UNIVERSITÉ DE LIMOGES
ÉCOLE DOCTORALE "Sciences et Ingénierie pour l'Information"
FACULTÉ DES SCIENCES ET TECHNIQUES

Département de Mathématiques et Informatique - Laboratoire XLIM
Thèse N°

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline / Spécialité : Informatique

présentée et soutenue par

Mohamed Amine RIAHLA

le 06/03/2013

Contributions au routage et l'anonymat des échanges dans les réseaux dynamiques

Thèse dirigée par **Philippe Gaborit et Karim Tamine**

JURY

Président
M. Jean Pierre CANCES Professeur à l'université de Limoges

Rapporteurs
M. Christophe BIDAN Professeur à SUPELEC, HDR de l'université de Rennes I
M. Alexis BONNECAZE Professeur des universités, Aix-Marseille Université

Examinateurs
M. Carlos AGUILAR HDR à l'université de Limoges
M. Gentian JAKLLARI Maitre de Conférences INP - ENSEEIHT (Toulouse)
M. Philippe GABORIT Professeur à l'université de Limoges
M. Karim TAMINE Maitre de conférences à l'université de Limoges

Remerciements

Je tiens à remercier, Monsieur TAMINE KARIM, pour ses conseils judicieux, sa grande disponibilité et les précieuses discussions que nous avons eu ensemble. Je lui exprime ma profonde gratitude pour m'avoir fait profiter de ses connaissances, mais aussi de ses méthodes de travail. Grâce à lui, j'ai découvert un domaine de recherche qui aujourd'hui me passionne.

J'adresse mes plus vifs remerciements à mon encadreur Monsieur P. GABORIT Professeur à l'université de limoges, pour la confiance qu'il m'a accordée, ainsi que pour toute l'aide et tous les encouragements qu'il m'a apporté.

Je remercie également et pleinement M. Christophe BIDAN HDR de l'Université de Rennes 1 et Monsieur Alexis BONNECAZE Professeur des universités (Aix-Marseille) qui m'ont fait l'honneur d'accepter de juger ce modeste travail ainsi que tous les membres du jury.

Je remercie infiniment ma femme Sihem qui m'a toujours soutenue (moralement et physiquement) dans les moments difficiles et qui a toujours su trouver les méthodes pour me réconforter.

Enfin je souhaite aussi témoigner toute mon amitié à l'ensemble de mes collègues de travail à l'université de BOUMERDES et à l'entreprise.

Dédicace

A la mémoire de mes grands-parents paternels

A mes grands-parents maternels

A celle qui m'est plus chère au monde, qui m'a soutenue durant toute ma vie grâce à son amour et son affection, que Dieu te bénisse maman

A mon très chère père qui grâce à ses sacrifices, je suis devenu ce que je suis aujourd'hui

« J'espère que j'ai tenu ma promesse »

A mes chers frères Yacine, Samir, Smail, Mounir, à ma chère sœur Amel

A ma chère femme Sihem

A mes oncles et mes tantes

A tous mes cousins et cousines

A toute la famille de Karim TAMINE : Maya, Nassim et Melissa ainsi que ses parents et beaux parents

A tous mes amis

Je dédie ce mémoire

Résumé

L'évolution rapide des technologies d'internet a permis l'émergence de nouveaux types de réseaux très dynamiques avec des architectures fortement décentralisées et dont les services sont organisés de manière autonome. Ces spécificités ont un réel avantage : le déploiement et la mise en place rapide et peu couteux de ce type de réseaux. Mais en contrepartie, ces réseaux peuvent engendrer des difficultés lors de la mie en œuvre de certains services tel que le routage et la sécurité en général.

Dans le cadre de cette thèse, nous avons pris comme cas d'étude les réseaux P2P et les réseaux ad hoc. Nous proposons dans ce mémoire un ensemble de contributions que nous classons en deux parties :

- L'anonymat des échanges dans un réseau P2P.
- Le routage dans les réseaux ad hoc.

La première partie de cette thèse a porté sur les problèmes de confidentialité et d'anonymat des échanges d'informations dans les réseaux P2P. Les systèmes Pair-à-Pair (P2P) sont en plein essor depuis plusieurs années. Ce paradigme permet de concevoir des réseaux de grande taille, à forte disponibilité, à faible coût et sans recourir à une infrastructure centrale. Les réseaux P2P sont des réseaux virtuels construits au-dessus des réseaux physiques, décentralisés, où tous les nœuds, appelés pairs (peer en anglais), jouent à la fois le rôle de serveur et de client. L'organisation dans un tel réseau repose sur l'ensemble des pairs et par conséquent il n'y a aucune entité chargée d'administrer le réseau.

Les réseaux P2P sont essentiellement utilisés par un bon nombre d'utilisateurs d'internet pour échanger des ressources diverses (fichiers, partage d'applications, etc.). Les réseaux P2P sont aussi utilisés pour des applications collaboratives où un grand nombre d'utilisateurs doivent pouvoir travailler sur les mêmes objets (documents, tables, etc.). Ces échanges s'opèrent au travers de canaux de communications non sécurisés et par conséquent il faut prendre en compte la possibilité d'écoute et de modification des messages échangés.

Dans cette thèse, nous nous sommes intéressés au problème du maintien de la confidentialité et de l'intégrité des ressources échangées par les utilisateurs d'un réseau P2P non structuré ainsi qu'à la préservation de l'anonymat de ces mêmes utilisateurs lors des différents échanges : protection de l'anonymat du fournisseur de la ressource, de l'anonymat du demandeur de la ressource et enfin de l'anonymat mutuelle entre le fournisseur et le demandeur de la ressource. Nous avons dans un premier temps étudié les principaux systèmes P2P qui se sont penchés sur ces problématiques (Mute [Cho06]

[Cho07], Freenet [CSWH01] [CHMSW02], OneSwarm [IPKA10] et Ants [CC05]). Afin de répondre en partie à certains problèmes et limites de ces systèmes, nous avons proposé un nouveau système de partage de ressource Pair à Pair (P2P) qui intègre la propriété de confidentialité et d'intégrité des ressources échangées entre utilisateurs ainsi que la préservation de l'anonymat de ces derniers lors des échanges. Le nouveau protocole fonctionne à la façon d'un système Multi Agents mobiles s'adaptant bien au caractère dynamique d'un réseau P2P et permet aussi de minimiser l'impact des propriétés de sécurité sur les performances et la robustesse du système. Les résultats de ce travail ont donné lieu à une publication dans la conférence "Sciences of Electronic, Technologies of Information and Telecommunications" [RTG12] ainsi que la soumission d'un article à la revue « Peer-To-Peer Networking and applications ».

La deuxième partie de la thèse a porté sur les problèmes de routage dans les réseaux ad hoc. Le nombre croissant des terminaux (Pdas, ordinateurs, téléphones) et l'expansion des réseaux sans fil témoignent de la demande croissante des utilisateurs pour un accès à des ressources (données ou services applicatifs) à tout moment et n'importe où. Dans ce cas, les réseaux ad hoc peuvent jouer un rôle important puisqu'ils ne nécessitent pas d'infrastructure pour communiquer. Les méthodes de routage proposées dans les réseaux avec infrastructure ne peuvent pas être utilisée dans le cas des réseaux ad hoc en raison de l'aspect dynamique de ce type de réseau induit par le changement de topologie dû à la mobilité des nœuds, et/ou par l'épuisement des réserves énergétiques des batteries de ces mêmes nœuds. Les solutions proposées dans la littérature pour les réseaux ad hoc se basent sur des techniques de diffusion d'informations de contrôle pour le calcul de routes, qui conduisent à surcharger le réseau. Les protocoles de routage existants se regroupent en trois grandes familles : les protocoles réactifs, qui permettent d'établir des routes à la demande, les protocoles proactifs qui établissent des routes à travers l'échange régulier des paquets de contrôle (qui sont généralement des tables de routage) afin de s'adapter aux changements de la topologie du réseau, et enfin les protocoles hybrides qui utilisent les avantages des deux approches. Nous avons étudié les principaux protocoles de routages existants et associés aux trois familles citées précédemment, il s'agit des protocoles DSDV [PB94], OLSR [OLS03], DSR [DBDAJ01], AODV [PBRD03], ZRP [Haa97] et AntHocNet [DCDG04].

Dans cette thèse, nous présentons un nouveau protocole de routage dédié aux réseaux ad hoc. Le protocole de routage proposé permet de réduire considérablement le coût de la surcharge réseau relativement aux autres protocoles. Il fonctionne à la manière d'un système Multi Agent mobiles et dont les agents collaborent pour l'établissement de « bonnes » routes. L'algorithme utilisé pour le calcul des routes, est complètement distribué et s'inspire fortement du fonctionnement d'une colonie de fourmis. Le caractère autonome des agents ainsi que leur mobilité aléatoire dans le réseau s'adapte

parfaitement à l'aspect dynamique d'un réseau ad hoc. Les résultats obtenus ont donné lieu à une publication dans la revue" WIRELESS NETWORKS " [ART12].

La majorité des protocoles de routage proposés pour les réseaux ad hoc supposent que tous les nœuds collaborent pour « router » les messages des autres nœuds (messages de contrôle ou messages de données), ce qui n'est pas toujours le cas dans un réseau où les ressources énergétiques et les puissances de calcul de ces nœuds sont limitées. Certains nœuds peuvent tout simplement refuser de router les messages des autres nœuds afin d'économiser leurs ressources énergétiques : On dit, dans ce cas, que les nœuds ont un comportement égoïste. Le comportement égoïste de quelques nœuds du réseau suffit pour perturber le bon déroulement du protocole de routage (on risque par exemple dans ce cas d'avoir beaucoup de paquets perdus). Pour cela, nous avons doté notre protocole de routage d'un mécanisme de renforcement de la coopération entre les nœuds. Le mécanisme proposé permet de « punir » les nœuds qui ne coopèrent pas durant les phases de calcul de routes, en les isolants du réseau s'ils persistent à rester égoïstes. Il permet aussi de « réinsérer » dans le réseau tout nœud déjà puni et dont le temps de punition a expiré. Ce mécanisme, de « punition » et de « réinsertion » est totalement distribué et s'adapte donc bien au caractère autonome de chaque nœud du réseau ad hoc. Les résultats obtenus ont donné lieu à une publication dans la conférence "Sciences of Electronic, Technologies of Information and Telecommunications" [ARAT12].

Mots-clés : Anonymat, Cryptographie, Systèmes Multi-Agents, Réseaux P2P, Réseaux ad hoc, Algorithme ACO.

Abstract

The quick development of Internet technologies gave birth to new types of network architectures with highly dynamic and highly decentralized services that are organized independently. These characteristics have a real advantage: the deployment and implementation are rapid and inexpensive for this type of networks. But in return, these networks can lead to difficulties when setting-up some services such as routing and security in general.

In this thesis, we have taken as a case study P2P networks and ad hoc networks. We propose in this paper a set of contributions that we classify into two parts:

- The exchange anonymity in a P2P network.
- Routing in an ad hoc networks.

The first part of this thesis focused on the issues of confidentiality and anonymity of information exchange in P2P networks. Peer-to-Peer (P2P) systems are booming for several years. This paradigm allows the design of large-scale networks, high-availability, low cost and without resort to a central infrastructure. P2P networks are virtual networks constructed on top of physical networks, decentralized, where all the nodes, called peers, playing the role of both client and server. The organization in such a network based on all peers and consequently there is no single entity responsible for managing the network.

P2P networks are essentially used by many Internet users to share various resources (files, application sharing, etc.). P2P networks are also used for collaborative applications where a large number of users need to work on the same objects (documents, tables, etc.). These exchanges take place through unsecured communication channels and consequently must take into consideration the possibility to read and modify exchanged messages.

In this thesis, we are interested in the problem of maintaining the confidentiality and integrity of the resources exchanged by users of unstructured P2P networks as well as preserving the anonymity of users in these various exchanges: protecting the anonymity of the resource provider, the anonymity of the resource requester and finally the mutual anonymity between the provider and the requester of the resource. We initially studied the major P2P systems that have handle these problems (Mute [Cho06] [Cho07], Freenet [CSWH01] [CHMSW02] OneSwarm [IPKA10] and Ants [CC05]). In order to solve some problems and limitations of these systems, we have proposed a new system of resource sharing Peer to Peer (P2P) that integrates the property of confidentiality and integrity of resources exchanged between users and the preservation of anonymity in these exchanges. The new protocol

works as a multi-agent system adapting well to the dynamic characteristic of a P2P network and can also minimize the impact of security properties on the performance and robustness of the system. The results that we present in this area have led to a publication in the conference "Sciences of Electronic, Technologies of Information and Telecommunications" [RTG12] as well as the submission of an article in the journal "Peer-To-Peer Networking and applications."

The second part of the thesis focused on routing issues in ad hoc networks. The increasing number of devices (PDAs, computers, mobiles) and the expansion of wireless networks reflect the growing demand from users to access resources (data and application services) at anytime and anywhere. In this case, ad hoc networks can play an important role since they require no infrastructure to communicate. Routing methods proposed in networks with infrastructure cannot be used in the case of ad hoc networks due to the dynamic aspect of this type of network induced by the topology change due to node mobility, and / or the depletion of batteries for these nodes. The solutions proposed in the literature for ad hoc networks are based on control information broadcasting technics to calculate routes that lead to overload the network. The existing routing protocols fall into three main categories: reactive protocols, which allow to build routes on demand, proactive protocols that establish routes through the regular exchange of control packets (which are usually routing tables), and then the hybrid protocols that use the advantages of both approaches. We studied the main existing routing protocols and associated with the three families mentioned above, that concern protocols DSDV [PB94], OLSR [OLS03], DSR [DBDAJ01], AODV [PBRD03], ZRP [Haa97] and AntHocNet [DCDG04].

In this thesis, we present a new routing protocol dedicated to ad hoc networks. The proposed routing protocol can greatly reduce the cost of network overhead compared with other protocols. It operates as a Multi-Agent System, whose mobile agents working for the establishment of "good" routes. The algorithm used for the calculation of routes is completely distributed and draws heavily on the functioning of an ant colony. The autonomous nature of agents and their mobility in the random network adapts to the dynamic aspect of an ad hoc network. The results led to a publication in the journal "WIRELESS NETWORKS" [Art12].

The majority of routing protocols proposed for ad hoc networks presume that all nodes collaborate together to "route" messages from other nodes (control messages and data messages), which is not always the case in a network where energy resources and computing power of these nodes is limited. Some nodes may simply refuse to route messages from other nodes to save their energy resources; in this case, we say that the nodes are selfish behavior. The selfish behavior of a few nodes in the network is enough to disturb the routing protocol (for example we may in this case have a lot of lost packets).To this end, we propose a new cooperation enforcement protocol which is integrated in our proposed routing protocol. The proposed mechanism can "punish" nodes that do not

cooperate during the phases of calculating routes in the network and isolate them if they continue to remain selfish. It also allows “reinsert” any node in the network already punished and whose time of punishment has expired. This mechanism of "punishment" and "reinsertion" is fully distributed and therefore fits well to the autonomous nature of each node in the ad hoc network. The results led to a publication in the conference "Sciences of Electronic, Technologies of Information and Telecommunications" [ARAT12].

Keywords: Anonymity, Cryptography, Multi-agent Systems, P2P networks, Ad Hoc Networks, ACO algorithm.

Table des matières

TABLE DES MATIÈRES IX
TABLE DES FIGURES XII
LISTE DES TABLEAUX XIV
1. INTRODUCTION GÉNÉRALE 1
2. LES RÉSEAUX P2P 10
2.1. LE CONCEPT DE RESEAU P2P 11
2.2. CARACTERISTIQUES DES RESEAUX P2P 11
2.3. CLASSIFICATION DES RESEAUX P2P 12
2.3.1. Les systèmes centralisés 13
2.3.2. Les systèmes décentralisés 13
2.3.3. Les systèmes P2P hybrides 17
2.4. LA SECURITE DANS LES RESEAUX P2P DECENTRALISES 18
2.4.1. Notions sur la cryptographie 18
2.4.2. Sécurité dans les réseaux et services P2P 22
2.5. CONCLUSION 24
3. PROTECTION DE LA VIE PRIVÉE ET ANONYMAT DES ÉCHANGES DANS LES RÉSEAUX P2P 25
3.1. NOTIONS GENERALES SUR L'ANONYMAT DES ECHANGES DANS LES RESEAUX 25
3.2. ANONYMAT DES ECHANGES DANS LES RESEAUX P2P : ETAT DE L'ART 27
3.2.1. Mute 27
3.2.1.1. Principe de fonctionnement 27
3.2.1.2. Analyse de performances et de sécurité du système 30
3.2.2. Ants 30
3.2.2.1. Principe de fonctionnement 30
3.2.2.2. Analyse de performances et de sécurité du système 31
3.2.3. Freenet 31
3.2.3.1. Principe de fonctionnement 31
3.2.3.2. Analyse de performances et de sécurité du système 33
3.2.4. OneSwarm 34
3.2.4.1. Principe de fonctionnement 34
3.2.4.2. Analyse de performances et de sécurité du système 36
3.3. NOTRE PROPOSITION : LE PROTOCOLE ANONYMP2P 36
3.3.1. Principe de fonctionnement du protocole 37
3.3.2. Fonctionnement détaillé du protocole 38
3.3.2.1. Notations et terminologie 38
3.3.2.2. Gestion de la connectivité 39
3.3.2.3. Recherche et transfert de données 39
3.3.2.4. La gestion des paquets perdus 44
3.3.2.5. La gestion des pertes de liens avec les voisins 44

3.3.3. Analyse des performances, de la sécurité et de l'anonymat des échanges dans AnonymP2P 45
3.3.3.1. Caractéristiques de AnonymP2P 45
3.3.3.2. Anonymat 46
3.3.3.3. Confidentialité et intégrité des données échangées 51
3.3.4. Simulation, tests et résultats 51
3.3.4.1. Mesures de performances (recherche d'information) 52
3.3.4.2. La diversité des routes 53
3.3.4.3. Résistance du protocole à l'attaque Man In The Middle 54
3.3.4.4. Anonymat du fournisseur 55
3.4. CONCLUSION 57
4. LES RÉSEAUX AD HOC 60
4.1. LES RESEAUX AVEC INFRASTRUCTURE 60
4.2. LES RESEAUX SANS INFRASTRUCTURE : LE CONCEPT DES RESEAUX AD HOC 62
4.2.1. Domaines d'application des réseaux ad hoc 63
4.2.2. Propriétés et spécificités des réseaux ad hoc 63
4.3. LA SECURITE DANS LES RESEAUX AD HOC 64
4.3.1. Modèles d'attaquant 64
4.3.2. Description des principales attaques spécifiques aux réseaux ad hoc 65
4.4. CONCLUSION 67
5. LE ROUTAGE DANS LES RÉSEAUX AD HOC 69
5.1. LE ROUTAGE DANS LES RESEAUX AD HOC : ETAT DE L'ART 69
5.1.1. Le routage proactif 71
5.1.1.1. Le protocole DSDV 72
5.1.1.2. Le protocole OLSR 72
5.1.2. Le routage réactif 73
5.1.2.1. Le protocole DSR 74
5.1.2.2. Le protocole AODV 75
5.1.3. Les protocoles de routage hybrides 77
5.1.3.1. Le protocole ZRP 77
5.1.3.2. Protocoles hybrides basés sur les colonies de fourmis 79
5.1.3.2.1. Fonctionnement des colonies de fourmis 79
5.1.3.2.2. Protocoles de routage basés sur les colonies de fourmis 80
5.1.3.2.3. Le protocole AntHocNet 82
5.2. NOTRE PROPOSITION : UN NOUVEAU PROTOCOLE DE ROUTAGE HYBRIDE POUR RESEAUX AD HOC 84
5.2.1. La découverte de voisinage 87
5.2.2. La découverte de routes 87
5.2.2.1. La phase proactive 87
5.2.2.2. La phase réactive 89
5.2.2.3. Fonctionnement détaillé des agents Ant 93
5.2.2.3.1. Notations et terminologie 93
5.2.2.3.2. Les champs transportés par l'agent Ant 94
5.2.2.3.3. Les algorithmes décrivant quelques comportements des agents Ant 95
5.2.3. La maintenance des routes 98

