`

- Probabilité d'infecter la librairie principale (/bin, \system32) du SE (*très forte*)

`MALWARE_COREINFECTION`

4.1.5 Cycle de vie des agents

Un agent nait (est activé) et « vit » au rythme du moteur synchrone : à chaque pas d'horloge, il exécute une routine donnée. Dans notre simulation, cette routine consiste à se déplacer, à éventuellement changer de configuration (lancer/terminer un logiciel ...) et générer des interactions avec les agents autour.

Activation:

Exploration de l'environnement

L'agent se voit assigner :

- une vitesse aléatoire (Distribution normale suivant une moyenne `NODE_MEAN_SPEED` et une déviation standard `NODE_STD_SPEED`)
- une position aléatoire (x, y) sur la carte
- une direction aléatoire : un vecteur v (a, b / $y = a*x + b$)

Configuration logicielle

- Choisir (selon la probabilité de présence à la Figure 4.2) les logiciels à installer dans la base de données des logiciels. Ce choix détermine les composants logiciels disponibles présents sur le nœud. Si le nœud est infecté (suivant le paramètre de

probabilité d'infection), un malware est attaché au système de fichiers et infecte ses cibles.

- L'agent se voit assigner une valeur pour rendre compte de sa disposition à changer plus ou moins souvent de configuration. Cette valeur *config_stability* est attribuée suivant une loi normale de moyenne `CONFIG_MEANVERSATILITY` et une déviation standard `CONFIG_STDVERSATILITY`.

Evolution :

Mouvement

- Le nœud peut changer de direction (le vecteur de direction est renouvelé) selon une certaine probabilité (très faible) ou s'il atteint les limites du terrain

Configuration

- A la fin de chaque période *config_stability*, le nœud pourra lancer un logiciel installé. La terminaison d'un logiciel dépend de sa « durée de vie » (cf Figure 4.2)

Interaction

- S'il a des voisins (dans les cellules adjacentes), le nœud interagit avec eux.

4.2 Plan d'expériences

La définition du plan d'expériences est une tâche ardue qui demande des préalables sérieux. Nous commencerons donc par faire un survol rapide des tests préliminaires et des observations que nous en avons tirés. Nous continuerons sur une présentation des paramètres généraux de notre simulation et des données d'expérimentation qui sont statiques. Nous argumenterons le choix des facteurs primaires que nous avons identifiés pour introduire par la suite les indices de performance retenus. Enfin, seront présentées les sessions retenues pour notre plan d'expériences.

4.2.1 Tests préliminaires

La génération de l'univers physique (terrain) et logiciel de notre simulation est complexe et les paramètres à régler sont nombreux. Des tests préliminaires nous ont toutefois permis de constater l'influence négligeable sur nos indices de performance de la plupart des paramètres. La majorité des paramètres influent surtout sur le nombre moyen de composants logiciels dans une configuration donnée, et n'influencent que légèrement l'évolution du système. Le résultat obtenu est similaire à l'effet produit par des variations d'autres paramètres tels que le nombre de logiciels et de librairies.

Un seul paramètre se détache très nettement : celui de la vitesse des nœuds. En effet, lorsque la vitesse des nœuds est faible, ils se déplacent très lentement et conservent de fait le même ensemble de voisins pendant un certain temps. Cet état de faits, certes intéressant dans un modèle de réputation des nœuds donne de très mauvais résultats pour un modèle uniquement basé sur les configurations logicielles. La raison en est assez évidente et même générique : on ne peut tirer de bonnes conclusions sur un échantillon trop réduit de nœuds. L'information apportée par des interactions même nombreuses avec un nombre très réduit de nœuds ne cumule pas quantité et qualité. D'ailleurs, si l'on considère un réseau routier, ne serait-ce qu'avec les véhicules circulant en sens inverse, il y a de bonnes chances d'avoir un voisinage suffisamment dynamique. Nous avons donc opté pour une mobilité importante des nœuds.

En moyenne, chaque nœud a un vecteur vitesse v dont la norme est la taille d'une cellule. A chaque pas d'horloge, le nœud est « translaté » de v.

Une fois ce paramètre ainsi ajusté, les paramètres servant dans la simulation des nœuds comme systèmes logiciels complexes perdent toute influence significative. Cela nous permet de nous concentrer sur des paramètres généraux plus significatifs.

4.2.2 Paramètres généraux

C'est l'occasion de présenter à la Figure 4.5 le GUI, assez complet, développé pour notre simulation qui nous permet entre autres d'entrer les paramètres essentiels.

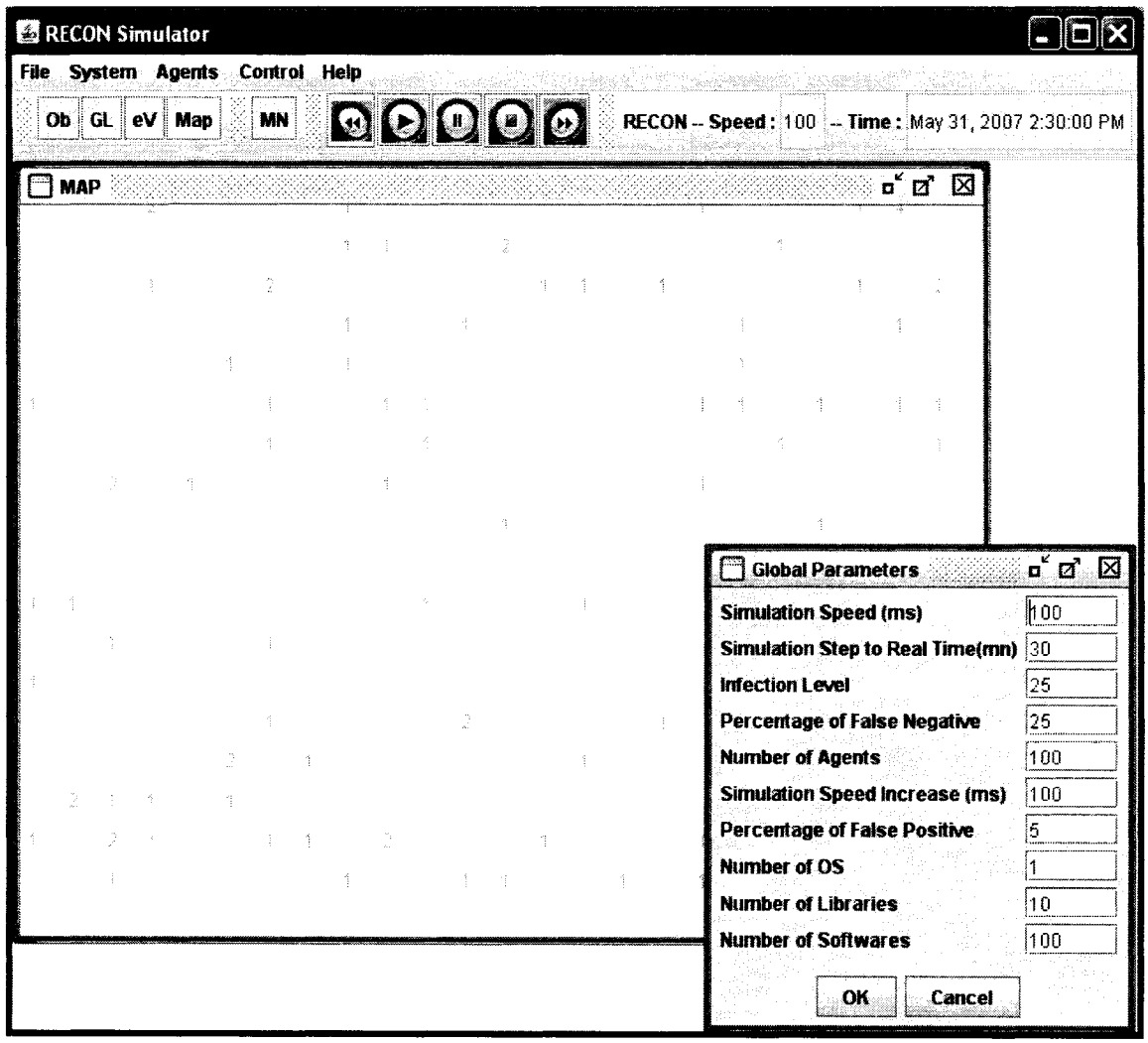


Figure 4.5 RECON : REputation des CONfigurations

Base de données des logiciels

- Nombre de logiciels
- Nombre de librairies

Évaluation des interactions

Notre modèle n'implémente pas de mécanisme d'évaluation des interactions. En lieu et place, nous utilisons un mécanisme abstrait et générique reconstitué à partir des indices de performances usuels de tels mécanismes.

- Pourcentage de faux positifs de l'évaluation d'une interaction (I-FP)

  Une interaction avec un nœud « sain » est notée comme « malsaine ».

- Pourcentage de faux négatifs de l'évaluation d'une interaction (I-FN)

Une interaction avec un nœud infecté est notée comme « saine ». Nous ne distinguons pas ici le cas où l'interaction est effectivement correcte (stratégie de nœud malveillant) de celui où l'interaction bien que malsaine n'est pas détectée

Nœuds

- Nombre de nœuds
- Niveau d'infection : Pourcentage de nœuds infectés
- Vitesse des nœuds

4.2.3 Données d'expérimentation

Il faut évoquer ici le manque de standards dans les simulations MANET. La définition de ces standards est un champ plus ou moins actif de la recherche mais il n'existe à l'heure actuelle aucun consensus sur la manière dont il faudrait conduire des simulations MANET. Nous nous sommes donc surtout attachés à faire des expériences qui nous fournissent une grille d'évaluation des techniques utilisées.

La génération de la base de données sur l'environnement logiciel nous fournit avec 100 logiciels et 10 groupes de librairies, une base de données d'environ 500 fichiers qui totalisent environ 1800 versions. 50 logiciels malveillants sont générés et répartis aléatoirement sur les nœuds infectés suivant le niveau d'infection.

En résumé, nous avons

100 logiciels, 10 groupes de librairies, 50 logiciels malveillants

=> 500 fichiers, 1800 versions de fichiers

Les configurations des nœuds contiennent en moyenne 100 composants logiciels

*Le terrain parcouru par les nœuds est un rectangle de 24*18 cellules.*

Dans notre simulation, nous ne raisonnons pas sur le « temps » mais sur le nombre d'interactions. C'est pourquoi nous présenterons nos résultats en substituant au temps le nombre d'interactions effectuées par les nœuds. Il nous semble que l'évolution du système prend mieux son sens dans un tel cadre.

4.2.4 Facteurs primaires

Nous avons choisi de ne nous intéresser qu'à un seul Système d'Exploitation (SE), étant entendu que les résultats recueillis sur un seul SE seront assez significatifs. En effet, les statistiques de composants logiciels appartenant à différents SE sont dûment séparées suivant la logique des niveaux présentés à la Figure 3.2 (Section 3.3). Le Tableau 4.1 présente les facteurs primaires retenus et leurs niveaux associés.

Même si notre modèle de réputation est basé sur les composants logiciels, le nombre de nœuds est de toute évidence un facteur dont l'influence est intéressante à analyser. Nous avons retenu 3 ordres de grandeur : 25, 50, 100 nœuds.

Nous considérons ici que le système est mis en œuvre dans un environnement déjà infecté. Le pourcentage de nœuds infectés est donc un facteur d'importance. Les 3 niveaux 5%, 15% et 25% témoignent d'un environnement initialement assez peu infecté. Un environnement majoritairement et durablement infecté demande une autre approche pour déterminer les composants malsains.

Le pourcentage de faux négatifs dans l'évaluation des interactions I-FN rend compte de deux phénomènes cumulés :

1. le mécanisme d'évaluation des interactions n'arrive pas à détecter un mauvais comportement ;
2. le logiciel malveillant ne se comporte pas systématiquement mal, pour éviter de se faire repérer.

Le premier cas est plus ennuyeux car le préjudice causé par le *malware* est effectif mais on peut raisonnablement espérer des pourcentages très faibles. Dans le second cas, il n'y a pas de préjudice mais le nœud corrompu peut se comporter comme il faut tout le temps. Mais là encore, tout comme un individu « fou » mais se comportant

toujours bien en société, un nœud contenant un *malware* mais n'affichant jamais de mauvais comportements n'a pas à être isolé. Nous avons donc choisi comme niveau le plus élevé 50%, pour nous limiter aux *malwares* se comportant mal au moins une fois sur deux.

Tableau 4.1 Facteurs primaires et niveaux associés :

Facteurs		**Niveaux**	
Nom	**Symbole**	**Nom**	**Description**
Nombre de nœuds	N	Petit	25
		Moyen	50
		Grand	100
Pourcentage de nœuds infectés	M	Petit	5%
		Moyen	15%
		Grand	25%
Faux négatifs après interaction	I-FN	Petit	10%
		Moyen	25%
		Grand	50%
Faux positifs après interaction	I-FP	Aucun	0%
		Moyen	2%
		Grand	5%
Vitesse des nœuds	S	Petit	½ cellule
		Moyen	1 cellule
		Grand	2 cellules

Le pourcentage de faux positifs dans l'évaluation d'une interaction a été choisi dans un intervalle assez bas. Il représente essentiellement les défaillances passagères d'un nœud à fournir l'information telle qu'il l'a perçue ou à coopérer au bon

fonctionnement du réseau. Nous préférerons un mécanisme d'évaluation des interactions qui laisse passer - rarement - un mauvais comportement à un mécanisme qui épingle trop souvent des comportements licites.

Il était également intéressant de présenter les résultats obtenus avec différentes vitesses des nœuds. Nous avons choisi de tester un système uniquement basé sur les configurations logicielles sans gestion des identités des nœuds. Mais, en cas de partitionnement durable du réseau où on a des groupes de nœuds qui restent longtemps entre eux sans contacts avec d'autres nœuds, les compteurs au niveau des statistiques des composants logiciels deviennent moins qualitatifs. Une vitesse plus faible des nœuds favorise un tel phénomène, tandis que des vitesses plus élevées facilitent un meilleur brassage des configurations. Ce point pose la question de l'intégration des identités des nœuds dans notre modèle, au moins sur certains types d'environnements.

4.2.5 Indices de performance

Le but de notre modèle est de détecter et d'isoler les nœuds malsains. Avant d'entreprendre une interaction, les nœuds s'envoient leur configuration. L'interaction n'est effective que lorsque chacun des nœuds juge acceptable la configuration de l'autre.

C étant la configuration présentée et I l'interaction, ¬ I le rejet de l'interaction

$$C \text{ contient un composant malicieux} \Leftrightarrow \neg I$$

Partant de là, nous définissons un C-FN faux négatif comme étant

$$(C \text{ contient un composant malicieux}) \text{ ET } (I)$$

Et un C-FP faux positif comme étant

$$\neg (C \text{ contient un composant malicieux}) \text{ ET } \neg I$$

Nos indices de performance sont donc :

- le pourcentage de faux positifs générés par notre modèle C-FP
- le pourcentage de faux négatifs générés par notre modèle C-FN

Ces pourcentages devraient être nuls, entendu qu'idéalement, un nœud sain n'interagit jamais avec un nœud corrompu mais ne rejette jamais une interaction avec un autre nœud sain.

4.2.6 Sessions

Nous avons fait le choix d'effectuer des expériences « un facteur à la fois » pour analyser l'influence des facteurs primaires sur les indices définis. Le Tableau 4.2 présente les sessions retenues pour notre plan d'expériences. A chaque session, une série de 50 tests a été effectuée avec des graines aléatoires dépendant du temps système.

Tableau 4.2 Sessions du plan d'expérimentation

Session	N	M	I-FN	I-FP	S
Influence du Nombre de nœuds					
1	Petit	Moyen	Moyen	Moyen	Moyen
2	Moyen	Moyen	Moyen	Moyen	Moyen
3	**Grand**	**Moyen**	**Moyen**	**Moyen**	**Moyen**
Influence du niveau d'infection initial					
4	Grand	Petit	Moyen	Moyen	Moyen
5	Grand	Grand	Moyen	Moyen	Moyen
Influence des faux négatifs de l'évaluation d'une interaction					
6	Grand	Moyen	Petit	Moyen	Moyen
7	Grand	Moyen	Grand	Moyen	Moyen
Influence des faux positifs de l'évaluation d'une interaction					
8	Grand	Moyen	Moyen	Aucun	Moyen
9	Grand	Moyen	Moyen	Grand	Moyen
Vitesse des nœuds					
10	Grand	Moyen	Moyen	Moyen	Petit
11	Grand	Moyen	Moyen	Moyen	Grand
Expérience Détaillée					
	Grand	Grand	Moyen	Grand	Moyen

La session 3 est la session référence de notre plan d'expériences. Dans l'analyse de l'influence d'un facteur primaire sur les résultats, elle sera à chaque fois utilisée.

4.3 Résultats

Une fois les indices de performance et les sessions définis, on peut mener les expériences et procéder à leur interprétation. Nous nous intéressons à l'évolution du système sur les indices de performance C-FP et C-FN identifiés. Nous avons choisi de nous référer au nombre d'interactions effectuées par un nœud sain. Ces interactions incluent aussi bien les interactions avec des nœuds sains qu'avec des nœuds corrompus. Selon le pourcentage de nœuds infectés, on pourra retrouver le nombre d'interactions effectives ou rejetées avec des nœuds corrompus. Dans nos représentations graphiques, nous utilisons un pas de 100 interactions au bout duquel est relevé, pour chaque nœud sain, les C-FP et C-FN qui ont eu lieu durant cette période (100 dernières interactions). Pour chaque session et à chaque pas, on calcule la moyenne de ces indices de performance sur tous les nœuds sains. C'est cette moyenne qui est représentée en ordonnée de nos graphes qui présentent les résultats sur les 5000 premières interactions des nœuds sains dans la mesure où la quasi-totalité (sauf indication contraire dans l'analyse) des sessions convergent à ce niveau d'évolution du système.

Dans les sous sections suivantes, nous discuterons de l'influence sur l'évolution du système des facteurs primaires considérés. Le nombre de nœuds, le pourcentage de nœuds infectés, les faux négatifs (I-FN) et positifs (I-FP) dans l'évaluation des interactions et la vitesse des nœuds seront ainsi successivement étudiés dans leur influence sur l'évolution du système. Nous reviendrons enfin sur l'expérience supplémentaire pour présenter avec plus de détails les résultats obtenus.

4.3.1 Influence du nombre de nœuds

Selon que la densité soit plus ou moins grande, le nombre d'interactions sera plus ou moins élevé à un temps t de la simulation. Il convient de faire cette remarque dans l'analyse de l'influence des nœuds. Il est évident qu'un nœud dans la session 1 (25 nœuds) mettra plus de temps à effectuer 100 interactions qu'un autre nœud dans les sessions 2 ou 3 (respectivement 50 et 100 nœuds).

Sessions considérées

Session	N	M	I-FN	I-FP	S
Nombre de nœuds					
1	Petit (25 nœuds)	Moyen (3 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)
2	Moyen (50 nœuds)	Moyen (7 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)
3	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)

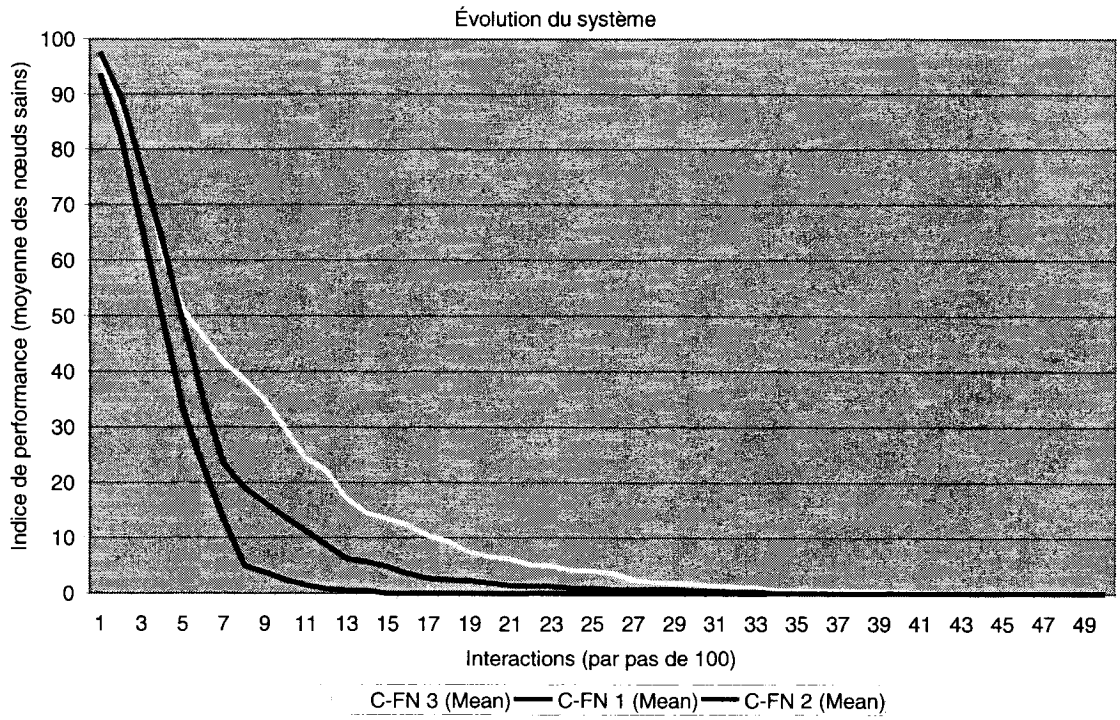


Figure 4.6 Évolution du C-FN selon le nombre de nœuds

Notre système finit par détecter toutes les configurations malicieuses, comme illustré à la Figure 4.6. On remarque qu'avec un petit nombre de nœuds (courbe C-FN 1), le système détecte plus rapidement (en nombre d'interactions) toutes les configurations malicieuses. Les 3 courbes sont ordonnées suivant le nombre de nœuds dans le système. On constate que moins on a de nœuds, moins on a besoin d'interactions pour détecter les configurations malicieuses. Cela peut s'expliquer par le fait que le nombre de composants logiciels et de leurs combinaisons augmente avec le nombre de nœuds. Le point positif, c'est l'augmentation de la densité : à terrain constant, le nombre

d'interactions est d'autant plus vite atteint qu'il y a plus de nœuds dans l'environnement (plus de chances de rencontres).

A la différence d'un modèle de réputation pour les nœuds, la convergence du système est plus significative puisque l'information accumulée sera utile sur un ensemble beaucoup plus large de nœuds. Tout nœud infecté par l'un des *malwares* repérés et écartés se verra automatiquement isolé.