5.2.3.1. Fonctionnement détaillé de l'agent « RectifierAnt » 99
5.2.4. Simulations 101
5.2.4.1. Environnement de simulation 101
5.2.4.2. Les résultats de simulation 102
5.2.4.2.1. Le taux de perte de paquet 102
5.2.4.2.2. Délai de communication 105
5.2.4.2.3. La taille totale des messages de contrôle 107
5.3. CONCLUSION 109
6. ETUDE DE LA RÉPUTATION DES NŒUDS DANS LES RÉSEAUX AD HOC 110
6.1. SOLUTIONS POUR RENFORCER LA COOPERATION DES NŒUDS 110
6.2. NOTRE PROPOSITION : UN MODELE DE COOPERATION DANS LES RESEAUX AD HOC 113
6.2.1. Le module collecte d'information 115
6.2.2. Le calcul de la réputation 116
6.2.2.1. La note de réputation directe 116
6.2.2.2. La note de réputation externe 117
6.2.2.3. La note de réputation finale 118
6.2.3. La punition 118
6.2.4. Simulation 120
6.2.4.1. Graphes 1 : Le nombre de nœuds atteints par le nœud surveillé 122
6.2.4.2. Graphes 2 : Le nombre de voisin dans les routes d'un nœud surveillé 123
6.3. CONCLUSION 125
7. CONCLUSION ET PERSPECTIVES. 126
BIBLIOGRAPHIE 128

Table des figures

Figure 2-1- Principe de fonctionnement d'un système P2P centralisé 13
Figure 2-2- Principe de fonctionnement d'un système P2P décentralisé 14
Figure 2-3- Principe de fonctionnement du système P2P Gnutella 15
Figure 2-4- La recherche dans Gnutella 16
Figure 2-5- Principe de fonctionnement d'un système P2P hybride 17
Figure 2-6- le principe de chiffrement et de déchiffrement 18
Figure 2-7- Le chiffrement à clé secrète 19
Figure 2-8- Le chiffrement à clé publique 20
Figure 2-9- Attaque par déni de service distribué 23
Figure 3-1- Message envoyé du nœud S au nœud D 28
Figure 3-2- La réponse du nœud D au nœud S 29
Figure 3-3- La signature d'un fichier dans Freenet 32
Figure 3-4-La génération de la clé de recherche dans Freenet 32
Figure 3-5- Le principe de fonctionnement de Freenet 33
Figure 3-6- Le principe d'un réseau social utilisé dans AnonymP2P 38
Figure 3-7- Le processus de recherche dans AnonymP2P 41
Figure 3-8- La réponse et le transfert de la ressource 43
Figure 3-9- La gestion des pertes de liens avec les voisins 45
Figure 3-10- L'envoi de requêtes par AnonymP2P 47
Figure 3-11- L'anonymat du demandeur 48
Figure 3-12- L'attaquant P avec ses complices P1,…Pk testent si le nœud C partage un fichier 50
Figure 3-13- La nouvelle architecture de PeerSim 52
Figure 3-14- Pourcentage de requêtes satisfaites en fonction du nombre de nœuds 53
Figure 3-15- Pourcentage de requêtes satisfaites en fonction du nombre de requêtes 53
Figure 3-16- La moyenne des nombres de routes anonymes entre deux nœuds quelconques du réseau, en fonction du nombre de nœuds 54
Figure 3-17- Le pourcentage de réponses qui sont envoyées (par le fournisseur d'une ressource) au nœud voisin qui a relayé la requête en fonction du nombre de requêtes 56
Figure 3-18-Le pourcentage de réponses qui sont envoyées (par le fournisseur d'une ressource) au nœud voisin qui a relayé la requête en fonction du nombre de nœuds 56
Figure 4-1- Les réseaux sans fils avec infrastructure 61
Figure 4-2- L'architecture d'un réseau GSM 62
Figure 4-3- Les réseaux ad hoc 63
Figure 4-4- Attaque par trou de ver 66
Figure 5-1- Les familles de protocoles de routage dans les réseaux ad hoc 71
Figure 5-2- Le routage proactif 71
Figure 5-3- La diffusion optimisée de OLSR 73
Figure 5-4- Le principe de fonctionnement de DSR 74
Figure 5-5- Le format du message RouteRequest 76
Figure 5-6- Le format du message RouteReplay 76
Figure 5-7- Le principe de fonctionnement de ZRP 78

Figure 5-8- Le principe de fonctionnement d'une colonie de fourmis 80
Figure 5-9- L'idée de base de notre protocole 86
Figure 5-10- La phase aller d'un agent *Ant* proactif 88
Figure 5-11- La phase retour d'un agent *Ant* proactif 89
Figure 5-12- Influence des phéromones sur le déplacement des agents *Ant* 91
Figure 5-13- Rôle d'un agent *Ant* réactif 92
Figure 5-14- La maintenance de routes dans notre protocole 99
Figure 5-15- Pourcentage des paquets perdus en fonction du nombre de nœuds dans le réseau 103
Figure 5-16- Pourcentage des paquets perdus en fonction du nombre total de paquets de données ... 103
Figure 5-17- Pourcentage des paquets perdus en fonction de la vitesse (m/s) des nœuds 104
Figure 5-18- Délai de transmission (s) en fonction du nombre de nœuds dans le réseau 105
Figure 5-19- Délai de transmission (s) en fonction du nombre total de paquets de données 105
Figure 5-20- Délai de transmission (s) en fonction de la vitesse (m/s) des nœuds 106
Figure 5-21- La taille totale des messages de contrôle (octets) en fonction du nombre de nœuds dans le réseau 107
Figure 5-22- La taille totale des messages de contrôle (octets) en fonction du nombre total de paquets de données 107
Figure 5-23-La taille totale des messages de contrôle (octets) en fonction de la vitesse (m/s) des nœuds 108
Figure 6-1- Les modules de notre système de réputation 115
Figure 6-2- Le module collecte d'informations 116
Figure 6-3- Fonctionnement global de notre système de réputation 120
Figure 6-4- Le nombre de nœuds que le nœud surveillé atteint en fonction du temps (durée de simulation 100 secondes) 122
Figure 6-5- Le nombre de nœuds que le nœud surveillé atteint en fonction du temps (durée de simulation 200 secondes) 122
Figure 6-6- La variation du nombre de voisins dans la table de routage du nœud surveillé en fonction du temps (durée de simulation 150 secondes) 123
Figure 6-7- La variation du nombre de voisins dans la table de routage du nœud surveillé en fonction du temps (durée de simulation 180 secondes) 124

Liste des tableaux

Tableau 3-1- La table de routes anonymes de notre protocole 42

CHAPITRE 1

1. Introduction générale

Un réseau dynamique et distribué est constitué d'un ensemble de nœuds auto-configurables qui sont en constante évolution (le nombre de nœuds et de liens évolue au fil du temps). Le changement de topologie est aussi l'une des propriétés de ces réseaux, dû au fait que des nœuds du réseau peuvent rejoindre et/ou quitter le réseau de manière spontanée. L'avantage principal de ce type de réseaux est le déploiement et la mise en place rapide et peu couteux d'une telle infrastructure. Les réseaux dynamiques représentent aujourd'hui un défi pour le déploiement d'applications distribuées sur des machines complètement autonomes.

Mais en contrepartie, ces réseaux peuvent engendrer des difficultés lors de la mise en œuvre de certains services tels que le routage et la sécurité en général. Nous avons pris comme cas d'étude les réseaux P2P et les réseaux ad hoc.

A – Contexte réseau P2P

Depuis le début d'internet, le modèle client/serveur était le modèle de référence pour la mise à disposition des ressources. Dans ce modèle, le système repose sur un serveur dédié qui centralise et maintient l'ensemble des ressources et des services. Dès lors, l'augmentation du nombre d'utilisateurs exige un plus grand investissement des fournisseurs de service. Il est en effet nécessaire de garantir la disponibilité des ressources et des services, malgré le grand nombre de demandes simultanées. Pourtant l'augmentation du nombre de participants implique aussi que cet ensemble possède une forte ressource cumulée et une multitude de services : c'est ainsi qu'est né le paradigme Pair à Pair (Peer to Peer). Les réseaux P2P sont des réseaux virtuels construits au-dessus des réseaux physiques, décentralisés, où tous les nœuds, appelés pairs, jouent à la fois le rôle de serveur et de client. Il y a trois grands modèles de réseaux P2P : Les réseaux centralisés, décentralisés et hybride (ou super pairs). Dans la modèle décentralisé, on peut les classer suivant les réseaux non structurés et les réseaux structurés. Dans les réseaux non structurés chaque pair est complètement autonome et la propagation des requêtes (recherche de ressources) se fait par inondation (broadcast). L'avantage de ce type de modèle est la facilité de mise en place du réseau et de sa flexibilité. Les réseaux structurés organisent les pairs ainsi que la répartition des ressources sur les pairs, selon une structure stricte et efficace, notamment des tables de hachage distribuées (DHT) [SMKKB01]. Ces réseaux gagnent à la fois en efficacité et donnent des garanties de recherche. L'inconvénient est la perte d'autonomie de placement des données et le coût de maintenance qui peut être élevé. Les réseaux super-Pairs sont des réseaux

hybrides entre les réseaux non structurés et le client-serveur. Comme le réseau client-serveur, certains pairs, appelés super-pairs, jouent le rôle de serveur pour un ensemble de pairs et effectuent des fonctions complexes comme le traitement des requêtes, le contrôle d'accès et la gestion de la répartition des données. Les supers pairs sont organisés en mode P2P et choisis automatiquement en fonction de leurs caractéristiques (bande passante, vitesse de traitement, capacité mémoire, etc.).

Motivations et contributions

Le cadre de notre travail concerne les modèles P2P décentralisés et non structurés. Ces modèles P2P ouvrent de nouveaux horizons aux applications déployées. Ils permettent en effet de décentraliser la réalisation des services et de mettre à disposition des ressources partagées dans un réseau. Cette décentralisation présente de nombreux avantages qui repoussent les limites du modèle client-serveur tel que la tolérance aux pannes, le passage à l'échelle et la minimisation des coûts.

Cependant, le caractère distribué des services et l'absence d'entité centrale pour gérer les échanges entre les nœuds ont vu apparaître divers problèmes de sécurité qui peuvent avoir un impact important sur la popularité des réseaux P2P. Certaines attaques informatiques sont spécifiques aux réseaux P2P tel que l'attaque Eclipse [SNDW06] [SCDR04], à cause du caractère complètement distribué du réseau. Les réseaux P2P peuvent être aussi victime d'attaques génériques existants dans les réseaux classiques mais dont l'influence peut être plus importante lorsqu'il s'agit de réseaux P2P ; c'est le cas par exemple des attaques de type DDoS [NR06].

Mais la véritable motivation de notre travail dans le contexte des réseaux P2P a été d'étudier l'aspect préservation de la vie privée entre les différents communicants. Parmi les différents domaines de la protection de la vie privée dans les réseaux informatiques, l'anonymat des communicants est celui qui a été le plus étudié. Cependant, la majorité des réseaux P2P populaires sur internet négligent cet aspect de la protection de la vie privée en favorisant les performances et la facilité d'utilisation du réseau au détriment de la protection de l'anonymat des fournisseurs et des demandeurs de ressources ; c'est le cas des systèmes BitTorrent [IPKA07] [PIAKV07], et Gnutella[Ber03]. Pourtant, en assurer la protection de l'anonymat des communicants et de la confidentialité des échanges pourrait être un argument de taille pour inciter plus d'internautes à utiliser un réseau P2P et par conséquent augmenter les performances du réseau.

Nous avons étudié un ensemble de systèmes P2P ayant intégré dans leur mode de fonctionnement une certaine dose d'anonymat des communicants durant les échanges ainsi que le maintien de la confidentialité des ressources échangées ; il s'agit des systèmes Mute [Cho06] [Cho07], Ants [CC05], Freenet [CSWH01] [CHMSW02], et OneSwarm [IPKA10].

Dans Mute, chaque nœud du réseau possède une adresse virtuelle servant au routage des données dans le réseau. Un nœud appartenant à un chemin donné ne connait que l'adresse IP du nœud successeur immédiat (son voisin direct) et ne peut connaitre les adresses IP de la source ou la destination du message par ce qu'il ne « voit », dans sa table de routage et dans les entêtes qui circulent que leurs adresses virtuelles. L'identification des nœuds par des adresses virtuelles sert essentiellement à préserver l'anonymat du nœud source et de la destination. Le calcul des chemins pour le transfert des ressources entre les nœuds est réalisé au sein du réseau virtuel à l'aide un ensemble de paquets de contrôle dont le fonctionnement est inspiré de celui des colonies de fourmis. Cependant, dans Mute il est possible à nœud attaquant de briser la propriété d'anonymat d'un autre nœud ; il suffit, en effet de surveiller les communications de tous les voisins du nœud attaqué afin de déduire la correspondance de son adresse IP avec son adresse virtuelle.

Ants est un autre protocole étudié et qui présente beaucoup de similitudes avec Mute ; il utilise aussi un système d'identification des nœuds par des adresses virtuelles et réalise un calcul des chemins de demande et de transferts des ressources à l'aide d'un algorithme inspiré des colonies de fourmis. Le système Ants est doté d'une nouvelle fonctionnalité absente dan Mute et qui consiste à proposer un mécanisme pour assurer le chiffrement des paquets de bout en bout, c'est à dire entre le demandeur et le fournisseur de ressources. Pour se faire, les requêtes sont chiffrées grâce à un protocole de chiffrement asymétrique. Dans Ants tous les pairs intermédiaires ont accès au contenu de la requête de recherche mais seul le nœud demandeur peut déchiffrer les résultats de sa demande. Cependant, pour que le processus de chiffrement puisse se réaliser les nœuds demandeur et fournisseurs de ressources sont obligés de réaliser des échanges de clé de chiffrement. Dans un réseau distribué et sans autorité de certification tel que les P2P, cet échange est vulnérable face aux attaques de type Man In The Middle [OV03] du fait que le chemin suivi par la requête de demande de la ressource se trouve être le même que le chemin de transfert de cette même ressource. Lorsque l'attaque Man In The Middle réussit, l'attaquant peut ainsi connaître les contenus des ressources échangées (perte de confidentialité).

Dans cette thèse nous avons aussi étudié le système Freenet, un système P2P Anonyme pour le partage de fichiers et qui est très en vogue sur internet actuellement. Pour préserver en partie l'anonymat des communicants, Freenet reprend le principe de communication à travers un réseau virtuel (qui est au-dessus du réseau physique d'internet) qui consiste à éviter les communications directes entre les demandeurs et fournisseurs de ressources. Dans Freenet, chaque fichier est chiffré par une paire de clés asymétriques, la partie privée de la clé est utilisée pour signer le fichier, alors que la partie publique, concaténée et hachée avec un texte descriptif du fichier, représente la clé de la recherche pour identifier le fichier. Lorsqu'un nœud télécharge un fichier, le chiffrée de ce dernier sera dupliqué sur tous les nœuds intermédiaires le long du chemin menant vers le demandeur de ce fichier.

Ce procédé de duplication permet ainsi la disponibilité des fichiers échangés même si les propriétaires de ces fichiers ont « quitté » le réseau. Dans Freenet l'emplacement d'un fichier est donc non seulement pas unique, mais peut changer à tout moment. Bien que Freenet soit réputé comme efficace pour la protection de l'anonymat des nœuds, il présente néanmoins quelques limites. Un gros inconvénient de Freenet est le temps important pour la récupération d'un fichier dû essentiellement à son système complexe de diffusion de clé. En effet, dans Freenet il faut d'abord aller chercher la clé de recherche d'une ressource avant de lancer la recherche de la dite ressource. De plus, les clés de recherches sont hébergées par une infrastructure centrale ce qui va un peu à l'encontre du paradigme P2P.

Le dernier système P2P étudié dans ce manuscrit est OneSwarm. L'objectif du système est de fournir aux participants le contrôle explicite et paramétrable de leurs données. Chaque utilisateur peut partager ses données de manière publique ou anonyme. Les recherches et les transferts de données dans OneSwarm se font via un maillage de pairs fiables et non fiables, agrégés à partir de réseaux sociaux (Tor, FaceBook, etc.). Les concepteurs de Oneswarm pensent que cette combinaison de pairs fiables et non fiables donne de meilleurs résultats en termes de robustesse et de confidentialité. Les réponses aux requêtes de recherches dans OneSwarm se déplacent suivant le chemin inverse des requêtes de recherches correspondantes. Le nœud fournisseur envoie son message de manière différente selon que son voisin direct recevant la réponse est fiable ou pas. Dans le premier cas la réponse est immédiate alors que dans le deuxième cas elle sera différée pour simuler le retard qui correspond à un chemin plus long (1 ou 2 sauts) ; la valeur du retard est prise au hasard entre 150 et 300 ms. L'utilité de ce retard permet au nœud fournisseur d'empêcher un nœud voisin non fiable de déduire que son voisin soit le fournisseur d'une ressource ; en effet sans ce délai, il suffit au nœud non fiable de lancer une recherche et de mesurer le temps d'aller et retour de la recherche afin de déduire que le nœud voisin est la source de données, ce qui cause une perte d'anonymat du fournisseur. Cependant, ce retard peut selon nous, causer une diminution de performances du protocole. Pour limiter de saturer le réseau par les paquets de recherche, chaque nœud intermédiaire retarde le message de recherche d'au moins 150ms avant de le relayer vers le nœud suivant. Pour mettre fin à une recherche, le nœud origine envoie des messages dits « d'annulation » qui prennent le même chemin que la recherche mais sans retard pour atteindre rapidement les frontières de la recherche. Nous pensons cependant que cette méthode peut permettre à un attaquant de localiser la source de recherche par une étude fine du trafic ce qui peut causer une perte d'anonymat du demandeur de ressource.

Le travail effectué a permis l'étude d'un ensemble de systèmes P2P ayant intégré des fonctionnalités pour la préservation de la vie privée des communicants. Nous avons mis en avant les méthodes utilisées pour garantir l'anonymat des pairs et pour certains la garantie de la confidentialité

des ressources échangées. Ce travail a permis aussi de mettre en avant une synthèse des problèmes rencontrés par ces systèmes afin justement de préserver ces propriétés d'anonymat et de la confidentialité des ressources échangées.