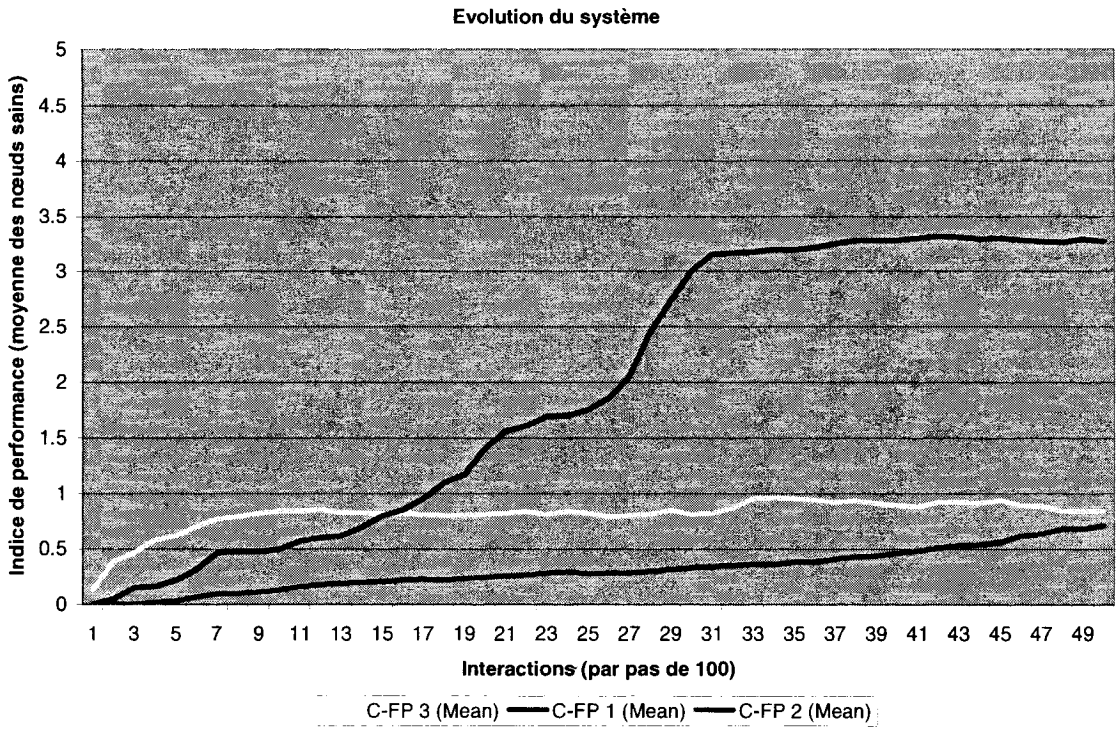


Figure 4.7 Évolution du C-FP selon le nombre de nœuds

A la Figure 4.7 on peut noter dans le pire des cas que le pourcentage de faux positifs générés par notre système C-FP reste en moyenne autour de 3 %. La session 1 (25 nœuds) se distingue par un C-FP beaucoup plus élevé. En effet, autant les interactions que les échanges d'information de réputation y sont rares. Les erreurs sont donc plus fréquentes. Cet état de faits nous donne une première indication sur la part accrue de faux positifs quand les nœuds ne sont pas suffisamment nombreux.

4.3.2 Influence du pourcentage de nœuds infectés

Sessions considérées

Session	N	M	I-FN	I-FP	S
Pourcentage de nœuds infectés					
3	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)
4	Grand (100 nœuds)	Petit (5 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)
5	Grand (100 nœuds)	Grand (25 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)

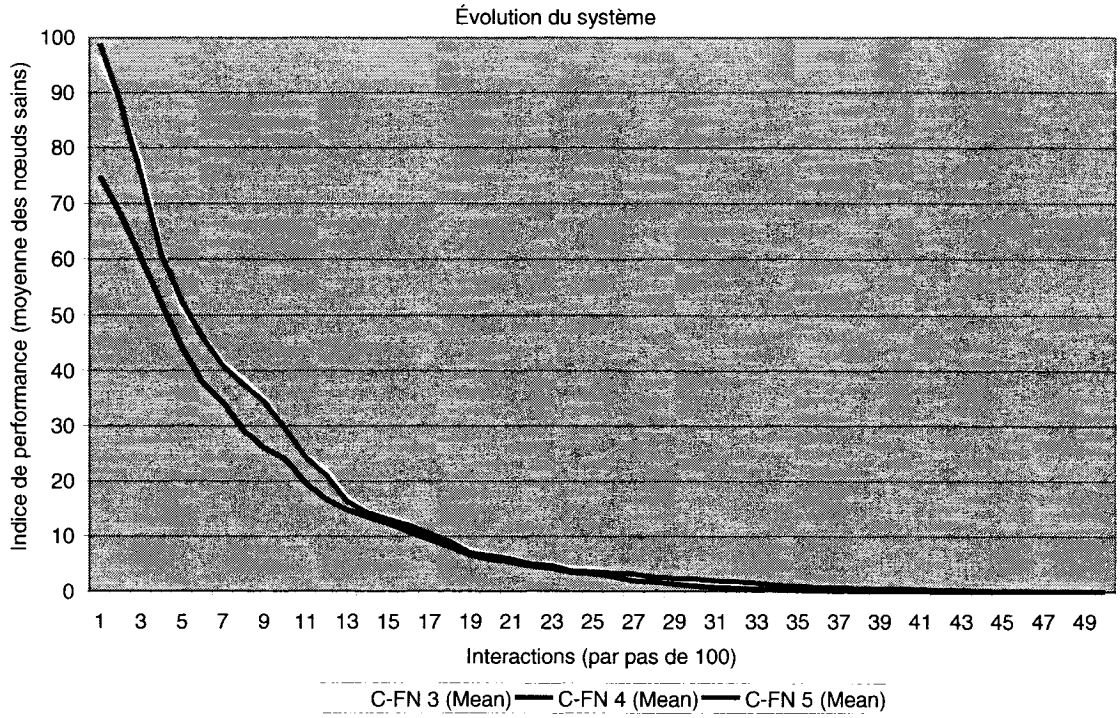


Figure 4.8 Évolution du C-FN selon le pourcentage de nœuds infectés

Les courbes d'évolution du C-FN selon le taux d'infection, représentées à la Figure 4.8 sont assez semblables. Avec 5% d'infection (C-FN 4), le système démarre plus fort dans la détection des mauvais composants logiciels. Dès les 100 premières interactions, le pourcentage de non-détection descend à 75% environ alors que les sessions à 15 %(C-FN 3) et 25 % (C-FN 5) ne détectent quasiment aucune mauvaise configuration dans cette période. Les trois courbes se rejoignent cependant relativement vite et ne présentent plus de différences significatives.

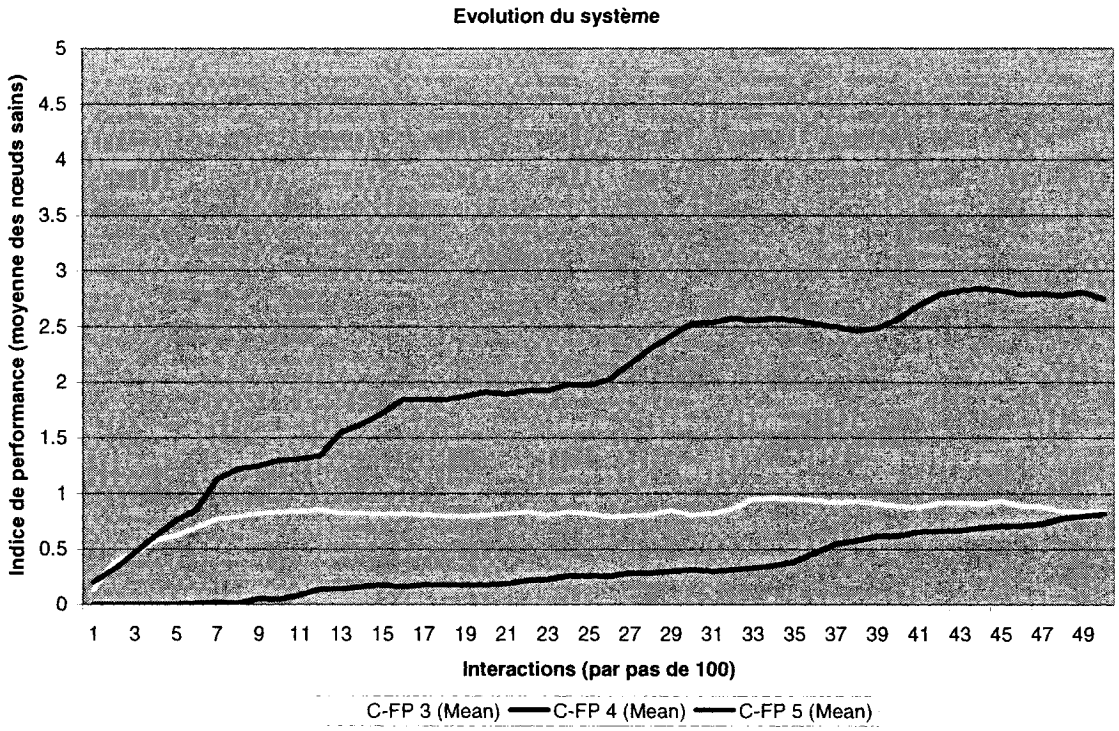


Figure 4.9 Évolution du C-FP selon le pourcentage de nœuds infectés

Le pourcentage de faux positifs C-FP augmente avec le taux d'infection initial de l'environnement comme illustré à la Figure 4.9. Les composants logiciels marginaux ont plus de chances d'être classifiés comme malsains. Ils ne bénéficient pas d'une grande distribution pour pallier les mauvaises évaluations qu'ils reçoivent lorsqu'ils se trouvent dans des configurations infectées. Soulignons néanmoins qu'avec moins de 3% de faux positifs générés par le modèle, nous avons là des résultats que nous estimons acceptables

4.3.3 Influence du I-FN

On peut considérer de plusieurs manières les sessions retenues. La session extrême 7 à 50% de I-FN nous fournit deux hypothèses extrêmes :

1. Le *malware* se comporte toujours mal mais le mécanisme d'évaluation est très mauvais, ne détectant les mauvais comportements qu'une fois sur 2;
2. Le *malware* ne se comporte mal qu'une fois sur 2, et le mécanisme d'évaluation est parfait, détectant toujours un mauvais comportement.

Entre ces 2 hypothèses, s'intercalent une multitude d'autres qui pourraient amener au même I-FN. Par exemple, le *malware* se comporte mal 3 fois sur 4 mais ce mauvais comportement n'est détecté que 2 fois sur 3.

Sessions considérées

Session	N	M	I-FN	I-FP	S
			Interaction – Faux Négatifs		
3	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)
6	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Petit (10%)	Moyen (2%)	Moyen (1 cellule)
7	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Grand (50%)	Moyen (2%)	Moyen (1 cellule)

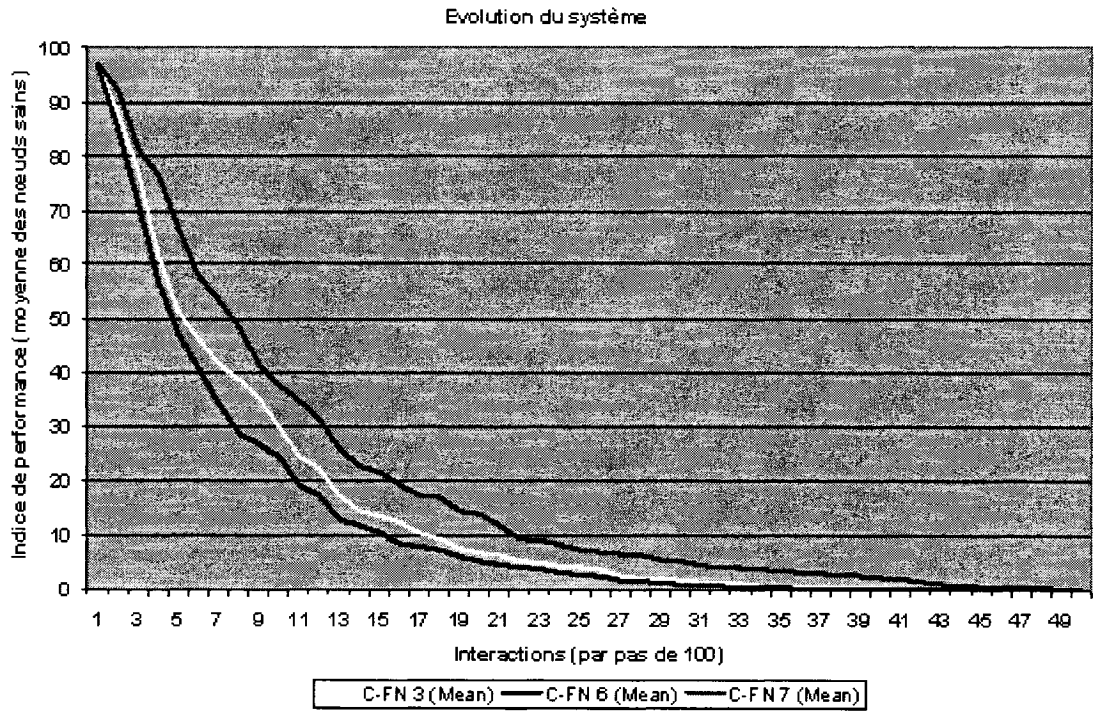


Figure 4.10 Évolution du C-FN selon le I-FN

Comme le montre la Figure 4.10, même à 50% de I-FN (C-FN 7), notre système est efficace dans la détection des mauvaises configurations. Il converge certes moins vite mais sa pente initiale reste sensiblement raide. Dans le cas où le malware s'emploie à ne pas se comporter mal à chaque interaction, on a là des résultats prometteurs quant à la capacité du modèle à isoler des composants *relativement* plus problématiques.

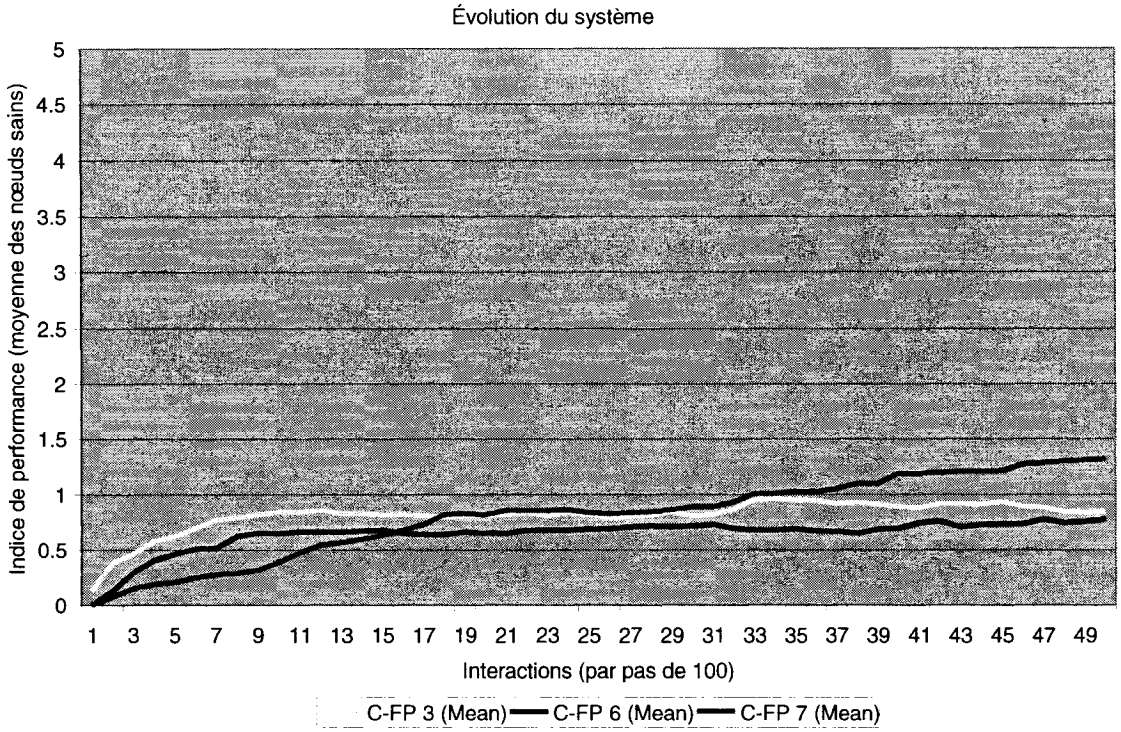


Figure 4.11 Évolution du C-FP selon le I-FN

Les courbes de faux positifs illustrées à la Figure 4.11 sont toutes très basses (autour de 1%) car si les mauvais comportements (ou leurs détections) sont plus rares, les risques d'associer des composants logiciels marginaux (mais sains) à des comportements délictueux en sont logiquement réduits.

4.3.4 Influence du I-FP

Sessions considérées

Session	N	M	I-FN	I-FP	S
				Interaction – Faux Positifs	
3	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	**Moyen (2%)**	Moyen (1 cellule)
8	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	**Aucun (0%)**	Moyen (1 cellule)
9	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	**Grand (5%)**	Moyen (1 cellule)

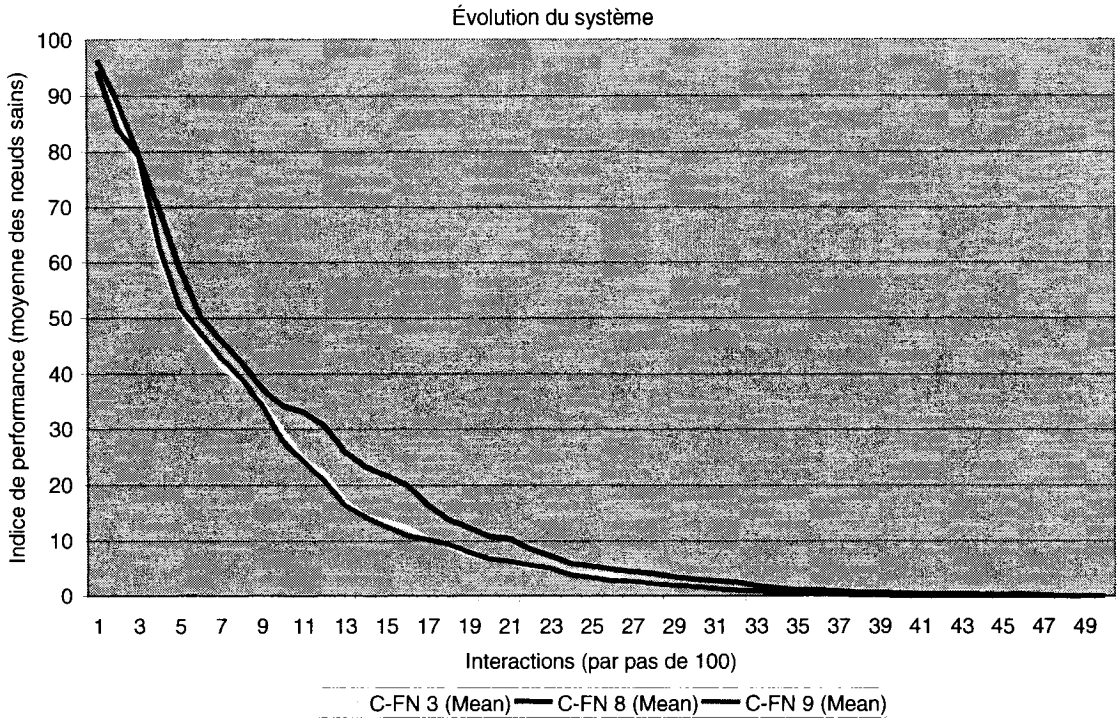


Figure 4.12 Évolution du C-FN selon le I-FP

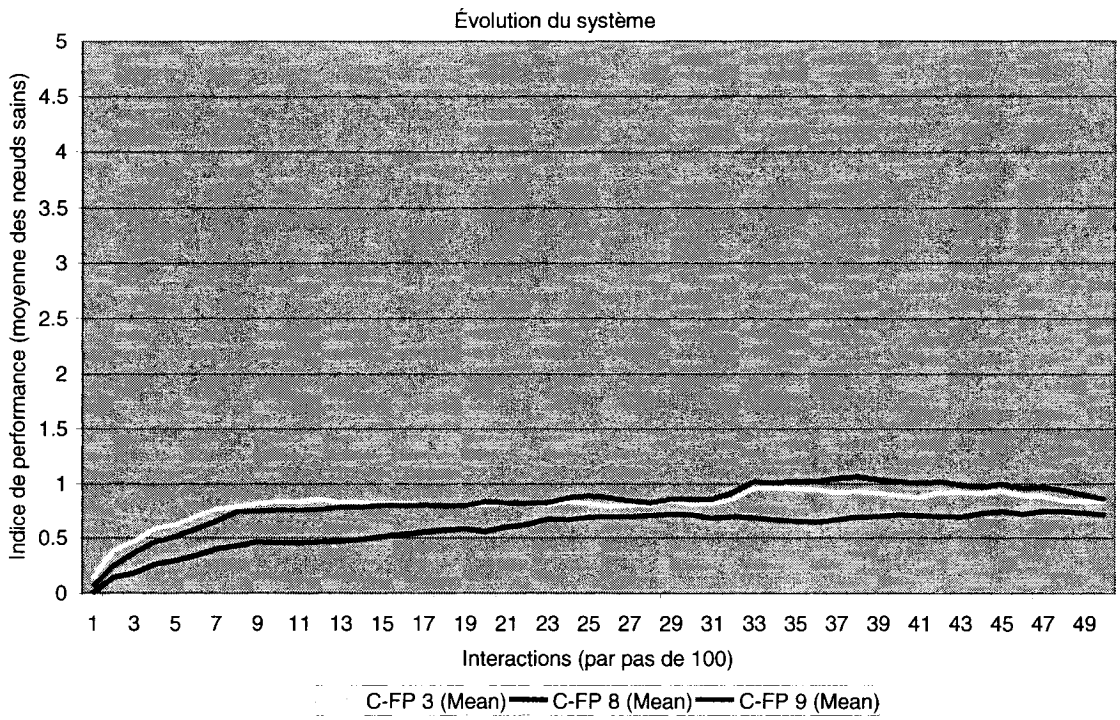


Figure 4.13 Évolution du C-FP selon le I-FP

Les courbes d'évolution C-FN et C-FP observables sur les Figures 4.12 et 4.13 respectivement ne présentent pas de différences significatives. On peut remarquer sur la figure 4.12 que la courbe C-FN 8 (aucun I-FP) se distingue par une détection un peu plus lente des mauvaises configurations. Les courbes C-FP sont toutes très basses (autour de 1%) et assez similaires dans leur évolution.

Au final, le I-FP, du moins dans les intervalles que nous lui avons définis (0 à 5%), a peu d'incidences sur l'évolution des indices de performance retenus.

4.3.5 Influence de la vitesse des nœuds

Sessions considérées

Session	N	M	I-FN	I-FP	S
					Vitesse des nœuds
3	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	Moyen (2%)	Moyen (1 cellule)
10	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	Moyen (2%)	Petit (½ cellule)
11	Grand (100 nœuds)	Moyen (15 nœuds infectés)	Moyen (25%)	Moyen (2%)	Grand (2 cellules)

On peut remarquer sur la Figure 4.14 que la courbe CFN-11 (vitesse des nœuds à 2 cellules) converge beaucoup plus vite. L'information accumulée gagne manifestement en qualité avec le nombre de nœuds rencontrés dans un intervalle de temps réduit. Ce ne sont pas tant les nœuds en eux-mêmes qui sont intéressants mais la diversité des combinaisons logicielles ainsi rencontrées. Ainsi, par opposition, quand la vitesse est faible avec des interactions récurrentes entre les mêmes nœuds, le pourcentage de non détection souffre d'erreurs répétées et peine à atteindre 0 au bout de 5000 interactions.