Pour répondre en partie à ces problématiques, nous décrivons dans ce manuscrit la conception et l'expérimentation, d'un nouveau système P2P de partage de données permettant de protéger l'anonymat des demandeurs et des fournisseurs de ressources ainsi que le maintien de la confidentialité des ressources échangées. Pour le calcul et le maintien des chemins de recherche et de transfert des ressources, le système utilise un ensemble de messages de contrôle qui fonctionnent à la façon d'un système Multi Agent Mobiles. Nous pensons en effet que la notion d'agents mobiles collaboratifs s'adapte bien au concept de réseaux P2P décentralisés. Le nouveau protocole que nous proposons, tente aussi de réduire l'impact des échanges confidentiels et anonyme sur les performances du système.

Afin de maintenir les propriétés d'anonymat des communicants et de la confidentialité des échanges, plusieurs éléments sont mis en place: Tout d'abord le réseau P2P fonctionne sur le principe d'un réseau social virtuel. En effet, chaque nœud possède une identité virtuelle qu'il utilise lors d'une demande de ressource. Chaque nœud du réseau possède un nombre défini de voisins qu'il modifie aléatoirement dans le temps afin d'éviter à un attaquant de briser son anonymat en contrôlant ses voisins. Les recherches de ressources sont propagées sans inondation à l'aide d'Agents Mobiles circulants dans le réseau logique afin de diminuer la charge sur le réseau. Chaque nœud met à disposition des autres nœuds du réseau un ensemble d'Agents qui coopèrent pour localiser les données. En effet, chaque Agent peut transporter non seulement les demandes de son nœud créateur mais aussi les demandes des autres nœuds et établir plusieurs routes entre les demandeurs et les fournisseurs de ressources de manière totalement anonyme. Grace à ce procédé collaboratif des Agents mobiles, plusieurs routes vont être construites entre les participants du réseau, et cela aura pour effet d'avoir des téléchargements rapides et multi sources d'une part, et permet d'autre part de mieux résister à l'attaque Man In The Middle lors de la phase de l'échange des clés de chiffrement entre deux nœuds puisque le chemin de la demande sera différent des chemins pour le transfert de la ressource. Nos mesures effectuées à l'aide du simulateur PeerSim[1] montrent que notre protocole répond en partie aux problématiques soulevées lors de l'étude des systèmes existants. Ainsi, les performances de notre protocole en termes de localisation des ressources, supportent la comparaison avec les systèmes qui n'assurent ni la confidentialité des échanges ni l'anonymat tel que Gnutella [Ber03]. Ces résultats ont donné lieu à une publication dans la conférence " The Sixth International Conference Sciences of

[1] http://peersim.sourceforge.net

Electronic, Technologies of Information and Telecommunications" [RTG12] ainsi que la soumission d'un article à la revue « Peer-To-Peer Networking and applications ».

B – Contexte réseaux ad hoc

Un réseau ad hoc est un ensemble autonome et coopératif de nœuds mobiles qui se déplacent et communiquent par une transmission sans fil et qui ne suppose pas l'existence d'une infrastructure. Le réseau ad hoc se forme de façon spontanée et provisoire dès que plusieurs nœuds mobiles se trouvent à portée radio les uns des autres. Les nœuds communiquent la distance qui les séparent, par deux modes de communications : soit les nœuds mobiles peuvent directement communiquer (en transmission radio) car ils sont à portée de transmission, soit ils doivent utiliser d'autres nœuds mobiles comme des relais (ou routeurs) pour acheminer les paquets à destination (on dit que la transmission est Multi Sauts).

Le choix des éléments relais dans un réseau ad hoc mobile s'effectue grâce à un *protocole de routage.* Plusieurs protocoles de routages ont été proposés à savoir AODV [PB94], DSR [DBDAJ01], DSDV [PB94], OLSR [OLS03], ZRP [Haa97] et AntHotNet [DCDG04]. Les principales différences entre ces protocoles concernent leur mode de fonctionnement. AODV et DSR fonctionnent selon un mode réactif, c'est à dire qu'ils établissent une route lorsque le nœud souhaite émettre une donnée alors que OLSR et DSDV sont en mode proactif, c'est à dire que le protocole établit les routes dans le réseau avant qu'une donnée de transfert n'ait lieu. ZRP et AntHotNet fonctionnent selon un mode hybride (réactif et proactif) afin de bénéficier des avantages des deux modes.

Motivations et contributions

Au-delà de l'intérêt d'étudier le domaine des réseaux ad hoc, qui ont suscité une activité intense de recherche ces dernières années en raison de leurs caractéristiques fort différentes des réseaux filaires traditionnels, nous nous intéressons au routage des données qui s'adapte au contexte dynamique de ce type de réseau. De nombreux travaux et évaluations de performances effectuées dans le domaine du routage font apparaitre que l'intérêt d'un mode (réactif ou proactif) est lié à l'environnement du réseau. Ainsi, un protocole proactif est efficace dans les réseaux de grande taille avec une mobilité faible alors qu'un protocole réactif conviendrait mieux à un réseau de petite taille.

Nous souhaitons mettre en place un protocole de routage pour réseaux ad hoc qui utilise les avantages des modes réactifs et proactifs mais aussi qui tient compte de l'environnement réseau dans lequel le protocole est déployé (réseau dense, mobilité forte ou faible, etc.) afin d'optimiser au mieux les performances du réseau par rapport à des métriques comme le délai de transmission de données, le

nombre de paquets de données transmis et le nombre de paquets de contrôle nécessaires pour la construction des routes.

Dans cette thèse nous présentons un nouveau protocole de routage pour réseaux ad hoc fonctionnant suivant les modes réactifs et proactifs. L'idée novatrice de ce nouveau protocole, c'est qu'en mode réactif, la demande de route n'est plus diffusée par inondation dans le réseau (comme c'est le cas des protocoles fonctionnant en mode réactif) mais seulement déposée localement au niveau du nœud demandeur. Un ensemble d'Agents Mobiles est chargé de diffuser cette demande en unicast dans le réseau, où chaque Agent est généré périodiquement par chaque nœud du réseau. Chaque agent généré a pour « mission » première d'être en mode proactif en construisant des routes du nœud qui l'a généré vers d'autres nœuds du réseau (et vice versa) et cela sur un une distance en nombre de sauts fixée par le nœud créateur de l'agent (TTL). Le déplacement de l'agent de nœud en nœud est totalement aléatoire. Lorsqu'un agent transite par un nœud ayant déposé une demande de route, il se met en mode réactif, retourne vers son nœud créateur (en empruntant le chemin inverse) et en y déposant une information (phéromones) le long des nœuds traversés, correspondant à la demande de route. Lorsqu'un Agent en mode proactif transite par un nœud du réseau contenant de la phéromone correspondant à des demandes de routes, il choisit aléatoirement son prochain voisin de façon aléatoire et proportionnellement à la quantité de phéromones en direction de ce voisin. Cette façon de fonctionner de l'ensemble des agents permet ainsi d' « attirer » plus d'agents vers les nœuds demandeurs de routes, et augmenter ainsi les chances de satisfaire le plus rapidement possible la construction de ces routes.

Le protocole de routage n'établit pas nécessairement les routes les plus courtes, mais calcule plusieurs routes entre deux nœuds du réseau (protocole Multi Chemins), ce qui rend le protocole plus robuste aux changements de topologie et ouvre d'autre perspectives pour la sélection de routes selon d'autres critères (bande passante, ressource énergétique des nœuds, nœuds malveillants, etc.).

Les simulations réalisées de notre protocole (sous NS2) ont donné des résultats encourageants en le comparant aux performances des protocoles DSDV, AODV et AntHotNet. Les résultats obtenus ont donné lieu à une publication dans la revue" WIRELESS NETWORKS " [ART12].

Une autre motivation de notre travail dans le cadre des réseaux ad hoc, concerne une des tâches dont est chargé chaque nœud du réseau à savoir sa participation aux fonctions de « routage » dans le réseau. La majorité des protocoles de routage proposés pour les réseaux ad hoc supposent que tous les nœuds collaborent pour « router » (ou relayer) les messages des autres nœuds (messages de contrôle ou message de données), ce qui n'est pas toujours le cas dans un réseau où les ressources énergétiques et les puissances de calcul de ces nœuds sont limitées. Lorsqu'un nœud refuse de

« router » les messages des autres nœuds, on dit qu'il a un comportement égoïste. Dans un réseau ad hoc, il suffit de quelques nœuds égoïstes pour perturber le bon déroulement des communications dans le réseau (risque notamment de beaucoup de pertes de paquets). Un modèle de renforcement de la coopération entre les nœuds est donc indispensable pour minimiser la dégradation des performances du réseau ad hoc.

Il existe deux modèles de renforcement de la coopération : le premier est basé sur le calcul de la réputation des nœuds et l'autre est basée sur l'échange des crédits entre les nœuds. Nous avons étudié plusieurs modèles de coopérations tels que CONFIDANT [BLB02] [BB02, BBo02, BB04, BBo04], SORI [HWK04] et TOKEN BASED [YML02]. Nous avons doté notre protocole de routage d'un mécanisme de renforcement de la coopération entre les nœuds. Ce nouveau mécanisme s'adapte bien au caractère distribué et aléatoire lié au fonctionnement de l'ensemble des agents mobiles du protocole de routage. En effet, la prise de décision « collective » de « punir » un nœud du réseau dont le comportement est égoïste, est réalisée de façon totalement distribuée et autonome par les nœuds du réseau. Le mécanisme proposé permet aussi à tout nœud de pouvoir se « réinsérer » dans le réseau dès que son temps de « punition » aura expiré. Les simulations réalisées de ce mécanisme de renforcement de la coopération montrent clairement que les nœuds égoïstes sont écartés du réseau et peuvent être réinsérés s'ils reprennent leur rôle de « router » correctement les paquets. Ces résultats ont donné lieu à une publication dans la conférence " The Sixth International Conference Sciences of Electronic, Technologies of Information and Telecommunications " [ARAT12].

Partie 1 : Sécurité et anonymat dans les réseaux P2P

CHAPITRE 2

2. Les réseaux P2P

Les services d'internet sont conçus selon une architecture client/serveur. Le principe consiste à ce que chaque serveur puisse gérer les ressources et les services fournis aux autres nœuds du réseau (appelés clients). La centralisation des services se justifiait essentiellement par le fait que les nœuds clients n'étaient pas assez puissants (en ressources et en bande passante) pour fournir du contenu et/ou des services aux autres nœuds. Malgré l'amélioration des performances de ces serveurs, l'augmentation du nombre d'utilisateurs ainsi que la taille et le nombre de données transférées sur internet a montré les limites de ces architectures centralisées. Il s'est avéré nécessaire de disposer de modèles et d'architectures réseaux qui permettent d'augmenter la disponibilité des ressources avec l'accroissement de la demande. En d'autres termes avoir des modèles avec plus de capacités au fur et à mesure que le nombre de participant augmente. Comme les machines des clients offrent de plus en plus de capacité calculatoires, un modèle et une architecture semble répondre à cette exigence : Il s'agit des réseaux de type Pair à Pair (P2P).

Un réseau P2P est constitué d'un ensemble de nœuds (pairs) qui sont à la fois clients et serveurs. C'est sur ces nouveaux types de réseaux que repose aujourd'hui une grande partie des échanges sur internet. Ils ne se limitent pas seulement au partage de fichiers mais aussi à de nombreuses applications telles que la messagerie instantanée, la téléphonie ou le travail collaboratif d'une manière générale.

Cependant, la nature décentralisée de ces réseaux peut engendrer de nombreux problèmes de sécurité et de protection de la vie privée. Différentes analyses des risques ont permis de mettre en avant des attaques spécifiques aux réseaux P2P telles que l'attaque Sybil [Dou02] et l'attaque Eclipse [SNDW06] [SCDR04], et d'autres attaques qui sont génériques mais qui peuvent avoir plus d'impact dans le cas des réseaux P2P telle que l'attaque DDoS [NR06] ainsi que la propagation de programmes malveillants (vers, virus, spyware, etc.).

Dans ce qui suit, nous présentons les concepts de base caractérisant les réseaux P2P en faisant ressortir leurs propriétés, leurs avantages et inconvénients, leurs domaines d'application, les différentes architectures ainsi que les aspects liés à la sécurité de ce type de réseaux.

2.1. Le concept de réseau P2P

Les réseaux Peer to Peer (P2P) sont des systèmes distribués dont les éléments sont à la fois clients et serveurs. Ces réseaux tirent parti des ressources de tous les nœuds permettant ainsi l'utilisation efficace du réseau. La caractéristique essentielle d'un réseau P2P est qu'il n'y a pas d'entité qui contrôle le réseau ; les nœuds (Peer) sont autonomes et l'échange de ressources entre deux nœuds se fait directement entre ces deux nœuds, les autres nœuds peuvent servir comme « routeurs » pour assurer cet échange. L'autre caractéristique de ces réseaux est l'aspect dynamique de leur topologie due au fait que les nœuds peuvent rejoindre ou quitter le réseau de façon totalement aléatoire.

Au milieu des années 90, ces réseaux sont devenus populaires grâce à des applications de partage de fichiers (musique, films, etc.) telles que Emule[1], KaZaA [LKR04] ou BitTorrent [IPKA07] [PIAKV07]. Les réseaux P2P ont depuis peu permis l'émergence d'autres types d'applications tel que la téléphonie sur internet (le logiciel Skype [2][BS06][GDJ06] en est le plus populaire), le calcul distribué, la télévision par P2P, les réseaux sociaux distribués, les moteurs de recherches distribués, les programmes de messageries instantanées tel que ICQ[3], les visioconférences et les applications de collaboration d'une manière générale. On peut citer aussi le grand projet JXTA[4] dont le but est de construire un réseau dont chaque composant peut communiquer, collaborer et partager des ressources avec tous les autres indépendamment de leurs environnement d'exécution (langage de programmation, plateforme réseau et système d'exploitation), de leur architecture (ordinateur, téléphone portable, PDA, etc) et de leur emplacement , l'objectif du projet étant de construire un réseau virtuel P2P qui permet de cacher les contraintes liées au réseau physique.

2.2. Caractéristiques des réseaux P2P

Les réseaux P2P sont des réseaux décentralisés où tous les nœuds jouent à la fois le rôle de serveur et de client. En distribuant les données et les traitements sur tous les pairs du réseau, les systèmes P2P peuvent passer à très grande échelle sans recourir à des serveurs très puissants. Nous pouvons caractériser les réseaux P2P par ce qui suit :

La tolérance aux fautes : Dans l'architecture client/serveur la disponibilité d'un service repose intégralement sur celle du serveur. Si le serveur disparait, suite à une panne ou une attaque

1 http://www.emule-project.net/

2 http://www.skype.com

3 http://www.icq.com

4 http://jxta.kenai.com

informatique par exemple, alors le service qu'il fournit devient indisponible. Dans un contexte P2P, si un nœud disparait, le service continuera d'être fourni par les nœuds restants. Dans les réseaux P2P la disponibilité d'un service n'est plus liée aux nœuds individuellement mais à la communauté des nœuds qui composent le réseau P2P.

La dynamicité : Dans un réseau P2P, les nœuds peuvent se connecter ou quitter le réseau de manière libre et totalement aléatoire. Cette propriété permet au réseau P2P de maintenir une topologie dynamique.

L'auto-Organisation : A cause de l'absence d'un serveur central dans un réseau P2P, les nœuds sont auto-organisés. Le maintien de cette propriété d'auto-organisation nécessite par conséquent la mise en place de traitements spécifiques qui obligent les nœuds à exécuter un traitement global de façon décentralisé.

Un réseau virtuel : Les nœuds qui forment un réseau P2P définissent un réseau virtuel (appelé overlay) qui est construit au-dessus du réseau physique réel. Un lien dans un réseau P2P désigne une connexion directe entre deux nœuds dans le réseau virtuel du P2P. Par conséquent une communication entre deux nœuds voisins peut passer par plusieurs nœuds du réseau physique (routeurs). Généralement un tel réseau virtuel présente une propriété de transparence par rapport à plusieurs aspects du réseau. Le réseau virtuel permet, d'une part, de faire abstraction des différences de nature des nœuds (architecture, systèmes d'exploitation et langages différents) et d'autre part, de permettre une transparence sur le routage effectué au niveau sous-jacent : en effet deux voisins de la topologie virtuelle ne le sont pas forcément physiquement (ils peuvent être situés dans des espaces physiques et sur des réseaux différents). Le réseau virtuel rend ainsi transparent le routage effectué au niveau physique et aussi rend transparent l'accès, la localisation et la réplication de ressources.

2.3. Classification des réseaux P2P

La classification des réseaux P2P se fait selon la répartition des tâches entre les nœuds, les liens de communication entre ces nœuds ainsi que la procédure de localisation des ressources partagées par ces nœuds. Nous distinguons trois classes de systèmes P2P : les systèmes centralisés, les systèmes décentralisés et enfin les systèmes hybrides. Dans la suite de cette partie, nous présentons en détail les architectures liées à ces trois classes.

2.3.1. Les systèmes centralisés

Ces systèmes se basent sur un serveur central dont le rôle est de recenser les ressources partagées par les clients (une sorte d'annuaire, voir figure 2-1). Contrairement au système client/serveur classique, les ressources proposées par les clients ne se trouvent pas sur le serveur mais au niveau des clients propriétaires. Le serveur contient uniquement des informations concernant le client (nom, adresse, nombre de ressources partagées, type de connexion, numéro de port pour le téléchargement, etc.) ainsi que la liste des ressources qu'il partage. De ce fait, lorsque un nœud recherche une ressources, il fait sa demande en premier au serveur, ce dernier lui envoie alors la liste des ressources partagées ainsi que les informations sur les pairs possesseurs de ces ressources. Le client n'a plus qu'à se connecter « directement » vers ces pairs et initier la phase de transfert.

Cette famille de systèmes permet de localiser rapidement les ressources qui existent dans le réseau, puisque l'indexation des ressources est centralisée. Cependant, cette architecture possède quelques inconvénients : Le serveur est le maillon faible du système. Il suffit, en effet, que le serveur cesse de fonctionner (panne au niveau du serveur ou blocage suite à une attaque de type déni de service) pour que le fonctionnement du réseau soit inactif. Le serveur peut être aussi inondé par les requêtes de demande de ressources des clients ce qui exige au serveur d'effectuer beaucoup de recherches.