Sur la Figure 4.15, la courbe C-FP 10 (faible vitesse) se détache nettement par son pourcentage d'erreurs non maitrisé. Au bout de 5000 interactions, nous avons plus de 3% de faux positifs générés mais ce pourcentage ne se stabilise qu'à plus de 5% après 10000 interactions. On peut retenir de ce qui précède, qu'on serait bien inspiré de rajouter à un système uniquement basé sur la réputation des configurations logicielles, un petit mécanisme pour gérer la prise en compte d'une suite d'interactions avec un même nœud sur une même courte période.

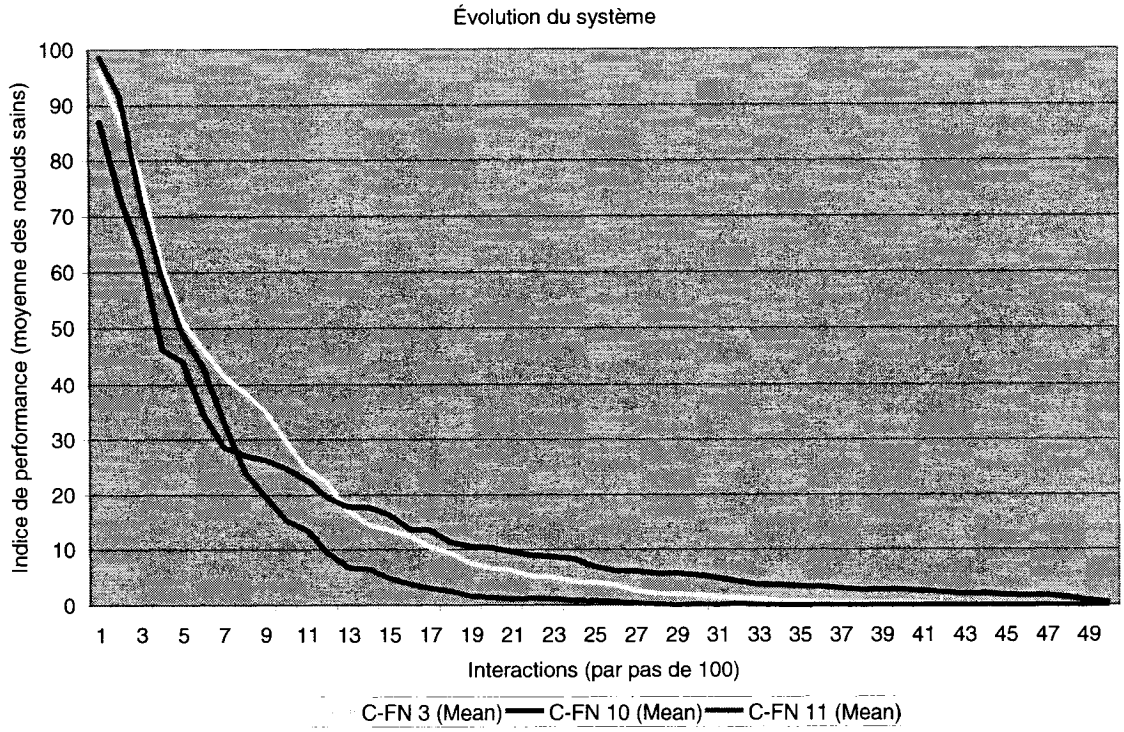


Figure 4.14 Évolution du C-FN selon la vitesse des nœuds

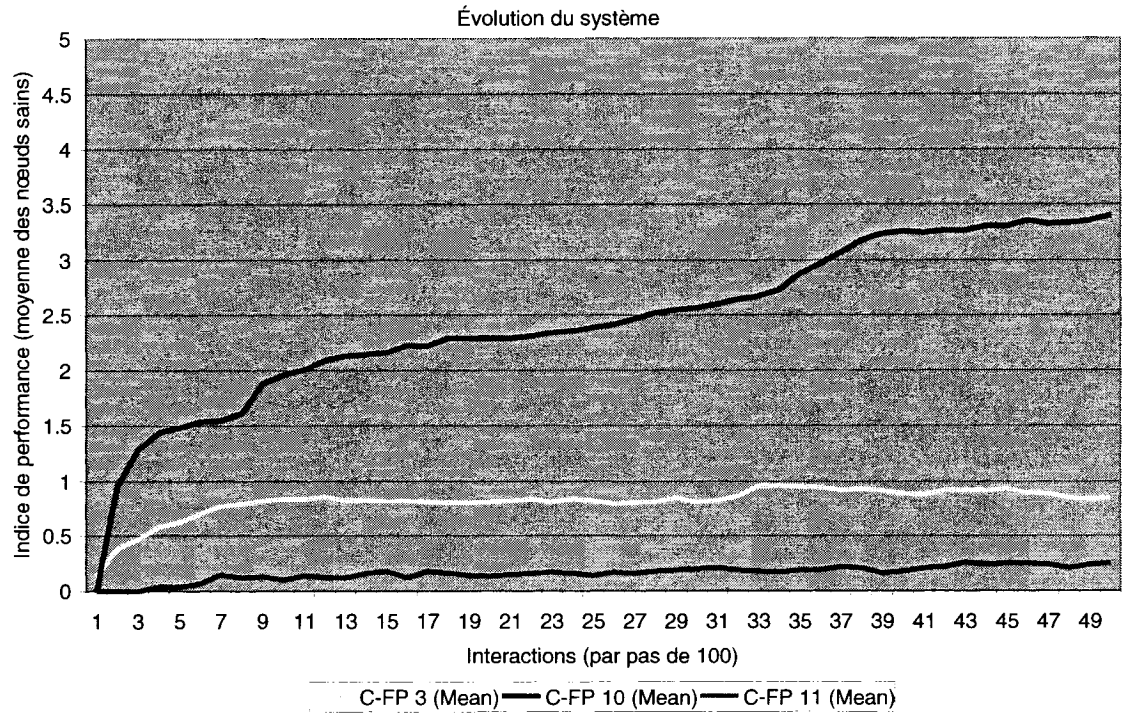


Figure 4.15 Évolution du C-FP selon la vitesse des nœuds

4.3.6 Une session détaillée

Dans ce qui précède, nous avons raisonné sur les moyennes de nos indices de performance. Ces moyennes sont calculées sur l'ensemble des nœuds sains du réseau. Elles permettent d'avoir des comparaisons cohérentes entre différentes sessions qui font varier le nombre de nœuds dans le réseau et le nombre de nœuds infectés. Mais elles ne rendent pas au mieux les différences notables qu'il y a entre les nœuds. Dans ce qui va suivre, nous présentons des résultats d'un test de la session 13 pour mieux rendre compte de l'expérience. La détection des composants logiciels malveillants et la génération de faux positifs sont perceptibles au niveau de chacun des 75 nœuds suivants. On remarquera ainsi que la quasi-totalité des nœuds génèrent très peu de faux positifs et que les moyennes observées sont le fait de quelques nœuds qui ont évolué vers un certain niveau de paranoïa.

Expérience Détaillée	
Nombre de nœuds	100
Taux d'infection	25 % : 25 nœuds corrompus, 75 sains
I-Faux Négatifs	25 %
I-Faux Positifs	5 %
Vitesse des nœuds	2 cellules

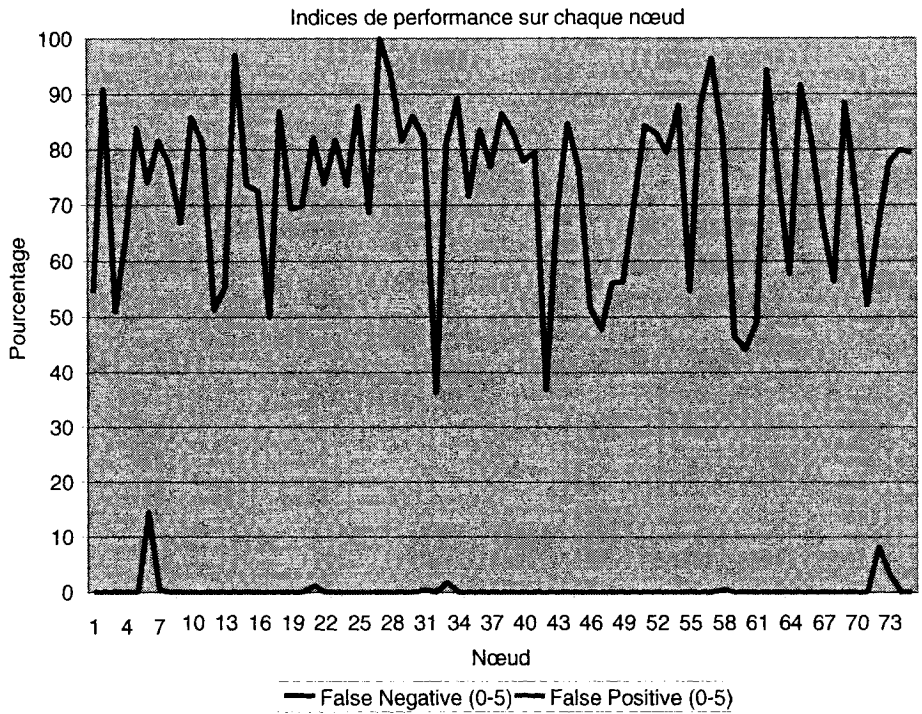


Figure 4.16 C-FN et C-FP des 75 nœuds sains (Interactions 0-500)

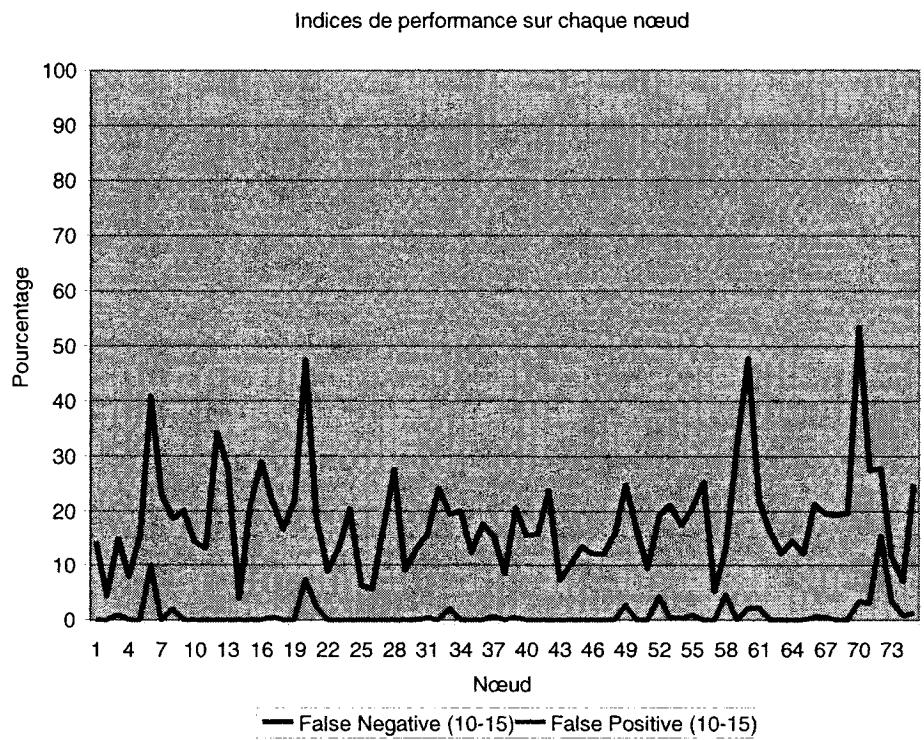


Figure 4.17 C-FN et C-FP des 75 nœuds sains (Interactions 1000-1500)

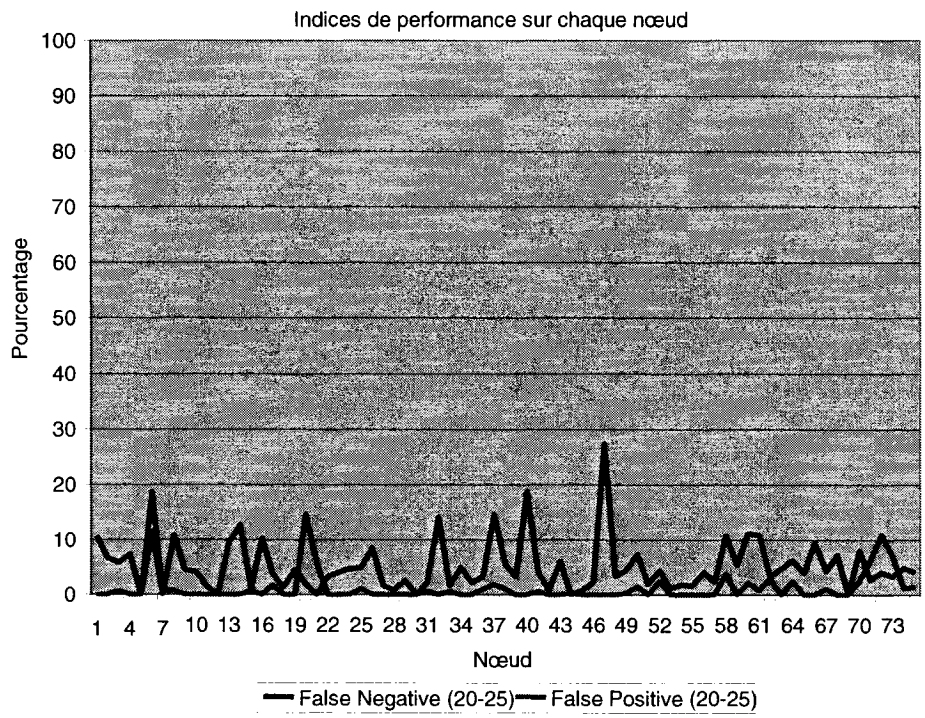


Figure 4.18 C-FN et C-FP des 75 nœuds sains (Interactions 2000-2500)

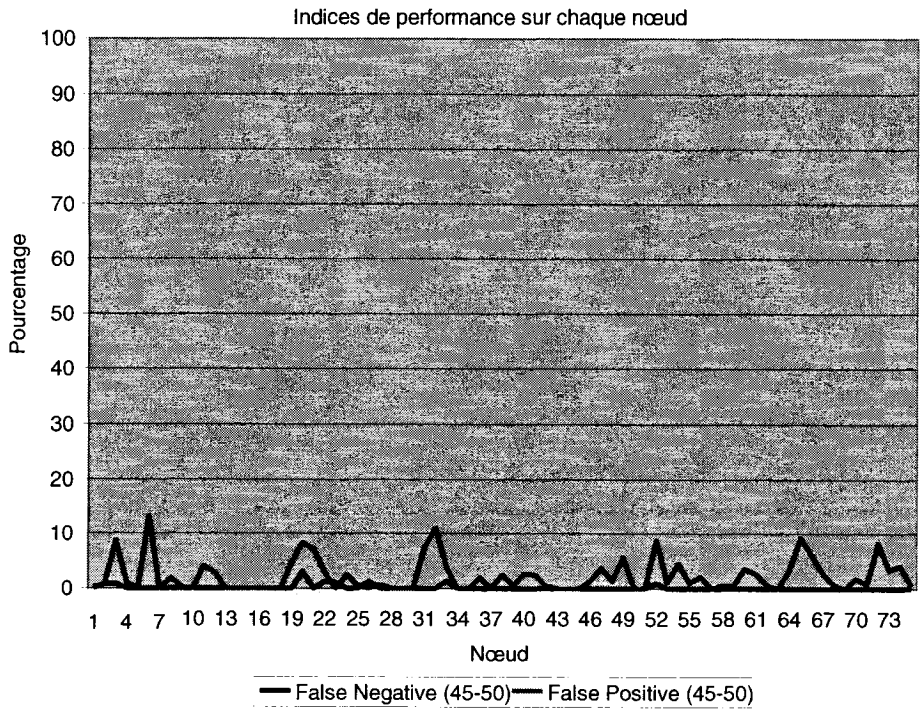


Figure 4.19 C-FN et C-FP des 75 nœuds sains (Interactions 4500-5000)

Les graphes des Figures 4.16, 4.17, 4.18, 4.19 nous renseignent sur les différences sensibles dans l'évolution du système au niveau de chaque nœud. Tous les nœuds finissent par détecter toutes les configurations malsaines. Par contre, certains nœuds atteignent des pics élevés (10%) de faux positifs.

Pour évaluer la vitesse de détection des composants malicieux, on peut s'intéresser au nombre de mauvaises expériences qu'un nœud a avec un composant malicieux donné avant de le bannir. Le graphe ci-dessous représente les composants malicieux rencontrés par le nœud 10 après 5000 interactions. A cette étape d'évolution, toutes les configurations logicielles contenant des composants logiciels malicieux sont rejetées. Le graphe de la Figure 4.20 nous renseigne sur le nombre de mauvaises expériences qu'il aura fallu avec chacun de ses composants malicieux avant d'aboutir à cette détection totale.

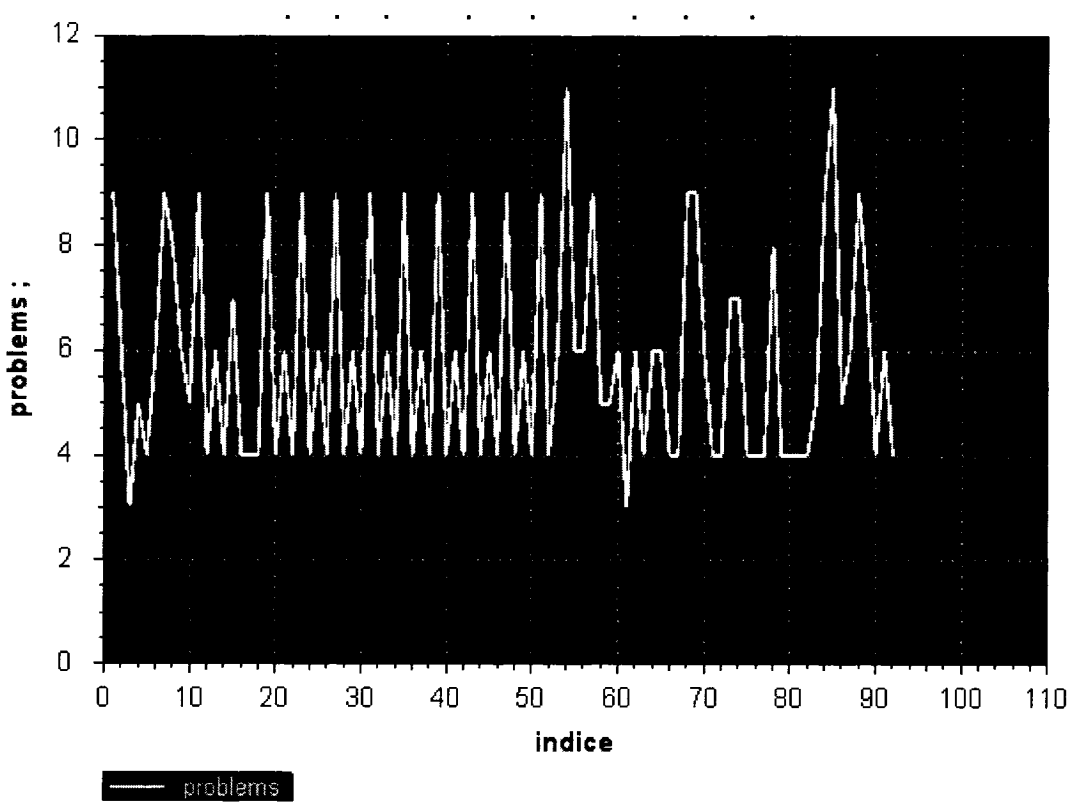


Figure 4.20 Vitesse de détection des composants malicieux pour un nœud

On a en abscisse les composants malicieux indexés et en ordonnée le nombre d'interactions malicieuses détectées lorsque le composant indexé est présent dans la configuration présentée. On a en moyenne *5.82* interactions problématiques par composant malicieux mais étant donné que, dans notre simulation, une configuration logicielle peut contenir plusieurs composants malicieux, il faut nuancer ce chiffre. Quoiqu'il en soit, dans le pire des cas, il a fallu dans ce cas précis 11 mauvaises interactions pour isoler un composant logiciel. C'est là un résultat tout à fait satisfaisant d'autant qu'il servira pour toutes les éventuelles configurations contenant ce composant logiciel.

4.3.7 Comparaison avec un modèle de réputation des nœuds

Pour compléter l'évaluation du modèle, nous allons illustrer ici sa comparaison avec un modèle de réputation des nœuds. Nous avons choisi SORI [5] que nous avons présenté en détail au chapitre II (Section 2.2.4). Quelques aménagements ont dû être faits pour rendre plus génériques les concepts de SORI. Le relais de paquets est ici considéré comme un service, i.e., une interaction effective à issue positive ou négative. La probabilité de rejet d'un paquet est donc assimilée à la probabilité de rejet d'une interaction telle que définie dans notre modèle. En reprenant les paramètres de la session 3 (voir encart ci-dessous) et avec un seuil δ de tolérance pour SORI tel que $\delta = 0.1$, on obtient les résultats présentés aux Figures 4.21 et 4.22. Rappelons que nous avons en abscisse, avec un pas de 100, les interactions (bonnes et mauvaises) d'un nœud et en ordonnée, la moyenne des indices de performance C-FN et C-FP sur tous les nœuds sains.