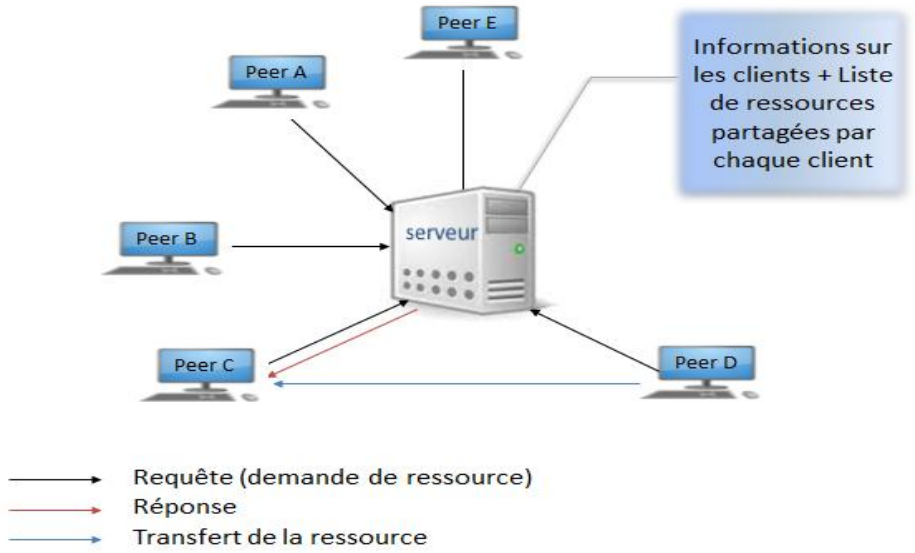


Figure 2-1- Principe de fonctionnement d'un système P2P centralisé

2.3.2. Les systèmes décentralisés

Dans ce type de systèmes, appelés aussi systèmes P2P purs, tous les nœuds jouent le même rôle (client et serveur). Chaque pair s'occupe de la sauvegarde et de l'indexation des ressources. Les nœuds jouent aussi le rôle de routeur pour les requêtes de recherche et de transfert des ressources.

Les nœuds du système forment un réseau dynamique appelé réseau logique (ou réseau virtuel) indépendamment de la topologie physique du réseau (cf. figure 2-2). Les pairs doivent donc collaborer pour assurer les processus de recherches, de routage et de récupération des objets dans le réseau logique. Les requêtes ne passent donc pas par un serveur central, mais à travers un sous ensemble des nœuds du réseau.

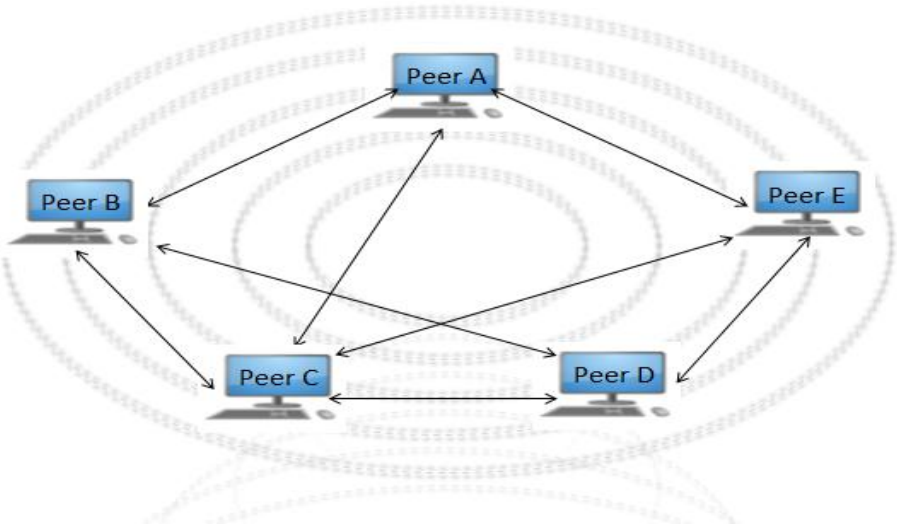


Figure 2-2- Principe de fonctionnement d'un système P2P décentralisé

Grace à leur caractère entièrement distribué, les systèmes décentralisés peuvent garantir le passage à l'échelle. Avec plus de nœuds dans le réseau (par conséquence plus de ressources répliquées), le système devient plus robuste et s'adapte bien à l'aspect dynamique des systèmes P2P. Cependant, ces systèmes soufrent de certains inconvénients. Par exemple, il est délicat de savoir quels sont les nœuds qui partagent des ressources de ceux qui sont uniquement des consommateurs de contenus. Un autre problème rencontré par certains systèmes décentralisés est l'utilisation de diffusion de requêtes pour la localisation des ressources demandées vu l'absence d'un index central de recherche, ce qui peut engendrer une grande consommation de bande passante.

La construction de la topologie du réseau logique peut se faire de deux façons différentes : Une connexion totalement aléatoire des nœuds au réseau : on dit alors que le système est décentralisé non structuré sinon on dit qu'il est structuré.

Les systèmes non structurés : Dans cette approche, les connexions et les liens entre les nœuds sont établis de façon totalement aléatoire. Les recherches de ressources s'effectuent généralement par des requêtes qui inondent le réseau. Pour assurer le routage dans un réseau dont on ignore la topologie, la recherche par inondation propose de retransmettre récursivement la requête de recherche à tous les voisins d'un pair et ce jusqu'à la localisation de la ressource recherchée ou l'expiration du nombre de sauts (TTL). Cette garantie d'acheminement est assurée même si la topologie change à la suite, par

exemple, de la défaillance de certains pairs. Dans cette catégorie de réseau P2P, nous pouvons citer le système Gnutella [Ber03] qui sera utilisé comme système cible afin de comparer ses performances en matière de recherche d'information avec celles du système *AnonymP2P* que nous avons développé dans le cadre de cette thèse (chapitre 3 section 3.3.4.1).

Gnutella utilise l'inondation de requêtes au sein du réseau logique afin de trouver les ressources recherchées. Comme tout autre système P2P non structuré, la topologie du réseau Gnutella est construite de façon aléatoire. Chaque nœud du réseau maintient un certain nombre de liens (connexions) avec d'autres nœuds (appelés voisins). Un nœud souhaitant se connecter au réseau Gnutella doit connaitre au moins un nœud déjà présent sur le réseau. Afin de s'insérer dans le réseau, un nœud N doit diffuser un message « PING » aux nœuds qu'il connaît (cf. figure 2-3) ; chaque nœud recevant ce message procède comme suit :

- Il répond au nœud N avec un message « PONG » qui contient son adresse, son numéro de port ainsi que d'autres informations telles que : le nombre et la taille des fichiers partagés.
- Il transfert le message du nœud N vers ses voisins, qui vont à leur tour réaliser le même processus.

Le nœud qui reçoit plusieurs messages PING provenant de la même source, traite le premier arrivé et ignore les autres. Grace aux messages PING/PONG, le nœud N peut donc trouver les adresses des autres nœuds du réseau.

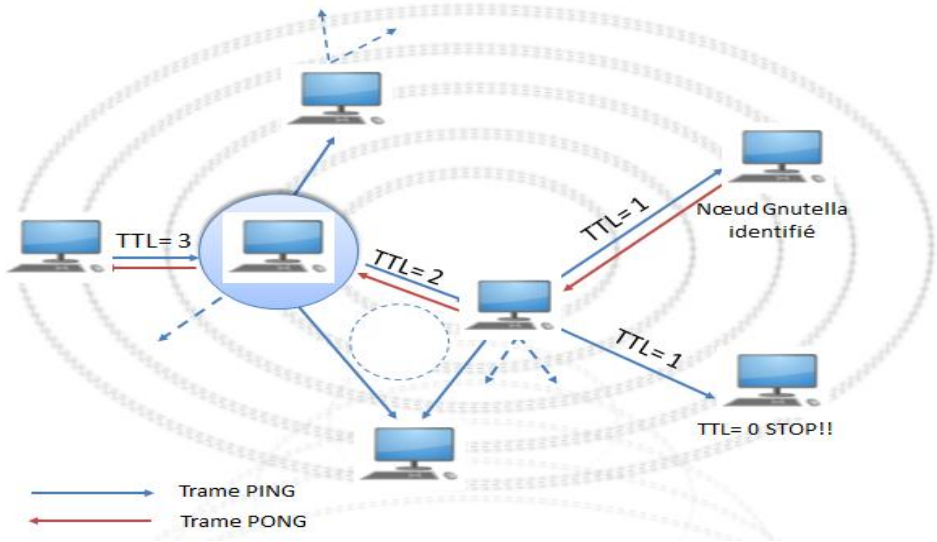


Figure 2-3- Principe de fonctionnement du système P2P Gnutella

Lorsque un nœud souhaite récupérer une ressources, il envoie une requête correspondante (message en rouge, figure 2-4) à ses voisins qui à leurs tours propagent cette requête à leurs voisins et ainsi de suite. La durée de vie d'une requête dans le réseau est fixée par une valeur TTL (Time To

Live) qui détermine le rayon de diffusion de la requête dans le réseau (cette valeur est décrémentée par chaque nœud intermédiaire durant le processus de diffusion). Apres avoir diffusé la requête de recherche, le nœud demandeur va recevoir une ou plusieurs réponses (message « QUERY HITS »), et peut par conséquent choisir la ressource qui correspond à sa recherche. Les adresses des nœuds possesseurs de la ressource sont indiquées dans les messages « QUERY HITS ». Le nœud demandeur se connecte directement au nœud possesseur de la ressource et initie la phase de transfert via un protocole classique d'internet (ftp, http, etc.).

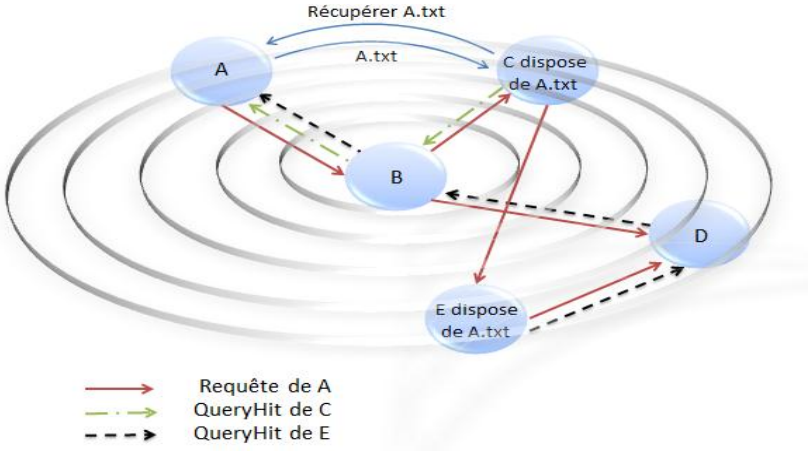


Figure 2-4- La recherche dans Gnutella

Les systèmes structurés : Dans un réseau structuré, chaque nœud possède un identifiant qui permet de le localiser en suivant un chemin déterministe parmi les pairs. Chaque ressource partagée possède aussi un identifiant (calculé à l'aide de fonctions de hachage) permettant ainsi de la localiser dans le réseau. Les P2P structurés sont basés généralement sur l'établissement d'une table appelée DHT (Distributed Hash Table) permettant de placer les nouveaux nœuds au sein du réseau. Chaque nœud reçoit une liste de voisins avec lesquels il pourra communiquer et chaque pair est responsable d'une partie spécifique du contenu du réseau. Un pair est responsable des entrées de la table égales ou proches de son identifiant. Plusieurs nœuds satisfaisant les contraintes de proximité peuvent stocker une ressource ; ce principe vise à assurer la pérennité de la ressource même dans un réseau qui contient des pairs non fiables. Les ressources partagées sont stockées donc à l'emplacement de leur identifiants. Les DHT offrent une localisation performante des pairs et des données. Le routage de proche en proche nécessite ainsi au maximum O (log(N)) messages (N représente le nombre de nœuds dans le réseau), cela signifie que les systèmes P2P structurés passent mieux à l'échelle que les systèmes non structurés (dont le nombre de message augmente linéairement O(N) avec la taille du réseau).

Dans de tels systèmes, un pair souhaitant récupérer une ressource doit d'abord posséder la clé de recherche correspondante. Cette clé permet d'identifier de façon unique la ressource à télécharger.

Par conséquent le pair demandeur n'aura une liste de propositions pour décider quelle est la ressource à télécharger mais récupèrera directement celle qui correspond à la clé de recherche. La distribution de cette dite clé est un problème principal dans les systèmes structurés. Certaines solutions proposent le déploiement de serveurs web pour le partage de ces clés de recherche.

Parmi les systèmes existants dans cette catégorie de réseaux P2P, nous pouvons citer CAN [RFHKS01], Chord [SMKKB01] et Freenet [CSWH01] [CHMSW02].

2.3.3. Les systèmes P2P hybrides

Le modèle hybride, appelé aussi modèle « super-peer » repose sur des interconnexions de super nœuds sur le niveau haut de la hiérarchie suivant le modèle distribué (voir figure 2-5). Chaque nœud feuille se rattache à un ou plusieurs super-nœuds. Un super-nœud gère un ensemble de nœuds feuilles. Par exemple, dans la nouvelle version de Gnutella V0.6[1], chaque super-nœud est connecté au plus à 10 autres super-nœuds et gère entre 10 et 100 nœuds feuilles.

Les informations relatives aux ressources partagées par un nœud feuille sont enregistrées sur le super nœud responsable de cette feuille. Lorsqu'un nœud feuille recherche un objet, il envoie sa requête à son super nœud. Celui-ci effectue alors la recherche parmi les objets des nœuds qui lui sont rattachés, voir les super-nœuds voisins si besoin. Le rôle des super-nœuds varie d'une application à une autre. Cependant, ils sont utilisés en général pour assurer les fonctions relatives à la localisation, au routage ou à l'organisation des pairs. La sélection des super-peers se fait selon plusieurs critères mais généralement suivant la nature de leur système d'exploitation, de la capacité de leur bande passante ainsi que de leur disponibilité dans le réseau.

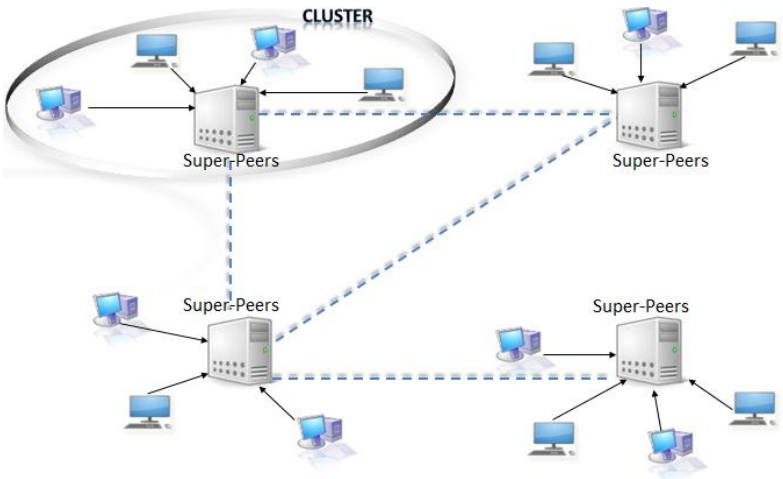


Figure 2-5- Principe de fonctionnement d'un système P2P hybride

[1] http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

2.4. La sécurité dans les réseaux P2P décentralisés

La popularité des réseaux P2P décentralisés est accompagnée de nombreux problèmes de sécurité. En effet l'absence de contrôle centralisé et l'autonomie des pairs a rendu les réseaux P2P victimes et supports d'activités malveillantes. Certains problèmes de sécurité sont spécifiques aux réseaux P2P. Ainsi dans le cas du modèle décentralisé, chaque nœud joue le rôle de « routeur » pour assurer les communications dans le réseau. Cela a engendré des protocoles de routage et de diffusion de messages qui admettent comme hypothèse de travail que tous les nœuds sont dignes de confiance, ce qui n'est pas toujours le cas dans un réseau publique. Dans ce qui suit, nous présentons une étude de la sécurité dans les réseaux P2P. Nous commencerons par donner quelques notions sur la cryptographie. Nous décrirons ensuite quelques attaques génériques qui affectent les réseaux informatiques en général et qui menacent un peu plus les réseaux P2P. Enfin nous présentons quelques attaques spécifiques aux réseaux P2P.

2.4.1. Notions sur la cryptographie

La Cryptologie est une science mathématique qui comporte deux branches : la cryptographie et la cryptanalyse. La cryptographie est l'étude des méthodes permettant de transmettre des données de manière confidentielle. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible : c'est l'opération de chiffrement, qui, à partir d'un texte clair, donne un texte chiffré ou cryptogramme. Inversement, le déchiffrement est l'action qui permet de reconstruire le texte en clair à partir du texte chiffré (voir figure 2-6). Dans la cryptographie moderne, les opérations de chiffrement et de déchiffrement sont assurées par des fonctions mathématiques qui dépendent d'un ou plusieurs paramètres appelés : clé de chiffrement et clé de déchiffrement. La cryptanalyse, à l'inverse, est l'étude des procédés cryptographiques dans le but de trouver des faiblesses aux fonctions de cryptographie en retrouvant le texte clair d'un message chiffré sans connaître la clé de déchiffrement.

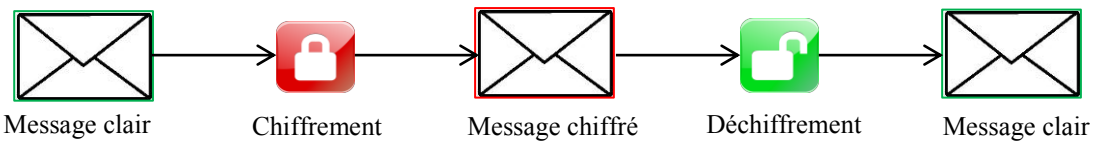


Figure 2-6- le principe de chiffrement et de déchiffrement

Le but initial de la cryptographie était d'élaborer des méthodes permettant de transmettre des données de manière confidentielle. La cryptographie moderne s'attaque plus généralement aux

problèmes de sécurité des communications dans les réseaux. Le but est d'offrir un certain nombre de services de sécurité comme la confidentialité, l'intégrité des messages échangés, l'authentification des communicants ainsi que la non répudiation. Il existe deux grandes familles d'algorithmes cryptographiques à bases de clés : les algorithmes à clé secrète et les algorithmes à clé publique.

Les algorithmes de chiffrement à clé secrète : La cryptographie symétrique [MVV96] [Sch96] dite également à clé secrète se fonde sur le principe de partage du même secret (la même clé, cf. figure 2-7) de chiffrement et de déchiffrement entre les communicants. Cette technique souffre du problème de la transmission de cette clé de façon sûre entre les communicants. Parmi les systèmes cryptographiques utilisés figurent les protocoles DES [SB88] et AES [DR01].

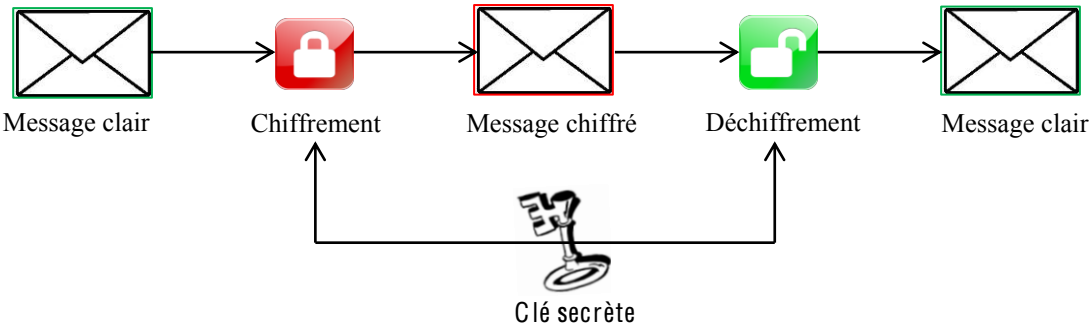


Figure 2-7- Le chiffrement à clé secrète

Les algorithmes de chiffrement à clé publique : Le concept de cryptographie à clé publique a été inventé par W. Diffie et M. Hellman [DH76] dans le but de résoudre le problème de distribution de clés posé par la cryptographie à clé secrète. De nombreux systèmes permettant de réaliser des chiffrements asymétriques ont été réalisés. Ils sont basés sur des problèmes mathématiques difficiles à résoudre. Le système RSA [RSA78] est un exemple d'un tel système et dont le fonctionnement utilise les propriétés des nombre premiers pour le calcul des clés publique et privé.

La cryptographie asymétrique utilise donc une paire de clés, une pour le chiffrement appelée clé publique (diffusée donc à tous les communicants) et l'autre, appelée clé privée (connue que de son propriétaire) qui sert au déchiffrement du message chiffré (cf. figure 2-8). Les deux clés utilisées sont distinctes et ne peuvent pas se déduire l'une de l'autre. De ce fait, tout le monde peut chiffrer un message, que seul le propriétaire de la clé privée pourra déchiffrer. De nombreux algorithmes ont été proposés pour réaliser un crypto système asymétrique tels : Elgamal [Elg85], RSA [RSA78] et Diffie-Hellman [DH76].

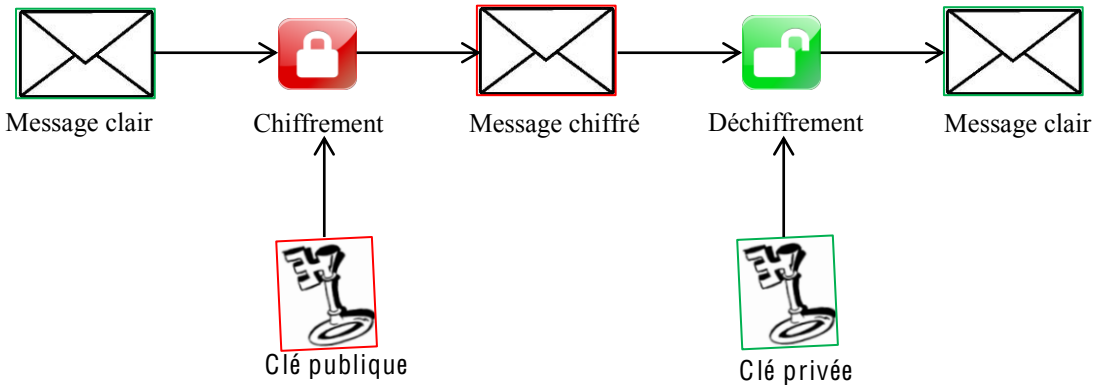


Figure 2-8- Le chiffrement à clé publique

Le problème du partage de secret **: L'algorithme de Diffie-**Hellman [DH76] : Inventé en 1976 par Diffie et Hellman, ce protocole permet à deux communicants de générer un secret partagé sans avoir aucune information préalable l'un sur l'autre. Ce protocole permet ainsi de résoudre en partie le problème de partage de clé dont souffrent les algorithmes de chiffrement symétrique. La sécurité de cet algorithme repose sur les propriétés mathématiques du logarithme discret sur un groupe fini. Il est à l'origine basé sur la cryptographie asymétrique d'où il fait appel à des valeurs publiques et des valeurs privées. Le protocole permet à deux interlocuteurs (Alice et Bob) de partager un secret sans qu'une troisième personne écoutant leur conversation ne puisse le découvrir. L'algorithme se déroule comme suit :

- Les deux communicants *Alice* et *Bob* se mettent d'accord sur deux entiers publiques n et g tel que : (n-1)/2 soit un nombre premier et g soit primitif par rapport à n.
- *Alice* et *Bob* choisissent au hasard deux grands entiers respectivement a et b qu'ils gardent comme secret.
- *Alice* calcule sa clé publique A de la manière suivante : $A= g^a \bmod n$. De même *Bob* calcule sa clé publique B tel que : $B= g^b \bmod n$.
- *Alice* et *Bob* échangent leurs valeurs publiques A et B (*Alice* envoie A à *bob* et *Bob* envoie B à *Alice*).
- *Alice* calcule $K_{AB}= B^a \bmod n$ et Bob calcule $K_{BA}= A^b \bmod n$.
- $K_{AB}= K_{BA} = B^a \bmod n = A^b \bmod n = g^{ab} \bmod n$ est le secret partagé par *Alice* et *Bob*.

Cependant, la méthode de Diffie-Hellman est vulnérable à l'attaque active dit de l'homme au milieu ou Man In The Middle [OV03].

Principe de l'attaque Man In The Middle: L'attaque de l'homme du milieu (HDM) ou *Man In The Middle attack* est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis. L'attaquant doit d'abord être capable d'observer et d'intercepter les messages d'une victime à l'autre.

L'attaque est particulièrement applicable dans le protocole original d'échange de clés Diffie-Hellman, quand il est utilisé sans authentification. L'attaque de l'homme du milieu suppose que l'attaquant ait la possibilité non seulement de lire, mais de modifier les messages. Voici le déroulement de l'attaque dans le cas de l'échange d'un secret entre deux communicants à l'aide de la méthode Diffie-Hellman :

- *Alice* et *Bob* souhaitent s'échanger un secret sur un canal de communication non sécurisé. *Oscar* peut écouter et modifier les communications entre *Alice* et *Bob.*
- *Alice* et *Bob* choisissent au hasard deux grands entiers respectivement a et b qu'ils gardent comme secret.
- *Alice* calcule sa clé publique A de la manière suivante : $A= g^a \bmod n$. De même *Bob* calcule sa clé publique B tel que : $B= g^b \bmod n$.
- *Alice* et *Bob* échangent leurs valeurs publiques A et B (*Alice* envoie A à *bob* et *Bob* envoie B à *Alice*)
- *Oscar* intercepte les messages de *Alice* et *Bob*. Oscar génère deux grands entier c et d qu'il garde en secret. *Oscar* calcule $O_1= g^c \bmod n$ et $O_2= g^d \bmod n$
- *Oscar* envoie la valeur O_1 à *Alice* en se faisant passer pour *Bob* et envoie la valeur de O_2 à *Bob* en se faisant passer pour *Alice*.
- *Alice* calcule $K_{A01}= O_1^a \bmod n$ et *Bob* calcule $K_{BO2}= O_2^b \bmod n$.
- $K_{AO1}= K_{O1A} = O_1^c \bmod n = g^{cb} \bmod n$ est le secret partagé par *Alice* et *Oscar* (*Alice* croit qu'elle le partage avec *Bob*)
- $K_{BO2}= K_{O2B} = O_2^d \bmod n = g^{bd} \bmod n$ est le secret partagé par *Bob* et *Oscar* (*Bob* croit qu'il le partage avec Alice).
- *Oscar* peut ainsi déchiffrer toutes les communications chiffrées entre *Alice* et *Bob*.

Les fonction de hachage: Parmi les problèmes auxquels s'attaque la cryptographie, on peut citer les problème de l'authentification des communicants et de l'intégrité des messages échangées : lorsque l'on communique sur un canal peu sûr, on souhaiterait que le destinataire d'un message s'assure que le message émane bien de l'auteur du message auquel il est attribué et que le message lui-même n'a pas été altéré pendant le transfert. La fonction de hachage à sens unique intervient dans la résolution de ce problème. Une fonction de hachage est une fonction qui convertit une chaîne de longueur quelconque en une chaine de taille généralement fixe appelée *empreinte* de la chaine. Une fonction de hachage est à sens unique dans le sens qu'il est facile d'engendrer une empreinte d'une chaine donnée mais qu'il est difficile de calculer une chaine à partir de son empreinte. La fonction de hachage possède aussi la propriété d'être sans collision, c'est à dire que la probabilité que deux chaines distinctes aient la même empreinte est très faible. Nous pouvons citer les fonctions MD5 [Riv92] et SHA [CR06] très utilisées dans les communications sécurisées des réseaux informatiques.

La signature numérique : La signature numérique dite aussi électronique est une norme définie dans L'ISO 7498-2[ISO89]. Par analogie à la signature manuscrite d'un document papier, la signature numérique est générée seulement par un expéditeur afin de prouver aux destinataires l'authentification de l'auteur des messages, l'intégrité et la non répudiation des messages échangés. Sur le plan conceptuel, la signature d'un message électronique est effectuée au moyen des deux fonctions:

- Une fonction de hachage H qui permet de convertir un message de longueur quelconque en chaine de longueur fixe pour générer l'*empreinte* d'un message.
- Une fonction de chiffrement asymétrique C ainsi qu'une clé privée (la clé est seulement connue par l'auteur du message qui va initier une signature) et une clé publique diffusée à tous les autres communicants.

La procédure d'envoi d'un message signé se déroule suivant le protocole suivant :

1. L'auteur du message calcule un message signé de M à l'aide de sa clé privée : $C_{priv}(H(M))$.
2. L'auteur du message M, construit un chiffré du message à envoyer à un destinataire dont il possède la clé publique : $C_{pub}(M)$.
3. Le destinataire déchiffre alors les deux chiffrés reçus à l'aide respectivement de la clé publique de l'expéditeur et de la clé privée du destinataire. Il calcule alors l'empreinte du message M (empreinte du déchiffré de $C_{pub}(M)$) et la compare avec l'empreinte envoyé par l'expéditeur (le déchiffré de $C_{priv}(H(M))$. S'il y a égalité alors ceci prouvera au destinataire l'authentification de l'expéditeur et l'intégrité du massage M.

2.4.2. Sécurité dans les réseaux et services P2P

Les réseaux P2P souffrent de beaucoup de problèmes de sécurité dus essentiellement à leur caractère distribué et l'absence d'autorité centrale pour la gestion des communications. Certaines attaques dont sont victimes les réseaux P2P sont celles qui affectent les réseaux informatiques en général et qui sont plus dangereuses dans le cas d'une architecture P2P. D'autres attaques sont plus spécifiques aux réseaux P2P. Nous présentons dans ce qui suit, les principales attaques qui perturbent le bon fonctionnement d'un réseau P2P.

L'attaque par Déni de Service Distribué (DDoS) : L'attaque DDoS [NR06] est générée par plusieurs machines formant une « coalition » et contrôlées généralement par une machine (celle de l'attaquant). Le but est saturer la bande passante d'une machine victime. Le principe consiste à envoyer une énorme quantité de paquets destinés vers la victime à partir de machines formant la « coalition ». Ces attaques proviennent donc de sources multiples et géographiquement distribuées

(des machines contrôlables à distance, cf. figure 2-9). Les réseaux P2P peuvent donc être facilement détournés pour lancer des attaques par Dénis de Services Distribués (DDoS).

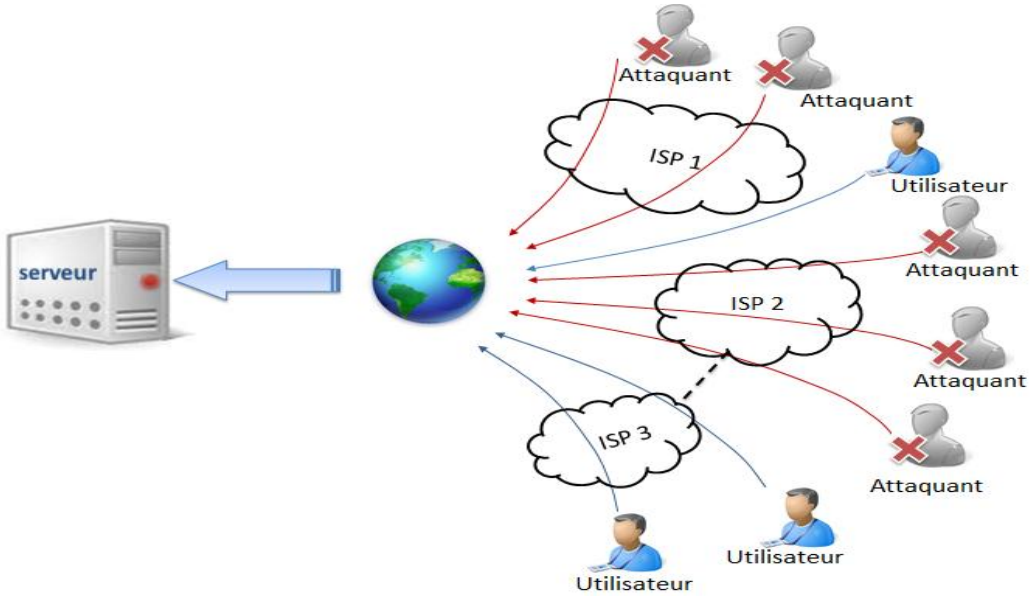


Figure 2-9- Attaque par déni de service distribué

Généralement, le but de l'attaque DDoS est de limiter ou empêcher le bon fonctionnement de certains serveurs. Les machines victimes peuvent être des équipements d'interconnexion tels qu'un routeur ou des machines offrant un service applicatif comme par exemple, un serveur web. Dans le cas des réseaux P2P, cette attaque peut cibler par exemple le serveur dans le cas du modèle P2P centralisé, les super-peers dans le cas du modèle hybride ou n'importe quel autre nœud dont on souhaite empêcher la diffusion de ses ressources. Les attaques DDoS sont très difficiles à contrer. Certaines solutions procédant au blocage de certaines adresses IP sources, limitent l'attaque mais ne l'arrêtent pas. D'autres solutions proposent de renforcer le niveau de sécurité des nœuds connectés au réseau afin d'éviter qu'ils soient compromis au lieu de protéger les machines cibles.

L'attaque Sybil [Dou02] : En présentant les réseaux P2P, nous avons établi que ce type de réseau peut réaliser des services fiables par un ensemble de pairs pas nécessairement fiables (pannes, non disponibilité permanente, comportement égoïste, etc.) et ceci est dû aux faits que les pairs sont indépendants les uns des autres, qu'ils sont aléatoirement répartis dans le réseau virtuel et par l'utilisation de mécanismes de réplication. L'attaque Sybil consiste pour une même entité à insérer un grand nombre de pairs dans le réseau. Les pairs ainsi crées ne sont plus indépendants et l'entité qui les a créées peur représenter une part significative du réseau. Ainsi, un fichier répliqué en théorie sur plusieurs pairs indépendants peut être indexé que sur des pairs contrôlés par l'attaquant Sybil.

Cette attaque est très difficile à contrer en l'absence d'une autorité de certification centrale pour gérer les identités des pairs (certifier les identifiants des pairs). Parmi les solutions proposées pour contrer l'attaque Sybil, celle qui a comme hypothèse le fait que tous les nœuds ont la même quantité de ressources qui peut être vérifiée pour chaque pair. De ce fait, une entité qui insère beaucoup de pairs dans le réseau ne pourra pas assumer la vérification pour chacun d'entre eux. Cependant, l'un des problèmes de cette méthode est l'hétérogénéité des pairs, la quantité de ressource exigée est généralement celle qui correspond au pair le moins puissant du réseau, une entité ayant des machines puissantes pourra donc maintenir de nombreux Sybils dans le réseau.

L'attaque Sybil est essentiellement utilisée pour réaliser plusieurs autres attaques telles que la pollution du réseau [CWC05], l'éclipse d'information dans le réseau [SNDW06] ainsi que la réalisation d'attaques de type DDoS.

L'attaque Eclipse : Cette attaque vise principalement les réseaux structurés. Le principe de l'attaque consiste à contrôler un certain nombre de pairs qui ont une influence sur le réseau afin d'empêcher le routage des messages vers les bons pairs, ce qui peut perturber le fonctionnement normal du trafic. Il est aussi possible de bloquer toute une partie du réseau via cette attaque si l'attaquant possède assez de ressources. Cette attaque peut être lancée aussi pour éclipser une donnée particulière. L'idée de base est de placer des pairs Sybils au plus près de l'identifiant d'une des clefs du contenu à éclipser, ce qui engendre des recherches sans réponses (si les Sybils ne répondent pas aux recherches, le contenu sera donc inaccessible).

Une solution pour contrer cette attaque est l'utilisation d'une autorité de certification (CA) qui attribue et vérifie les identités des pairs. Cependant, et avec une telle méthode, on passe d'un réseau dynamique et distribué à un réseau « centralisé ».

2.5. Conclusion

Pour conclure, nous avons présenté dans ce chapitre le concept des réseaux P2P, leurs caractéristiques, leurs domaines d'application ainsi que leurs classifications, à savoir les architectures centralisées, décentralisées et hybrides. Nous avons abordé aussi l'aspect sécurité (nous avons présenté une étude de la sécurité de ce type de réseaux) en présentant les attaques les plus spécifiques à cette famille de réseaux, notamment l'attaque Sybil et Eclipse. Ce genre de réseaux offre certes des avantages très intéressants grâce à son aspect distribué et son caractère dynamique, mais souffre d'une sécurité très fragile. Dans le prochain chapitre nous présentons l'un des problèmes majeurs rencontrés dans les réseaux P2P : ceux qui concernent l'anonymat des communicants et la confidentialité des échanges, qui, comme on va le montrer par la suite, représentent le souci actuel des internautes.

CHAPITRE 3

3. Protection de la vie privée et anonymat des échanges dans les réseaux P2P

Le modèle P2P ouvre de nouveaux horizons aux applications déployées. Il permet en effet de décentraliser la réalisation des services et de mettre à disposition des ressources partagées dans un réseau. Cette décentralisation présente de nombreux avantages qui repoussent les limites du modèle client-serveur tel que la tolérance aux pannes, le passage à l'échelle et la minimisation des coûts. Cependant, le caractère distribué des services et l'absence d'entité centrale pour gérer les échanges entre les nœuds ont vu apparaître divers problèmes de sécurité qui peuvent avoir un impact important sur la popularité des réseaux P2P. Parmi ces problèmes de sécurité figure celui de la préservation de la vie privée des différents communicants et particulièrement l'anonymat des fournisseurs et de demandeurs de ressources. La majorité des réseaux P2P populaires sur internet négligent cet aspect de la protection de la vie privée en favorisant les performances et la facilité d'utilisation du réseau: c'est le cas des systèmes BitTorrent [IPKA07] [PIAKV07] et Gnutella [Ber03]. Pourtant, assurer la protection de l'anonymat des communicants et de la confidentialité des échanges pourrait être un argument de taille pour inciter plus d'internautes à utiliser un réseau P2P et par conséquent augmenter ses performances.

Nous présentons dans la première partie de ce chapitre des notions générales sur la protection de la vie privée et des échanges anonymes dans un réseau de communication. Dans la deuxième partie, nous ferons un état de l'art des différents systèmes P2P ayant intégré dans leur architecture des fonctionnalités pour préserver l'anonymat des communicants durant les échanges ainsi que le maintien de la confidentialité des ressources échangées : il s'agit des systèmes Mute [Cho06] [Cho07], Freenet [CSWH01] [CHMSW02], OneSwarm [IPKA10], et Ants [CC05]. Enfin, Dans la troisième partie de ce chapitre, nous présentons notre contribution : Un système P2P de partage de données permettant de préserver l'anonymat des demandeurs et fournisseurs de ressources ainsi que le maintien de la confidentialité des ressources échangées. Une analyse de sécurité de notre protocole sera détaillée en faisant apparaître les apports de ce nouveau protocole par rapport aux solutions existantes.