Session 3	
Nombre de nœuds	100
Taux d'infection	15 % : 15 nœuds corrompus, 85 sains
I-Faux Négatifs	25 %
I-Faux Positifs	2 %
Vitesse des nœuds	1 cellule

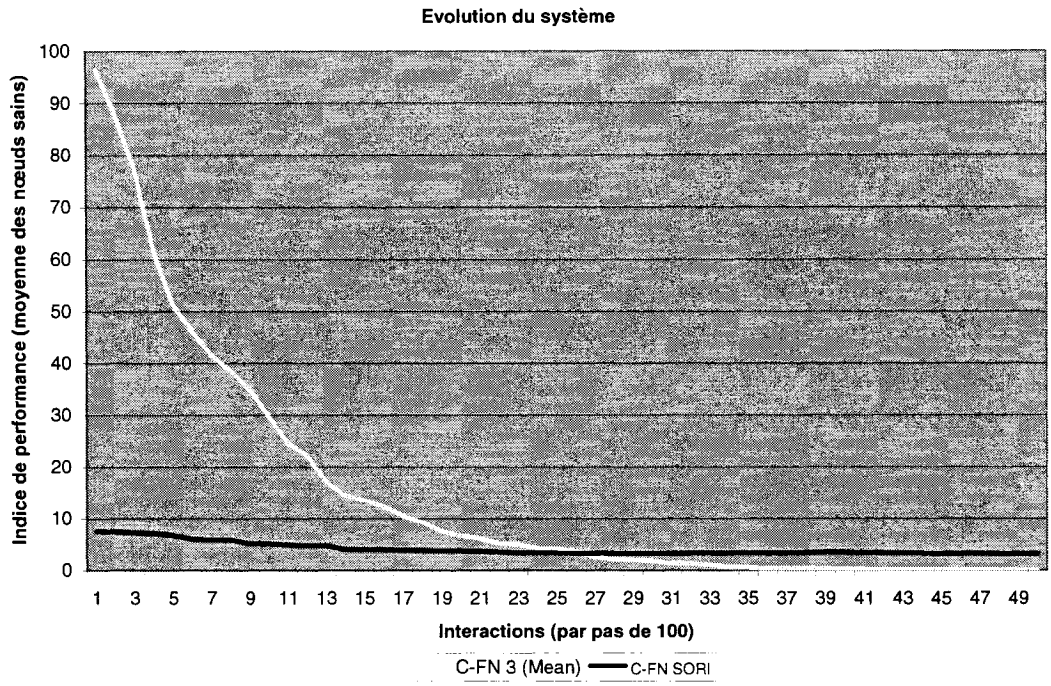


Figure 4.21 Comparaison avec SORI (C-FN)

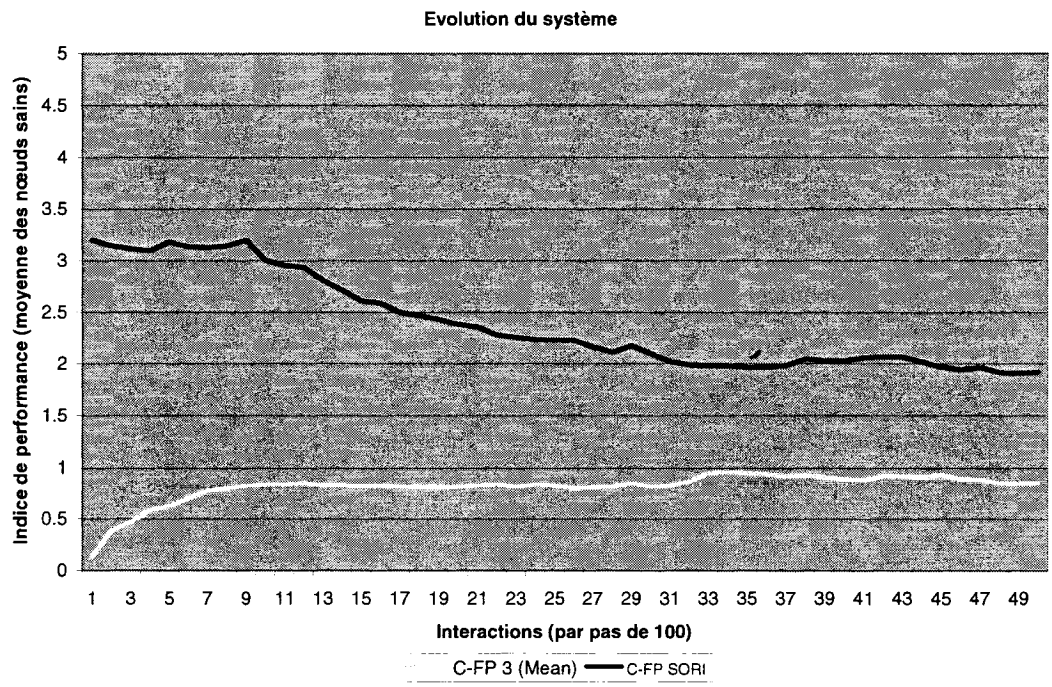


Figure 4.22 Comparaison avec SORI (C-FP)

On remarque sur la Figure 4.21 que SORI démarre avec un meilleur C-FN (autour de 8%) mais cet indice est très peu amélioré au fil de l'évolution du système et se stabilise autour de 3%. On a ainsi notre modèle qui rejoint SORI en moyenne après 2500 interactions et converge vers un C-FN nul. Pour ce qui est du C-FP, on constate sur la Figure 4.22 que SORI présente dès le départ des valeurs plus élevées que celles de notre modèle. Au final, environ 3% des interactions avec des nœuds corrompus restent non détectées avec SORI alors que le C-FN de notre modèle atteint 0%. De la même manière, le taux de rejets erronés avec SORI est autour de 2% tandis que notre modèle se stabilise à 1%. Mais pour mieux rendre compte des différences, on peut pousser l'expérimentation un peu plus loin.

Dans la comparaison avec un modèle de réputation de nœuds, deux paramètres sont particulièrement intéressants : le nombre de nœuds et la taille de l'environnement. En effet, ces deux paramètres influent sur la probabilité d'expériences communes entre deux nœuds quelconques du réseau. Lorsqu'on augmente les valeurs de ces deux paramètres, on diminue cette probabilité. Notons que pour un modèle de réputation des nœuds, un environnement trop étendu est clairement un handicap mais un nombre important de nœuds peut aussi être profitable puisqu'il augmente la densité de l'environnement et donc la probabilité d'échanges de rapports de réputation entre voisins.

Sur les figures 4.23 et 4.24, nous présentons les résultats des expériences effectuées en appliquant un coefficient multiplicateur de 4 pour les dimensions du terrain (largeur et longueur) et pour le nombre de nœuds.

Session 4x	
Terrain	4x plus grand
Nombre de nœuds	400
Taux d'infection	25 % : 100 nœuds corrompus, 300 sains
I-Faux Négatifs	25 %
I-Faux Positifs	2 %
Vitesse des nœuds	1 cellule

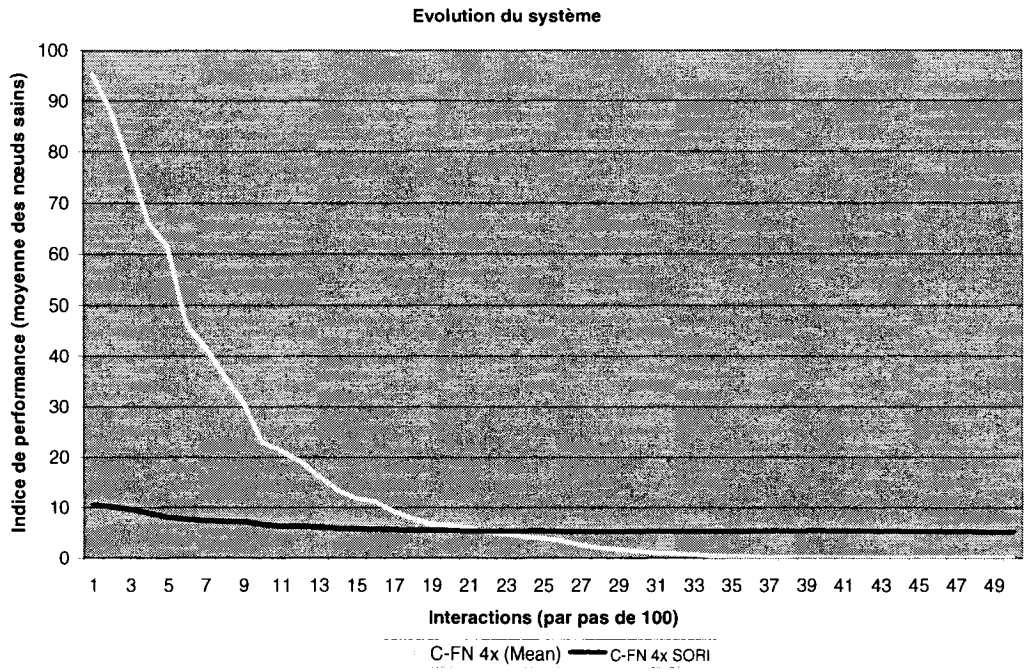


Figure 4.23 Session 4x : Comparaison avec SORI (C-FN)

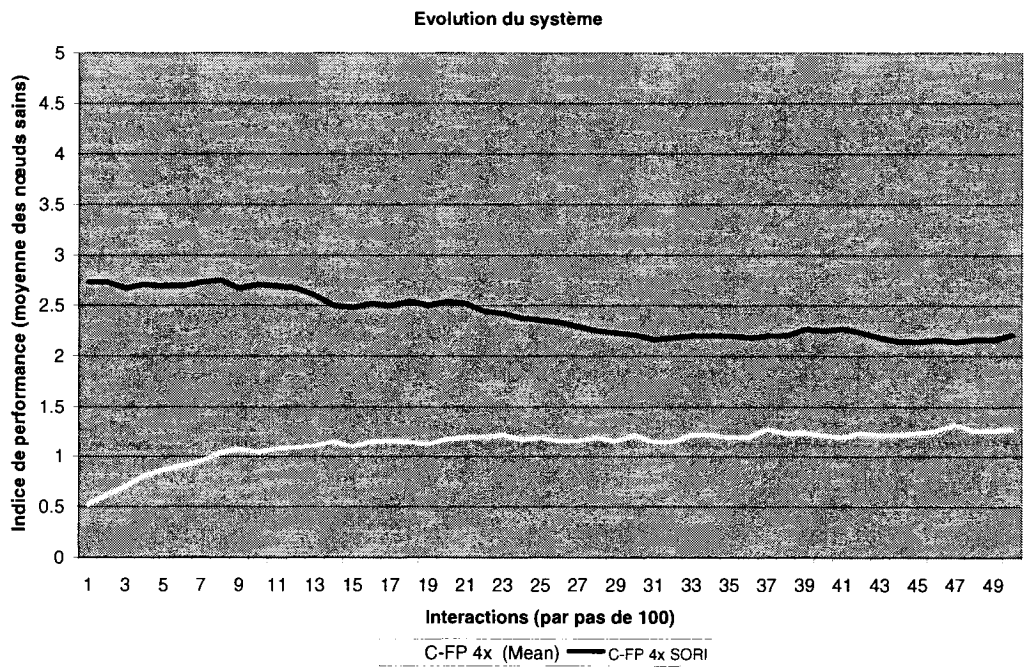


Figure 4.24 Session 4x : Comparaison avec SORI (C-FP)

On constate ici une dégradation notable des performances de SORI. On peut notamment observer sur la figure 4.23 que le C-FN se stabilise cette fois autour de 6%. Au niveau du C-FP de SORI, les différences avec les résultats de la session 3 ne sont pas significatives. Sur les deux figures, on note que notre modèle reste peu sensible à l'augmentation du nombre de nœuds et de l'étendue du terrain. Cela s'explique aisément par le fait qu'il ne s'intéresse qu'aux composants logiciels rencontrés dans l'environnement.

4.4 Synthèse des performances

À la suite de la présentation et de l'analyse détaillée des résultats, une synthèse s'impose pour situer la solution proposée. En l'absence dans la littérature [18] de scénarios d'expérimentation standards pour les MANET, il s'agit d'évaluer le modèle proposé par rapport aux indices de performance définis, indices dont les valeurs idéales sont parfaitement identifiées. En effet, dans l'idéal, le modèle ne devrait générer ni faux négatifs, ni faux positifs (i.e. C-FN = 0 et C-FP = 0). Les résultats obtenus rencontrent donc des objectifs d'efficacité globalement très satisfaisants, notamment sur la vitesse de détection et d'isolation des composants logiciels malveillants. Le pourcentage, en moyenne très faible, de faux positifs générés par le modèle reste cependant perfectible. En effet, certains nœuds atteignent des pourcentages (10%) de faux positifs que nous estimons raisonnables mais tout de même un peu élevés. À ce niveau, il faudrait pouvoir envisager sous conditions strictes une « amnistie » pour les composants logiciels bannis.