3.1. **Notions générales sur l'**anonymat des échanges dans les réseaux

Parmi les différents domaines de la protection de la vie privée dans les réseaux informatiques, l'anonymat des échanges entre communicants est celui qui intéresse le plus la communauté de

chercheurs. Les protocoles de communication qui effectuent des échanges d'informations au travers de réseaux de communication utilisent des systèmes de chiffrement et de signatures pour assurer la confidentialité et l'authenticité des informations échangées. Même si les informations échangées sont sécurisées, un attaquant peut, au travers des entêtes de paquets échangés, avoir des informations sur les identités des communicants.

Pfitzman et al [PK00] définissent un certain nombre de notions liées au respect de la vie privée dans le contexte des réseaux de communication en général. Le modèle est formalisé de la façon suivante : des émetteurs envoient des messages à travers un réseau de communication à des receveurs. Les émetteurs et les receveurs sont des sujets du réseau de communication. Dans ce contexte, un attaquant cherche à répondre à la question suivante : qui (émetteur) communique avec qui (récepteur). L'anonymat n'est pas seulement lié à l'identité des communicants mais à toute information permettant d'identifier de façon unique le sujet. Il existe deux modèles d'attaquants classiques dans le contexte de l'anonymat des échanges dans un réseau de communication : L'attaquant local, qui peut 'écouter' les paquets entrants et sortants d'un ou plusieurs nœuds du réseau alors que l'attaquant global, peut intercepter les paquets circulant n'importe où dans le réseau. Ces attaquants peuvent être passifs et se contenter d'observer les paquets qui circulent dans le réseau (émis ou reçus) ou bien ils peuvent être actifs dans le cas où ils peuvent insérer, modifier ou supprimer des paquets.

Dans Pfitzman et al [PK00] sont définis les concepts suivants :

Ensemble d'anonymat : On appelle ensemble d'anonymat, un ensemble de nœuds qui ont les mêmes attributs et susceptibles d'effectuer les mêmes actions dans le réseau de communication.

Anonymat : Un nœud est anonyme s'il n'est pas identifiable parmi un ensemble d'anonymat. Il y a anonymat de l'émetteur (respectivement du destinataire) lorsque l'analyse d'un message ne permet pas de déterminer son émetteur (respectivement son destinataire) dans l'ensemble d'anonymat.

Non-observabilité : La non-observabilité est l'impossibilité de déduire l'existence d'un événement. Il y a non-observabilité de l'émetteur (respectivement du récepteur) lorsqu'il est impossible de déterminer s'il a émis (respectivement reçu) un message.

Inassociabilité : L'inassociabilité de deux ou plusieurs items d'intérêts (sujets, messages, actions, etc.) d'un point de vue de l'attaquant signifie qu'il ne peut pas distinguer si ces items sont liés entre eux. Par exemple, l'inassociabilité entre deux messages évite de pouvoir lier deux messages provenant d'un même émetteur et à destination du même receveur. La non-observabilité est un cas particulier de l'inassociabilité.

Anonymat d'un nœud intermédiaire : Dans le cadre d'un réseau de communication dynamique (exemples les réseaux ad hoc ou les réseaux P2P décentralisés), un message est relayé par des nœuds intermédiaires. Le respect de la vie privée concerne également ces nœuds. Un nœud intermédiaire est anonyme s'il n'est pas identifiable dans l'ensemble des nœuds intermédiaires concernés par une communication entre un nœud émetteur et un nœud récepteur.

3.2. Anonymat des échanges dans les réseaux P2P : Etat de l'art

Les systèmes P2P qui sont largement utilisés sur le net n'intègrent pas la propriété d'anonymat des échanges en préférant la facilité d'utilisation et des performances du système ; c'est le cas des systèmes BitTorrent [IPKA07] [PIAKV07] et Gnutella [Ber03]. Pourtant, nous pensons que la prise en compte par un réseau P2P de l'anonymat des communicants ainsi que la confidentialité des échanges pourrait être un argument de taille pour inciter plus d'internautes à utiliser ce réseau P2P et par conséquent augmenter ses performances.

Certains systèmes, tel que Freenet [CSWH01] [CHMSW02], prennent en charge l'aspect anonymat des nœuds durant les échanges mais au dépens des performances du système. Certains protocoles proposent des améliorations en termes de performances et prennent en charge l'anonymat et la confidentialité des échanges, mais comportent toujours des failles qui permettent aux attaquants de connaitre les vraies identités des demandeurs et/ou des fournisseurs de contenus, c'est le cas de Mute et Ants.

Nous présentons dans ce qui suit une étude d'un ensemble de systèmes P2P ayant intégré dans leur mode de fonctionnement une certaine dose d'anonymat des communicants durant les échanges ainsi que le maintien de la confidentialité des ressources échangées ; il s'agit des systèmes Mute [Cho06] [Cho07], Ants [CC05], Freenet [CSWH01] [CHMSW02], et OneSwarm [IPKA10]. La solution générique adoptée par ces systèmes consiste à utiliser des adresses virtuelles d'une part et faire transiter des flux chiffrés par des pairs intermédiaires d'autre part, afin d'éviter des connections directes entre les nœuds demandeurs et les nœuds fournisseurs et éviter ainsi l'obligation d'avoir à partager leur adresses IP.

3.2.1. Mute

3.2.1.1. Principe de fonctionnement

Mute est un protocole de partage de fichiers dans les réseaux P2P qui prend en charge l'aspect anonymat des échanges ; c'est l'un des protocoles les plus populaires dans le domaine. Mute tente

d'assurer l'anonymat des communicants en évitant les connexions directes entre les demandeurs et les fournisseurs de ressources. Dans Mute chaque nœud du réseau possède une adresse virtuelle en plus de son adresse réelle (IP) qui est générée lorsqu'un nœud se connecte au réseau. Les adresses IP sont utilisées pour les connexions entre les nœuds voisins (un nœud connait les adresses IP de ses voisins mais pas leurs adresses virtuelles) tandis que les adresses virtuelles sont utilisées pour le routage des échanges entre les demandeurs et les fournisseurs de ressources (le nœud connait les adresses virtuelles des demandeurs et des fournisseurs de ressources mais pas de leurs adresses IP). Autrement dit : un nœud du réseau qui connait les adresses IP de ses voisins ne connait pas celles des nœuds depuis lesquels il télécharge des ressources ou vers lesquels il les envoie. Pour assurer l'anonymat d'un nœud, Mute maintient la propriété que pour chaque nœud du réseau on ne peut pas avoir la possibilité de faire la correspondance entre l'adresse virtuelle du nœud et son adresse IP.

Dans Mute, chaque nœud garde un certain nombre de connexions avec d'autres nœuds appelés voisins ; le rôle de ces connexions étant de router les messages qui circulent au sein du réseau. Pour expliquer le principe de fonctionnement de Mute, prenons l'exemple de la figure 3-1 :

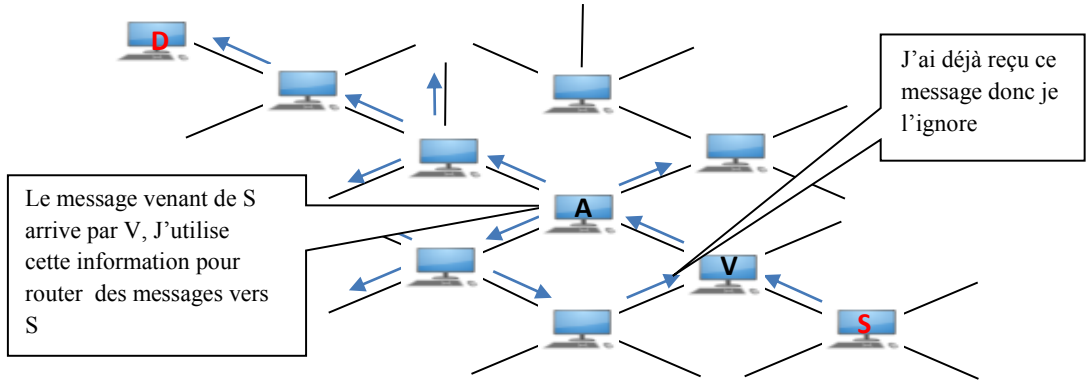


Figure 3-1- Message envoyé du nœud S au nœud D

Dans cet exemple, le nœud A reçoit un message dont le nœud source est S et la destination est le nœud D ; le message arrive au nœud A par le biais du nœud voisin V, cela permet au nœud A d'acquérir une connaissance qui est : « puisque le message venant de S arrive par V, les futurs messages à destination du nœud S vont être routés à travers le nœud V » ; notons que cette connaissance est locale au nœud A et rappelons aussi que le nœud A connait l'adresse IP du nœud V (mais pas son adresse virtuelle) et les adresses virtuelles des nœuds S et D (mais pas leur adresse IP). Si le nœud A n'a pas d'information sur la manière d'atteindre le nœud D, il procède à la diffusion du message à tous ses voisins directs ; si ses voisins n'ont pas l'information concernant la localisation du nœud D, ils diffuseront à leurs tours le message à tous leurs voisins (après avoir récolté la connaissance sur la manière d'atteindre le nœud S). Dans le cas où aucun nœud intermédiaire ne

possède cette information dans ses données locales de routage, le message en question finira par trouver le nœud destination (s'il existe dans le réseau bien évidement). Notons que chaque message possède un identifiant unique afin d'éviter le traitement multiple du même message par le même nœud intermédiaire. Un message déjà reçu par un nœud du réseau sera donc ignoré. La réponse du nœud D au nœud S va emprunter le chemin inverse du message précèdent (voir figure 3-2). De la même façon que le message envoyé par le nœud S, la réponse du nœud D va permettre aux nœuds intermédiaires entre S et D d'avoir des informations concernant la localisation du nœud D dans le futur.

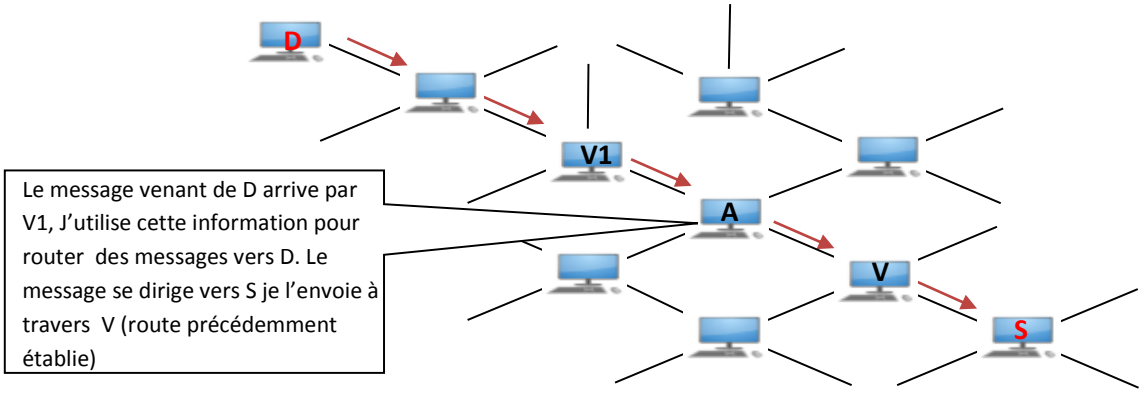


Figure 3-2- La réponse du nœud D au nœud S

Lorsque le nœud A reçoit la réponse du nœud D à destination de S, il l'envoie directement à travers le nœud V utilisant l'information récoltée lors de la phase précédente (figure 3-1) au lieu de procéder à la diffusion de la réponse aux voisins directs.

Afin de garder sa propriété d'anonymat, tout nœud Mute qui souhaite router un message vers une destination D, ne connaît, par conséquent, que l'adresse IP de son voisin qui, selon lui, a plus d'information concernant la localisation de la destination D que lui-même. Rappelons aussi que cette information est locale. Un nœud recevant un message de son voisin V, ne pourra pas déduire si V est bien la source du message ou bien s'il s'agit juste d'un relais. Dans l'exemple de la figure 3-2, le nœud A ne sait pas si V1 est la source de la réponse ou bien juste un nœud servant d'intermédiaire pour transporter la dite réponse. De la même façon, lorsque le nœud A relaye la réponse à travers le nœud V, il n'a pas la réponse à la question : « Es ce que le nœud V est le récepteur de la réponse ou bien c'est juste un relai qui figure dans le chemin menant vers S ? » et par conséquent il ne pourra pas déduire que son voisin V est le nœud S.

Dans Mute, les échanges entre les nœuds voisins sont chiffrés par une clef symétrique du protocole AES de 128 bits chiffrée elle-même par la clef publique du protocole RSA du nœud voisin, la clef symétrique est utilisée comme une clef de session. Cela permet d'éviter à un attaquant de surveiller le trafic entre les nœuds voisins.

Le principe du routage des messages dans Mute est basé sur un protocole de routage inspiré du fonctionnement d'une colonie de fourmis et dont le comportement est décrit par l'algorithme ACO « ant colony optimization » [DDC99].

Contrairement aux autres systèmes P2P décentralisés qui utilisent des valeurs TTL pour éviter à une requête de recherche d'être propagée vers tous les nœuds du réseau, Mute utilise une méthode probabiliste qui utilise une métrique appelée « Utility Counter » (UC). Cette métrique permet d'économiser la consommation de la bande passante et des ressources CPU. Cette méthode est utilisée aussi afin de protéger, d'une part, l'anonymat du demandeur de ressources en évitant à un attaquant d'analyser les TTLs et d'autre part l'anonymat du fournisseur de ressources en résistant contre les attaques basées sur le timing (cette attaque sera détaillée dans la suite de ce chapitre).

3.2.1.2. Analyse de performances et de sécurité du système

Du point de vu performances, le mécanisme de diffusion utilisé par Mute dans le but de propager les requêtes de recherche de ressources diminue considérablement les performances du système.

Concernant l'anonymat des communicants, Mute ne peut pas résister à certaines attaques [Cho07] pour la préservation d'anonymat ; en effet, si un attaquant arrive à contrôler tous les voisins d'un nœud Mute, il pourra observer et analyser des requêtes de recherche (resp des réponses) qui sont issues du nœud surveillé ce qui lui permet de déduire que le nœud surveillé est un nœud source de recherche (resp de données) « et non pas un relais » ; l'attaquant pourra dans ce cas faire la correspondance entre son adresse IP et son adresse virtuelle, ce qui peut avoir comme effet de briser l'anonymat de l'émetteur et/ou du récepteur (voir section 3.1).

La confidentialité des échanges n'est pas assurée dans Mute car il y a absence de chiffrement de bout en bout (les données échangées entre l'émetteur et le récepteur circulent en clair dans le réseau) ce qui permet à chaque nœud appartenant au chemin reliant un demandeur et un fournisseur de ressources de pouvoir consulter le contenu des fichiers échangées.

3.2.2. Ants

3.2.2.1. Principe de fonctionnement

Ants est un système P2P décentralisé qui vise à cacher les identités des communicants ; il présente des similitudes avec le protocole Mute en ce qui concerne l'utilisation des adresses virtuelles, des algorithmes de recherche et de transfert de données basés sur le fonctionnement de colonies de

fourmis ainsi que le chiffrement des échanges entre les nœuds voisins. L'apport de Ants par rapport à Mute est qu'il permet d'assurer le chiffrement des échanges de bout en bout (le chiffrement des données échangées entre les demandeurs et les fournisseurs de ressources). Les échanges dans le système Ants sont donc chiffrés par une méthode asymétrique, cela signifie que tous les nœuds peuvent consulter la recherche mais seul le nœud demandeur peut déchiffrer les résultats de la requête et les données transférées. Ce principe permet donc de répondre à l'objectif de la confidentialité des données échangées entre les deux extrémités (chose qui n'existe pas dans Mute).

3.2.2.2. Analyse de performances et de sécurité du système

Le modèle Ants est vulnérable face aux attaques de type Man In The Middle [OV03] (cf section 2.4.1). En effet des nœuds espions peuvent intercepter les requêtes pour les tracer avec leur propre clés de chiffrement. L'utilisateur espion peut de cette façon connaître les résultats de la requête ou les fichiers en transit puisque le « chemin » de la requête est le même que celui de la réponse. De ce fait, le protocole Ants ne peut pas préserver la confidentialité des ressources échangées.

3.2.3. Freenet

3.2.3.1. Principe de fonctionnement

Freenet est un système P2P de type décentralisé et résistant aux pannes ; l'objectif du système vise à assurer l'anonymat de ceux qui publient et aussi ceux qui consultent les données. Comme dans Ants et Mute, les données dans Freenet doivent transiter par les nœuds intermédiaires avant d'atteindre le nœud demandeur. Ainsi, pour protéger les hébergeurs, tous les fichiers (échangés ou stockés) sont chiffrés, de ce fait le propriétaire d'un nœud ne connait pas le contenu de tous les fichiers stockés sur ce nœud. Chaque nœud intermédiaire par lequel le fichier transite lors d'un transfert enregistre ce fichier dans son cache local avant de le faire passer au nœud suivant. En effet chaque fichier peut être stocké à plusieurs endroits différents du réseau; cette propriété permet ainsi d'assurer une certaine disponibilité de l'information et empêcher une quelconque censure puisque même lorsque le nœud propriétaire d'un fichier quitte le réseau, le fichier y est toujours présent.

Freenet est un système structuré où chaque fichier partagé possède donc un identifiant ; tous les nœuds du réseau Freenet possèdent une table de routage dont les entrées associent un identifiant avec un nœud. Lorsqu'un nœud reçoit un message de recherche d'un fichier qu'il ne possède pas, il transmet cette requête de recherche vers le nœud dont la clé (dans la table de routage) est la plus proche lexico-graphiquement de l'identifiant du fichier demandé. Après avoir effectué la recherche et trouvé le document recherché, ce document sera dupliqué sur tous les nœuds figurant dans le chemin

menant vers le nœud source de la recherche (via le chemin inverse de la requête de recherche), ce qui permet de rendre le système plus robuste.

Les fichiers dans Freenet sont chiffrés par un système de cryptage asymétrique. La clef privée du système sert pour signer les fichiers (figure 3-3) tandis que la clef publique, concaténée et hachée avec un texte descriptif du fichier permet de générer la clé de recherche (figure 3-4).

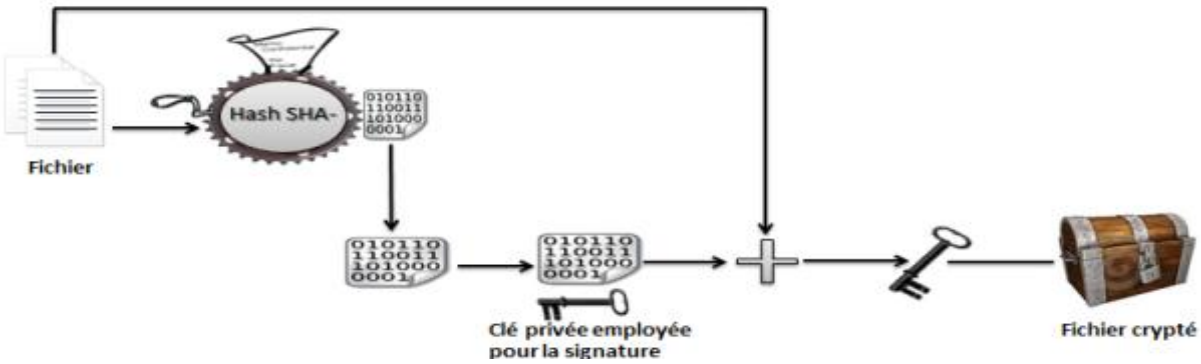


Figure 3-3- La signature d'un fichier dans Freenet

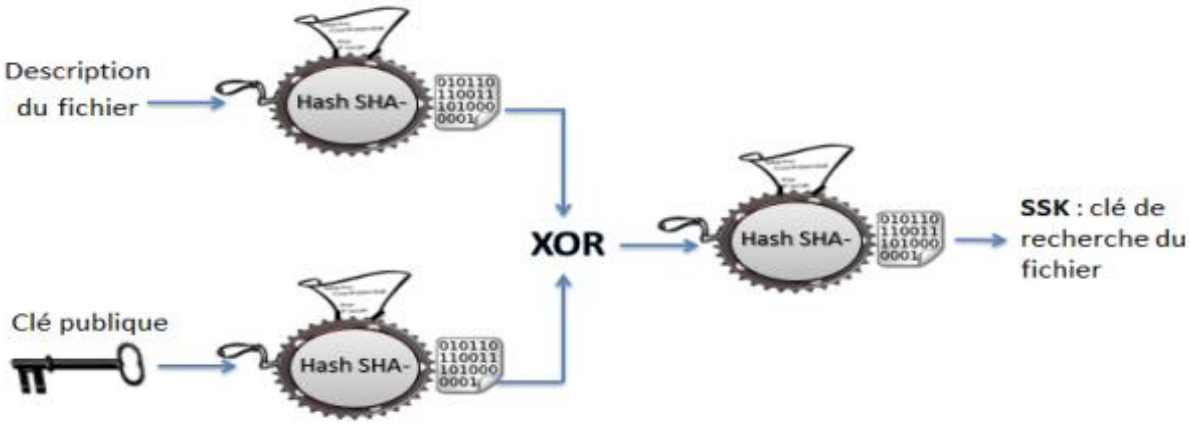


Figure 3-4-La génération de la clé de recherche dans Freenet

- Le Routage des messages dans Freenet

Un utilisateur désirant récupérer un fichier doit d'abord récupérer (via un site web par exemple) la clé de recherche qui correspond au document recherché. Le nœud demandeur va créer par la suite un message dit « DataRequest » (flèches vertes dans le schéma 3-5) qui contiendra la clé de recherche ainsi qu'une valeur TTL. Lorsqu'un nœud intermédiaire recevra ce message, il consultera d'abord ses données locales pour voir si le document recherché existe dans son espace de partage, si c'est le cas, le nœud en question va renvoyer à son voisin (celui par lequel il a reçu le message DataRequest) un message « DataReply » contenant le document recherché (flèches en bleu dans le

schéma), sinon il cherchera dans sa table de routage une similitude (déterminée par une distance lexicographique) avec la clé de recherche du fichier contenue dans la requête et il fera suivre le message DataRequest au nœud correspondant jusqu'à atteindre un nœud qui contient le document recherché.

Chaque nœud recevant un message DataReplay procédera comme suit :

1. Stocker le fichier envoyé avec la réponse dans son cache local.
2. Ajouter dans sa table de routage une nouvelle entrée associant la source de la donnée et la clé de la requête.
3. Relayer le message DataReplay vers le nœud suivant en direction du demandeur.

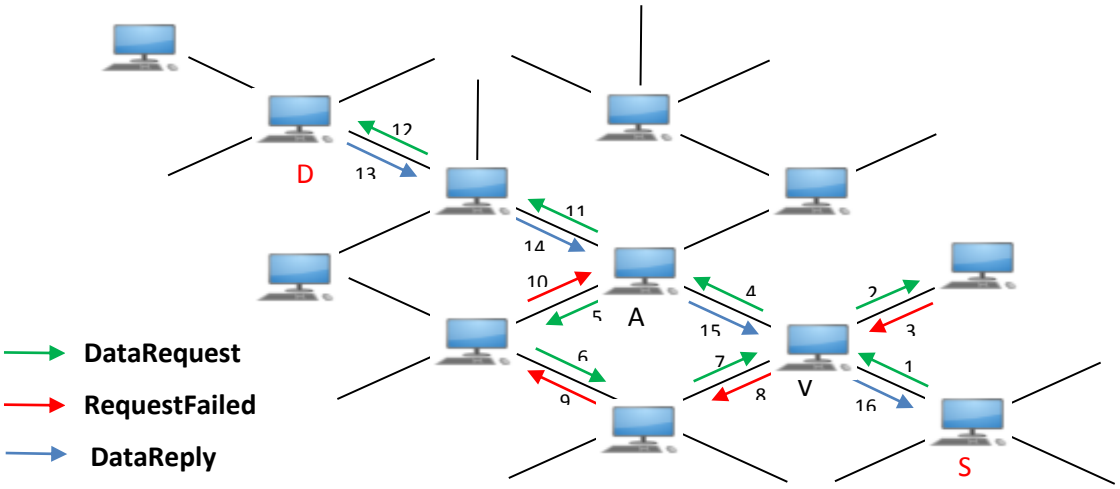


Figure 3-5- Le principe de fonctionnement de Freenet

Tout nœud qui ne peut pas satisfaire une requête de recherche peut répondre avec un message de type « RequestFailed » (flèches en rouge dans la figure 3-5) ; le message RequestFailed est envoyé au nœud voisin menant vers la source de la recherche. Ce message est envoyé dans l'un des cas suivants :

– Le nœud ne possède plus de voisins à qui faire suivre le message.
– Le TTL associé à la recherche a atteint la valeur 0.
– Le nœud a déjà reçu la requête.

3.2.3.2. Analyse de performances et de sécurité du système

Freenet souffre de problèmes de performance ainsi que de la difficulté d'utilisation du système. Cela est dû en partie à son système complexe de construction et de diffusion des clés de

recherche et au fait que les ressources transférées soient copiées sur tous les nœuds intermédiaires qui ont relayé une demande.

La diffusion des clés de recherche dans Freenet est une opération indispensable pour récupérer une ressource. Un client souhaitant partager un fichier doit d'abord trouver le moyen pour diffuser la clé de recherche avant d'insérer sa ressource dans le réseau. Une solution pour la diffusion de la clé serait l'utilisation d'un site web mais cette solution va à l'encontre du modèle défendu par Freenet qui est celui d'un système décentralisé.

Comme Freenet est un système P2P structuré, il est donc vulnérable aux attaques visant cette famille de réseaux P2P et qui ont été décrites précédemment.

Freenet reste un réseau puissant grâce à la possibilité d'échanger tous types de données dans un anonymat quasi total, mais reste peu performant au niveau des débits de transfert et souffre d'une certaines complexité dans la gestion et la distribution de clés de chiffrement.

3.2.4. OneSwarm

3.2.4.1. Principe de fonctionnement

OneSwarm est un système P2P de partage de données qui permet aux utilisateurs de contrôler et de paramétrer leurs données. Il offre le choix aux participants de partager des données de façon publique (Le partage des données n'exige aucune confidentialité) ou anonyme. Il permet aussi d'assurer le partage de données avec tous les amis ou juste une partie. L'un des objectifs de OneSwarm est de réduire l'impact de la confidentialité sur les performances. Pour cela l'opération de recherche et de transfert de données se fait via des pairs fiables et des pairs non fiables.

OneSwarm vise donc à permettre à l'utilisateur un processus de partage de données de manière efficace et sécurisée en lui assurant son anonymat lorsqu'il le souhaite. Le but est d'assurer un système de publication anonyme, un autre pour les téléchargements anonymes et enfin un anonymat pour le partage avec une certaine catégorie de personnes (des amis par exemple).

La localisation des fournisseurs de données se fait par une propagation de requêtes via le réseau logique. Comme dans Mute et Freenet, les transferts de données s'effectuent à travers les chemins inverses des recherches ; OneSwarm utilise la réécriture d'adresse afin de dissimuler les identités des sources et des destinataires. Notons que le transfert des données se fait via plusieurs chemins.

La nomination d'un fichier partagé dans OneSwarm est effectuée par le hash SHA-1 160 bits à partir de son nom et de son contenu. La récupération de ce hash dans le cas du partage publique se fait soit par courrier électronique, via un site internet, soit par des messages envoyés quand deux pairs se connectent appelés « listes de fichiers » ou par des recherches sur des mots-clés dans le réseau logique.

Chaque requête de recherche par mots-clés est dotée d'un identificateur unique. Contrairement à quelques systèmes P2P étudiés, cette requête ne contient pas de valeur TTL car les concepteurs de OneSwarm pensent que si un attaquant faisant partie d'un chemin donné analyse cette valeur, il peut casser l'anonymat des demandeurs et/ou des fournisseurs de ressources. Les pairs OneSwarm gardent un historique des requêtes de recherche pour éviter de propager des doublons. La propagation des messages de recherche parmi les pairs non fiables se fait juste sur quelques-uns et de façon aléatoire dans le but d'éviter les attaques par groupe. Cette méthode évite ainsi aux pairs non fiables agissant de concert, de déduire la source de données en remarquant l'absence de la propagation d'une requête de recherche. Pour éviter à une requête de recherche d'être propagée vers tous les nœuds du réseau, chaque client relai retarde ces requêtes d'au moins 150ms avant de les transmettre aux autres pairs. Pour mettre fin à une recherche, le client source de la recherche ou un nœud intermédiaire envoie un message dit d'annulation via les mêmes chemins de la recherche en question sans délai de propagation (pour atteindre rapidement les bornes de la recherche).

Le nœud possesseur d'un fichier recherché ne relayera pas directement la recherche et répondra en envoyant un message de réponse à la recherche au nœud demandeur. Cependant, un certain nombre de critères doivent être pris en considération avant d'envoyer cette réponse: entre les pairs de confiance, la réponse est envoyée immédiatement, mais lorsqu'il s'agit d'envoyer la réponse à un pair non fiable, le nœud répondeur doit retarder cette réponse (d'ailleurs c'est le cas de tous les messages du protocole destinés aux pairs non fiables). L'objectif par le retardement d'un message de réponse vers un pair non fiable, est d'éviter à ce pair de déduire que le nœud répondeur possède les données cherchées en observant une réponse reçue en moins de 150ms (qui correspond au délai de propagation par défaut des messages de recherche). Le but est donc de simuler que la recherche en question a été relayée vers les autres nœuds du réseau logique avant qu'il ait une réponse et par conséquence, le pair non fiable ne pourra rien déduire. Le nœud à l'origine de la réponse choisira aléatoirement une valeur entre 150 et 300ms comme temps de retard (ce qui correspond à 1 ou 2 saut).

Apres la découverte et l'utilisation des chemins de transfert de données (qui sont les chemins inverses des recherches), OneSwarm utilise des messages appelés « messages de maintien en vie » dont le rôle est de rafraîchir les chemins qui expirent après 30 secondes d'inactivité. Durant les téléchargements, OneSwarm effectue des recherches afin de découvrir de nouveaux chemins.

3.2.4.2. Analyse de performances et de sécurité du système

OneSwarm implémente quelques techniques originales pour la mise en place des processus de recherche et de transfert de ressources au sein du réseau, nous pensons cependant que certaines peuvent influencer négativement sur les performances et/ou l'anonymat des communicants.

Nous avons vu que OneSwarm utilise le principe d'augmentation artificielle des délais s'appliquant aux requêtes issues de pairs non fiables (toutes les réponses faites à des pairs non fiables subissent un délai aléatoire mais déterministe pour simuler le délai qui correspond à une requête propagée via un ou plusieurs intermédiaires de plus). Nous pensons que ce retardement de messages peut causer une diminution considérable des performances du système.

Ainsi, nous avons vu que OneSwarm [IPKA10] utilise des messages de maintien en vie pour rafraîchir les chemins qui expirent après trente secondes d'inactivité ; nous pensons que cette méthode peut permettre à un attaquant d'analyser le trafic en se basant sur cette caractéristique et déduire des sources de données (risque de casser l'anonymat du fournisseur de ressources). L'utilisation des messages « d'annulation de recherche » peut permettre aussi à des attaquants de localiser le demandeur par une étude fine du trafic et par conséquent casser l'anonymat du demandeur.

Tout comme les systèmes Mute, Ants et Freenet, dans OneSwarm les chemins de recherche et de transfert de données sont identiques. OneSwarm est donc vulnérable à l'attaque Man-In-The-Middle et par conséquent ne peux pas préserver la propriété de confidentialité des ressources échangées.

3.3. Notre proposition : le protocole AnonymP2P

Dans ce chapitre nous présentons la conception, le développement et l'expérimentation d'un protocole de partage de fichiers (ou de ressources d'une manière générale) qui vise à assurer l'anonymat et la confidentialité des échanges entre les différents nœuds : le système AnonymP2P. C'est un système P2P distribué de type non structuré. Le système vise à réduire le coût dû à l'anonymat et la confidentialité des échanges en se concentrant sur la robustesse et les performances. Afin de maintenir la propriété d'anonymat et de confidentialité, le protocole fonctionne sur le principe d'un réseau social avec des nœuds identifiés par des adresses virtuelles ; il permet à chaque nœud de modifier aléatoirement dans le temps ses voisins pour éviter à un attaquant de l'entourer et par conséquent déduire la correspondance adresse virtuelle-adresse IP. Le protocole utilise aussi un processus de recherche et de transfert de données qui diffère de ceux qui existent dans les autres réseaux P2P. En effet, le chemin suivi par la requête de demande de ressource est différent des

chemins de la réponse afin de réduire la probabilité d'une attaque de type Man In The Middle [OV03]. AnonymP2P utilise un ensemble de messages de contrôle fonctionnant à la façon d'un système Multi Agents mobiles car nous pensons que la notion d'agents collaboratifs s'adapte bien aux concepts de réseaux P2P décentralisés et de réseaux sociaux. Les requêtes de localisation ainsi que les transferts de données multi-chemins maintiennent la propriété d'anonymat du fournisseur, du demandeur et prennent en compte les problématiques de congestion, fournissant de bonnes performances au prix d'un surcoût raisonnable.

3.3.1. Principe de fonctionnement du protocole

AnonymP2P permet à tout nœud du réseau de rechercher puis de récupérer des ressources tout en assurant l'anonymat du demandeur et du fournisseur de ces ressources ; AnonymP2P permet aussi un anonymat entre le demandeur et le fournisseur puisqu'une fois le transfert achevé, ils ne déduisent rien de leur identité réelle respective. Afin de maintenir l'anonymat lors des échanges, plusieurs éléments sont mis en place: Tout d'abord le réseau P2P fonctionne sur le principe d'un réseau social virtuel, en effet chaque nœud possède une identité virtuelle qu'il utilise lors d'une demande de ressource (voir figure 3-6). Chaque nœud se fait connaître du réseau à travers une connexion à des nœuds voisins qu'il modifie aléatoirement dans le temps.

Lorsqu'un nœud désire récupérer une ressource, il lance une demande à travers ses voisins. Cette demande est acheminée par un ensemble de messages de contrôle fonctionnant à la manière d'un système Multi-Agent mobiles. Il n'y a pas de diffusion de requête comme dans les autres systèmes P2P, ce sont au contraire les agents mobiles collaboratifs qui s'occupent de la propagation de cette requête. Chaque agent mobile est généré et propagé régulièrement par chaque nœud du réseau virtuel. Lors de sa mobilité dans le réseau, l'agent mobile transporte une ou plusieurs demandes de ressources déposées par certains nœuds intermédiaires et établit des routes anonymes entre les nœuds du réseau lors de son parcours dans le réseau virtuel. Plusieurs routes vont être construites (par les agents collaboratifs) entre chaque paire de nœuds du réseau ; ceci a pour conséquence des téléchargements rapides et multi sources. Une gestion des pertes de paquets et des ruptures de liens entre les nœuds est aussi disponible grâce à des informations de routage diffusées par les agents.

Chaque nœud du réseau stocke ses propres requêtes mais aussi les requêtes des autres nœuds et cela pour des raisons de performances et aussi pour préserver l'anonymat du demandeur comme cela sera expliqué un peu plus loin.

Dès que le nœud demandeur envoie sa demande de ressources via les agents mobiles, il décide, avec une certaine probabilité, de changer certains de ses voisins puis initie le calcul des

chemins pour le transfert de la ressource ; de ce fait, le chemin de la demande de la ressource sera différent des chemins du transfert de cette même ressource. Ce processus original de calculs de routes lors de la demande et de transfert permet au demandeur et au fournisseur d'échanger des ressources chiffrées à l'aide d'une clé secrète établie par un protocole de type Diffie-Hellman [Res11] [DH76] et qui permet ainsi de réduire les risques d'attaques de type Man In The Middle [OV03] (cela sera expliqué plus loin dans le manuscrit). Enfin, pour renforcer la propriété d'anonymat des échanges, le protocole ne fera circuler dans le réseau que des paquets chiffrés qui ont sensiblement la même taille, de cette façon il est difficile aux attaquants qui surveillent le trafic de faire la distinction entre les paquets de contrôles et les paquets de données, et permet par conséquent de maintenir une propriété de non observabilité.

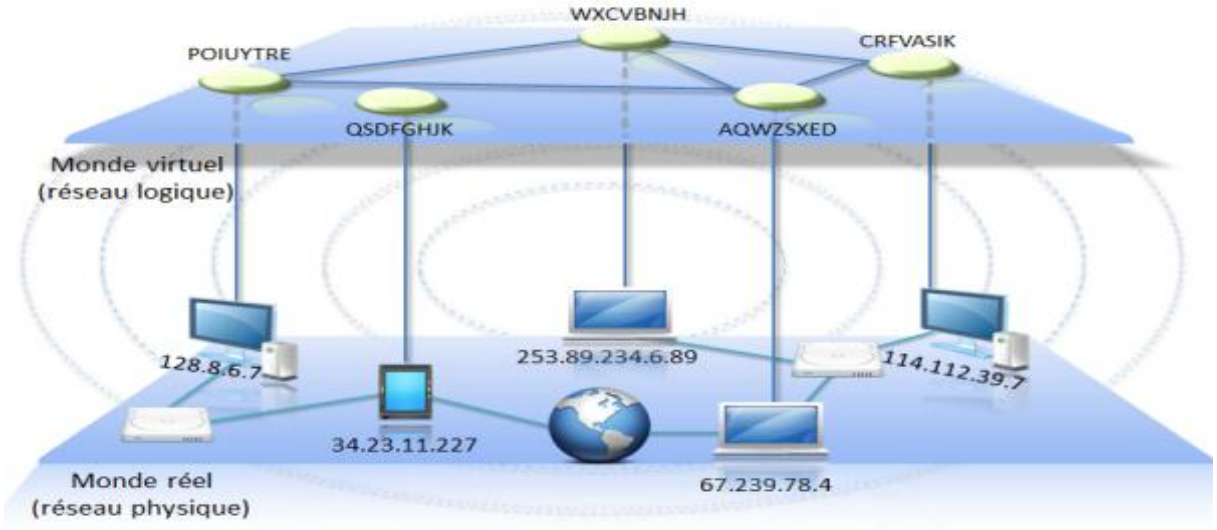


Figure 3-6- Le principe d'un réseau social utilisé dans AnonymP2P

3.3.2. Fonctionnement détaillé du protocole

3.3.2.1. Notations et terminologie

Les notations utilisées pour décrire le protocole AnonymP2P sont définies comme suit :

ID_A :	L'identité virtuelle du nœud A.
PK_{AB} :	La clé publique du nœud B reçu par le nœud voisin A.
K_A :	Le secret généré par le nœud virtuel A.
K_{AB} :	Le secret partagé par les nœuds dont les identités virtuelles sont ID_A et ID_B.
$E_{PKAB}(M)$:	Le message M est chiffré par la clé publique de B et envoyé par le nœud A au nœud voisin B.
$E_{KAB}(M)$:	Le message M est chiffré par la clé secrète K_{AB} échangée entre les nœuds ID_A et ID_B.
$DH_1(.)$:	L'échange du premier secret par la méthode Diffie-Hellman.
$DH_2(.)$:	L'échange du second secret par la méthode de Diffie-Hellman.
$Sign_s(M)$:	Le message M est chiffré et signé par la clef secrète s.