Dans tous les cas, le modèle proposé s'impose comme un premier pas prometteur dans une direction de recherche qui efface les limites inhérentes aux modèles de réputation des nœuds. Sous certaines hypothèses, notamment un grand nombre de nœuds et une grande mobilité dans le réseau ad hoc, le modèle de réputation des composants logiciels est intrinsèquement supérieur aux modèles de réputation des nœuds.

CHAPITRE V

CONCLUSION

Les réseaux mobiles ad hoc (MANET) constituent aujourd'hui des solutions pertinentes dans un nombre croissant de problématiques. Le scénario que nous avons privilégié dans notre projet est celui du V2V (Vehicule to Vehicule), technologie émergente dans laquelle les voitures sont équipées de senseurs et s'échangent toutes sortes d'informations sur leur environnement commun (conditions de route, signalement de positions respectives etc.). Nous sommes typiquement dans un cadre où la coopération des nœuds est essentielle au bon fonctionnement du réseau. Dans de tels environnements décentralisés, en plus d'éléments de cryptographie indispensables, les modèles de réputation de nœuds constituent une bonne approche pour évaluer les comportements des nœuds d'un MANET et inciter toutes ces entités à coopérer.

Parallèlement, les travaux du TCG permettent désormais de poser comme hypothèse que les nœuds peuvent se rapporter de manière fiable leurs mesures d'intégrité. L'interprétation de ces mesures reste un champ ouvert auquel nous avons apporté notre contribution. Le but de notre projet a été d'évaluer dans quelle mesure TCG pouvait être utilisé pour définir ou améliorer la confiance entre différentes plates-formes d'un MANET.

5.1 Synthèse du travail accompli

Dans ce projet, nous avons changé la perception d'un nœud dans un MANET qui dépasse sa représentation schématique de « cercle opaque » pour reprendre sa dimension de système autonome et complexe constitué de composants logiciels divers. Si l'on fait confiance au TPM pour rapporter les mesures d'intégrité de ces composants, nous pouvons utiliser ces données pour construire la confiance que s'accordent les nœuds entre eux. D'une réputation des nœuds, nous passons ainsi à une réputation des

composants logiciels rencontrés. Le constat effectué est le suivant : dans un environnement où les nœuds sont très nombreux et où les chances d'expériences communes antérieures entre deux nœuds quelconques sont réduites, l'efficacité d'une réputation des nœuds est discutable. Les composants logiciels seront par contre mieux distribués au niveau des nœuds, favorisant ainsi l'obtention de l'information nécessaire pour l'établissement de la confiance entre nœuds. Cette information pourra servir même lorsque le nœud se retrouve dans un environnement complètement différent (d'une ville à une autre).

Dans notre modèle, les nœuds stockent des informations sur les composants logiciels rencontrés pour les associer aux bonnes ou mauvaises expériences effectuées. Ces informations de réputation sur les composants logiciels sont périodiquement diffusées aux autres nœuds du système. A leur entrée dans le système, les composants logiciels se voient d'abord accorder un statut de *nouveau*, statut peu contraignant au début dans la mesure où la quasi-totalité des composants logiciels ont ce statut et ne présentent majoritairement aucun comportement répréhensible. Au fur et à mesure de l'évolution du système, un nœud ayant une configuration contenant un composant logiciel *nouveau* aura de plus en plus de mal à interagir avec les autres jusqu'à ce que ce composant soit jugé *normal* de par ses interactions positives. Un composant logiciel *nouveau* ou *normal* peut transiter vers un statut de *suspect* s'il dépasse un certain seuil (statique ou relatif) de mauvaises statistiques. Des pénalités pourront désormais lui être infligées s'il est identifié comme le « pire » composant impliqué dans une mauvaise interaction. A un certain seuil de pénalités, le composant logiciel est *banni* sans rémission et toute configuration le contenant n'est plus acceptable.

Pour tester notre modèle, nous avons développé un environnement de simulation multi agents épuré de certaines considérations « réseau » mais assez complet sur la vie interne et logicielle des nœuds. Cette simulation nous a permis d'évaluer notre modèle sur le pourcentage de faux négatifs et de faux positifs qu'il génère.

Les résultats obtenus nous paraissent très prometteurs, d'autant que les conditions initiales choisies rendent compte d'un environnement assez fortement hostile

dès le départ. Les composants logiciels malveillants sont rapidement isolés et ce, avec très peu d'erreurs sur la validité des composants sains.

5.2 Limites de l'approche proposée

L'approche adoptée perdra en pertinence dans des environnements faisant intervenir un petit nombre d'entités. En effet, si les composants logiciels actifs sont beaucoup plus nombreux que les entités des environnements visés, une réputation des nœuds paraît mieux indiquée. Plus précisément, si le nombre de composants logiciels est plus grand que le nombre de nœuds de l'environnement global considéré, le recours à notre modèle est discutable. Mais lorsque nous parlons d'environnement, il faut bien souligner qu'il s'agit ici de l'ensemble des nœuds avec lesquels un nœud donné pourra interagir à un moment donné. Dans le cas des VANET, il s'agira, le plus souvent, au minimum du parc automobile d'une ville. Le problème est donc moins préoccupant. Par contre, pour des entités évoluant en petit nombre dans un milieu fermé, il faudra s'intéresser de près aux cardinalités respectives de l'ensemble des entités et de l'ensemble des composants logiciels. Rappelons tout de même que nous avons, en connaissance de cause, effectué nos expériences en faisant intervenir en moyenne 1800 composants logiciels pour 100 nœuds et que les résultats restent satisfaisants quoique le système démarre plus lentement qu'un modèle de réputation des nœuds. Le gain d'informations pertinentes reste conséquent et utile quant à l'introduction de nouvelles entités dans l'environnement.

Par ailleurs, si l'environnement logiciel considéré est limité et maîtrisé, il sera plus aisé et bien évidemment préférable d'opter pour un modèle plus strict où tout composant logiciel inconnu serait d'office banni. En effet, pour un nombre trop réduit de composants logiciels ou pour des configurations logicielles fortement homogènes, il sera bien plus sécuritaire de se donner les moyens de recenser les composants logiciels présents et de traiter sans approximations les configurations présentées. Lorsqu'on met en œuvre un modèle de réputation, il faut être conscient de ses limites. L'hypothèse de

base selon laquelle le comportement antérieur d'une entité est un bon indicateur de son comportement futur ne sera pas toujours vérifiée.

De plus, lorsqu'on applique un modèle de réputation à des composants logiciels, il faut éviter certains pièges plus pernicieux. En effet, la question de la discrimination se pose ici très vivement. Outre les discriminations facilitées par le rapport d'intégrité de TCG, il faut pointer très précisément le risque des faux positifs de notre approche. Lorsqu'on fait l'analogie avec une société d'humains, ce que nous avons là est un système où les individus sont jugés suivant une de leurs caractéristiques. L'histoire de la science, en la matière, n'est pas exempte de certaines turpitudes pseudo-scientifiques. Cette analogie a toutefois ces limites dans la mesure où nous travaillons ici avec des entités logicielles dont les comportements sont essentiellement liés à leur configuration logicielle.

5.3 Travaux futurs

- Notre modèle de réputation suppose un mécanisme d'évaluation des interactions qui peut être plus ou moins coûteux (temps, ressources, etc.). Il pourrait être très intéressant de *réduire les recours à ce mécanisme d'évaluation* tout en prenant garde à ne pas ainsi donner prise à des stratégies malicieuses particulièrement élaborées.
- Le modèle que nous avons implémenté est uniquement axé sur la gestion des composants logiciels mais nos résultats d'expérimentation ont révélé une piste complémentaire. En effet, la *gestion des identités* des nœuds est une considération pertinente. Il s'agirait de gérer convenablement les interactions avec un même nœud sur une courte période. Plus généralement, notre modèle pourrait être, sous certaines conditions, *combiné à un modèle de réputation des nœuds*.
- Enfin, les premiers standards pour les communications de voiture à voiture ont été très récemment validés et nous n'avons pu les intégrer dans notre modèle. Il y a là de toute évidence matière à approfondissement.

BIBLIOGRAPHIE

[1] BUCHEGGER S. et LE BOUDEC J-Y. "A robust reputation system for peer-to-peer and mobile ad-hoc networks" In Proceedings of P2PEcon, Juin 2004.

[2] BUCCHEGER S., LE BOUDEC J-Y. "Self-Policing Mobile Ad-Hoc Networks by Reputation Systems" *IEEE Communications Magazine*, Vol. 43, No. 7, pp. 101-107, Juillet 2005

[3] DELLAROCAS C. "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior", ACM Conference on Electronic Commerce, Minneapolis, MN, pp. 150–157, 2000

[4] FRIEDMAN E. et RESNICK P. "The Social Cost of Cheap Pseudonyms" *Journal of Economics and Management Strategy* Vol 10, No. 2, pp. 173-199, 2001

[5] HE Q., WU D. et KHOSLA P., "SORI: A secure and objective reputation-based incentive scheme for ad hoc networks" IEEE Wireless Communications and Networking Conference, Vol 2, pp. 825- 830, Mars 2004.

[6] JOHNSON D. B., MALTZ D. A., et BROCH J., "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks in Ad Hoc Networking" *Ad Hoc Networking* Chapter 5, pp. 139-172, Addison-Wesley, 2001

[7] KARGL F., SCHLOTT S., KLENK A., GEISS A., Weber M., "Securing Ad hoc Routing Protocols" In Proceedings of 30th Euromicro Conference, pp. 514-519, Septembre 2004

[8] KURKOWSKI S., CAMP T., NAVIDI W., "Two Standards for Rigorous MANET

Routing Protocol Evaluation" Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference, pp. 256-266, Octobre 2006

[9] MARTI S., GIULI T. J., LAI K., et BAKER M., "Mitigating routing misbehavior in mobile ad hoc networks" In Proceedings of MOBICOM 2000, pp. 255–265, 2000

[10] MICHIARDI P. et MOLVA R. "CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks", Sixth IFIP conference on security communications and multimedia, CMS 2002, pp. 107-121, Septembre 2002

[11] OOI B. C., LIAU C. Y. et TAN K-L, "Managing Trust in Peer-to-Peer Systems Using Reputation-Based Techniques" In Proceedings of International Conference on Web Age Information Management, pp. 2-12, Août 2003

[12] PAUL K., WESTHOFF D., "Context aware detection of selfish nodes in DSR based ad-hoc networks" In Proceedings of Vehicular Technology Conference, 2002. pp. 2424 – 2429, Automne 2002

[13] SAILER R., ZHANG X., JAEGER T., VAN DOORN L., "Design and Implementation of a TCG-Based Integrity Measurement Architecture" Computer Science IBM Research Report, Janvier 2004

[14] SAILER R., VAN DOORN L., WARD J. P., "The Role of TPM in Enterprise Security" Computer Science IBM Research Report, Octobre 2004

[15] XIONG L. et LIU L. "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities", *IEEE Transactions On Knowledge And Data Engineering*, Vol. 16, No. 7, pp. 843 - 857, Juillet 2004

[16] « Communications V2V et V2I », Mars 2007 www.autonews.fr/fr/cmc/dossier/200713/communications-v2v-et-v2i_8367.html

[17] "Intelligent Transportation Systems Standards Fact Sheet", 2006, www.standards.its.dot.gov/fact_sheet.asp?f=80

[18] "Madkit : Multi Agents Development Kit", Septembre 2006 www.madkit.org

[19] TCG Specification Architecture Overview, Revision 1.3, Mars 2007, www.trustedcomputinggroup.org/groups/TCG_1_3_Architecture_Overview.pdf