3.3.2.2. Gestion de la connectivité

Chaque nœud du réseau va disposer d'un nombre limité de voisins. Initialement, lorsqu'un nœud veut s'insérer dans le réseau, il récupère de potentiels nœuds voisins en interrogeant un ensemble de nœuds connus sur le réseau. Afin de sélectionner comme voisins les nœuds les plus proches, le protocole mesure la proximité de chaque nœud nouvellement connu en calculant le temps d'aller-retour (TAR) d'un paquet via une communication directe à travers le protocole TCP. Pour des raisons de performances, chaque nœud du réseau peut faire périodiquement des calculs de proximité avec d'autres nœuds (surtout les nœuds nouvellement insérés dans le réseau) et qui souhaitent être ses voisins, afin de les choisir comme nouveaux voisins et supprimer les anciens. Cette méthode permet à un nœud d'avoir un voisinage dynamique et ceci offre une analogie entre une proximité en termes de réseau informatique réel et une proximité du réseau virtuel P2P. Afin de préserver la propriété d'anonymat, un nœud peut être contraint à choisir d'autres voisins qui ne sont pas forcément plus proches et supprimer les anciens. Ce principe de voisinage dynamique permet de simuler la mobilité d'un nœud dans le réseau et de « perturber » de ce fait un attaquant qui souhaiterait entourer un nœud donné (comme c'est le cas dans les systèmes Ants et Mute décrits précédemment). Nous supposons que lors du processus de connexion d'un nœud avec ses voisins, il s'établit un échange de clés publiques permettant aux nœuds voisins d'avoir des échanges chiffrés (messages de contrôles et données). AnonymP2P assure donc un flux chiffré entre un nœud et son voisin et cette communication chiffrée entre deux nœuds voisins est chiffrée utilisant une clé symétrique AES chiffrée elle-même avec la clef publique (RSA) du nœud voisin échangée lors de la phase de connexion. La clef secrète AES utilisée est une clé de session qui est modifiée à chaque fois qu'un nouveau flux est envoyé.

3.3.2.3. Recherche et transfert de données

Nous avons décrit comment les pairs de AnonymP2P accèdent au réseau logique, y entretiennent des connexions et mettent à jour les informations de connectivité. Dans ce qui suit, nous allons présenter les protocoles utilisés pour localiser (rechercher) et transférer des données entre les pairs.

Afin de permettre à certains nœuds du réseau de réaliser des demandes de ressources puis de les récupérer à travers d'autres nœuds, le protocole utilise un ensemble de messages de contrôles fonctionnant à la façon d'un système Multi Agents mobiles. Chaque message de contrôle est créé par un nœud et envoyé dans le réseau de façon totalement aléatoire avec un TTL choisi au hasard. Chaque message de contrôle est porteur d'un ensemble d'informations qui vont permettre aux différents nœuds de collaborer pour calculer les chemins de recherche de nœuds possesseurs de ressources et les chemins de transfert de ces mêmes ressources. Autrement dit, chaque nœud met à disposition des

autres nœuds du réseau, un ensemble d'agents mobiles permettant de localiser les ressources demandées par les différents nœuds ainsi que l'établissement des routes anonymes entre ces différents nœuds.

Chaque message de contrôle transporte deux types d'informations : une liste de requêtes (demandes de ressources), et une liste contenant des informations de routage pour le calcul des chemins 'anonymes' de transfert des ressources entre les nœuds demandeurs et fournisseurs. Les informations contenues dans ces listes sont échangées et mises à jours avec les différents nœuds intermédiaires lors de la mobilité des agents dans le réseau. Dans ce qui suit, nous présentons le processus de recherche et de transfert de ressources entres les différents nœuds.

- Le processus de recherche

Lorsqu'un nœud souhaite récupérer une ressource dans le réseau, il ne diffuse pas la requête (comme c'est le cas dans les réseaux P2P traditionnels), mais il choisit aléatoirement un agent mobile de passage par ce nœud et y insère les informations suivantes:

$E_{PKAV}((<ID_A><list\_mots\_cles><DH_1(K_A)><TTL><ID_{A\_session}>),...)$, où $<ID_A>$ représente l'adresse virtuelle choisie par le nœud demandeur et qui sera diffusée dans le réseau lors cette demande, *<list_mots_cles>* représente une liste de mots clés de la ressource recherchée, $<DH_1(K_A)>$ est la première partie du secret que vont échanger le demandeur et le fournisseur par une méthode type Diffie-Hellman [Res11] (l'autre partie du secret sera envoyée par le fournisseur lors de la réponse à la requête, et ceci sera expliqué plus loin dans ce mémoire), *<TTL>* représente le nombre de fois que la requête sera réexpédiée (forwardée) par le nœud en cours (cette valeur sera décrémentée à chaque fois que le nœud réexpédie la requête) et enfin $<ID_{A\_session}>$ est un entier généré aléatoirement par le nœud demandeur afin d'identifier la demande et d'éviter les boucles dans le réseau. Ces informations sont insérées dans un agent mobile chiffré avec la clé publique PK_{AV} où V est un voisin choisi aléatoirement auquel sera envoyé l'agent mobile. Les agents mobiles diffusent ainsi ces informations le long des différents nœuds qu'ils parcourent (cf. figure 3-7).

Afin de renforcer la fonctionnalité d'anonymat des échanges, un nœud quelconque ne stocke pas seulement ses propre requêtes mais aussi celles des autres nœuds du réseau ; de ce fait même si un attaquant arrive à observer des requêtes qui 'partent' d'un nœud sans être 'reçues', il ne pourra pas déduire s'il s'agit des requêtes du nœud surveillé ou d'autres nœuds. La durée de stockage d'une requête au niveau d'un nœud est gérée de la manière suivante : à la réception d'une requête, cette dernière est dotée d'une valeur entière TTL, cette valeur ne traduit pas le nombre de fois que la requête sera réexpédiée (forwardée) d'une manière générale (ce qui est le cas de la majorité des

protocoles existants), mais du nombre de fois que cette requête sera réexpédiée (forwardée) par un nœud quelconque et cela à travers les agents mobiles qui transitent par ce même nœud. Par exemple, si un nœud stocke une requête avec un TTL=3 cela veut dire qu'il peut la diffuser trois fois vers ses différents voisins et par trois agents mobiles différents, avant de la supprimer.

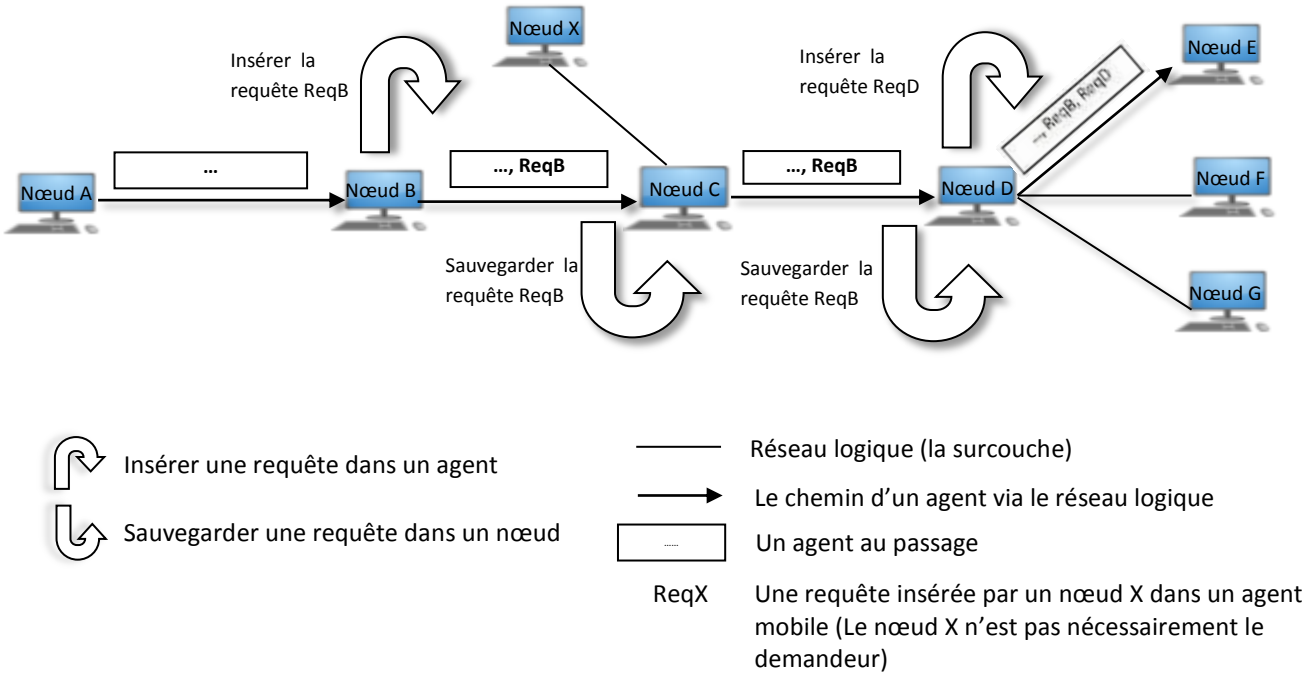


Figure 3-7- Le processus de recherche dans AnonymP2P

- Établissement des routes anonymes

Un des objectifs du protocole AnonymP2P est de protéger le fournisseur d'une ressource en préservant son anonymat. Dans la majorité des protocoles existants, la ressource est envoyée le long du chemin 'inverse' de celui de la requête de recherche. Dans AnonymP2P, le nœud possesseur d'une ressource va transférer sa ressource le long de 'nouveaux' chemins initiés par le demandeur en procédant comme suit :

Lorsqu'un nœud demandeur a diffusé sa demande de ressource, il décide, avec une certaine probabilité, de changer de voisinage dans le réseau (avec au moins un voisin) puis initie le calcul de plusieurs chemins pour le transfert de la ressource. Le nœud demandeur va initier cette route en déposant dans sa table de routes anonymes l'entrée suivante :

$<ID_A><ID_{A\_session}><distance><nœud\_voisin>$ ou $<ID_A>$ correspond à l'adresse virtuelle du nœud demandeur, $<ID_{A\_session}>$ permet d'identifier la demande, *<distance>* est un entier généré par

le demandeur correspondant à la distance (en nombre de sauts) qui sépare le nœud demandeur d'un nœud 'fictif' généré par le demandeur et enfin *<nœud_voisin>* correspond à un nœud voisin du demandeur choisi aléatoirement par ce dernier. Les agents mobiles se chargent alors de diffuser cette information afin de permettre aux différents nœuds de mettre à jour leur table de routes 'anonyme'. Les entrées des tables de routes 'anonymes' des différents nœuds sont mises à jour de la façon suivante : si l'entrée *<ID><ID_session><distance>* contenue dans un agent venant d'un nœud x correspond à une nouvelle adresse virtuelle dans la table du nœud courant alors le triplet *<ID><ID_session><distance+1><x>* est inséré dans la table de calcul de routes 'anonyme' du nœud courant. C'est ainsi que de proche en proche des chemins 'anonymes' seront établis entre le fournisseur et le demandeur de la ressource et cela dès que la table de calcul de chemins 'anonyme' du fournisseur contiendra des entrées avec une destination vers l'adresse virtuelle du nœud demandeur. Notons que si le prochain nœud que l'agent de cet exemple va visiter est le nœud y, alors ce dernier sera inséré dans le champ NList de la table de routage (cf. tableaux 3-1), cela est dans le but de garder l'information que le nœud y va probablement utiliser cette route anonyme (ce processus est utile dans la gestion des pertes de liens avec les voisins qui sera détaillée dans le reste de cette section).

Tableau 3-1- La table de routes anonymes de notre protocole

Destination (adresse virtuelle)	<ID_session>	Distance	**Next Hop (Nœud** suivant) (adresse IP)	NList
6ca1534291dab0c17cd1fcb35be7a510	3635.200	8	15.22.45.12	15.25.33.10
ace4e7065eb9cb23981182eb38976204	540.265	9	43.23.145.12	89.55.64.1
2e7ffc8e54390678ac3d4387098ffff4	2348.104	6	104.11.22.34	125.36.48.2
…	…	…	…	…

- La réponse et le transfert de la ressource

Le nœud fournisseur envoie une réponse via les chemins 'anonymes' calculés lors de la phase précédente. Les messages de réponse à une recherche incluent un identificateur de recherche : *<ID_session>*, une liste qui identifie les ressources correspondantes à la recherche et les métadonnées des ressources (chaque ressource proposée est dotée d'un identifiant unique « Idr »). Le nœud fournisseur envoie aussi avec la réponse, la clé *<$DH_2(K_B)$>* (cf. processus de recherche) qui correspond à la deuxième partie de la clé que s'échangent le fournisseur et le demandeur afin d'établir un secret, grâce à un protocole de type Diffie-Hellman. Afin de réduire le risque d'une attaque de type

Man In The Middle, les réponses à la requête ainsi que la clé $DH_2(K_B)$ seront envoyées le long des différents chemins 'anonymes' créés précédemment. Le nœud demandeur, pourra ainsi éviter l'attaque Man In The Middle en choisissant la 'bonne' clé $DH_2(K_B)$. Cette clé doit avoir la même valeur que celles reçues à travers tous les chemins anonymes de la réponse (cf. figure 3-8) sinon les réponses seront rejetées (cas d'un déni de service).

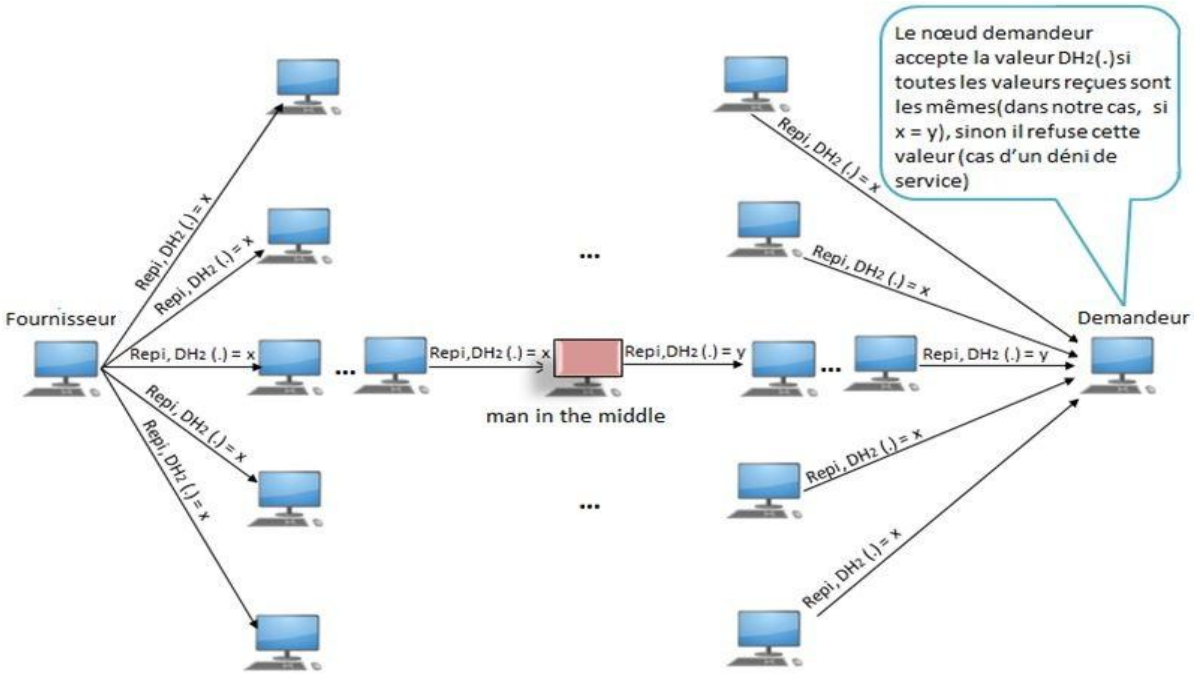


Figure 3-8- La réponse et le transfert de la ressource

Le nœud demandeur détermine la ressource à télécharger à travers les chemins inverses des routes 'anonymes' en diffusant comme message le paquet suivant : $E_{PKdv}(<ID_d>, <ID_f>, \underline{E_{Kdf}(<Idr>, <ID\_session>)})$, où $<ID_d>$ est l'adresse virtuelle du demandeur de la ressource, $<ID_f>$ est l'adresse virtuelle du nœud fournisseur de la ressource, $<Idr>$ représente l'identifiant de la ressource recherchée (identifiant envoyé avec la réponse du fournisseur) et $<ID\_session>$ est le numéro permettant d'identifier la requête. Les valeurs $<Idr>$ et $<ID\_session>$ sont chiffrées à l'aide du secret K_{df} partagé entre le demandeur et le fournisseur lors des phases précédentes. Le message ainsi construit est chiffré par la clé publique PK_{dv} où V représente le prochain nœud voisin qui se trouve sur le chemin 'anonyme' construit à l'étape précédente. Lorsqu'un nœud fournisseur B veut envoyer la ressource demandée et afin de respecter la contrainte de taille des messages qui circulent dans le réseau, il découpe la ressource f en un ensemble de paquets f_1, f_2,...f_n de taille sensiblement égales et les envois le long des différents chemins 'anonymes'. Chaque message aura la structure suivante :

$E_{PKAV}(<ID_f>, <ID_d>, <\underline{E_{Kdf}(f_i, i, ID\_session)}, sign_{Kdf}(f_i, ID\_session)>)$; où $<ID_f>$ représente l'adresse virtuelle du fournisseur, $<ID_d>$ est l'adresse virtuelle du demandeur lors de la phase demande,

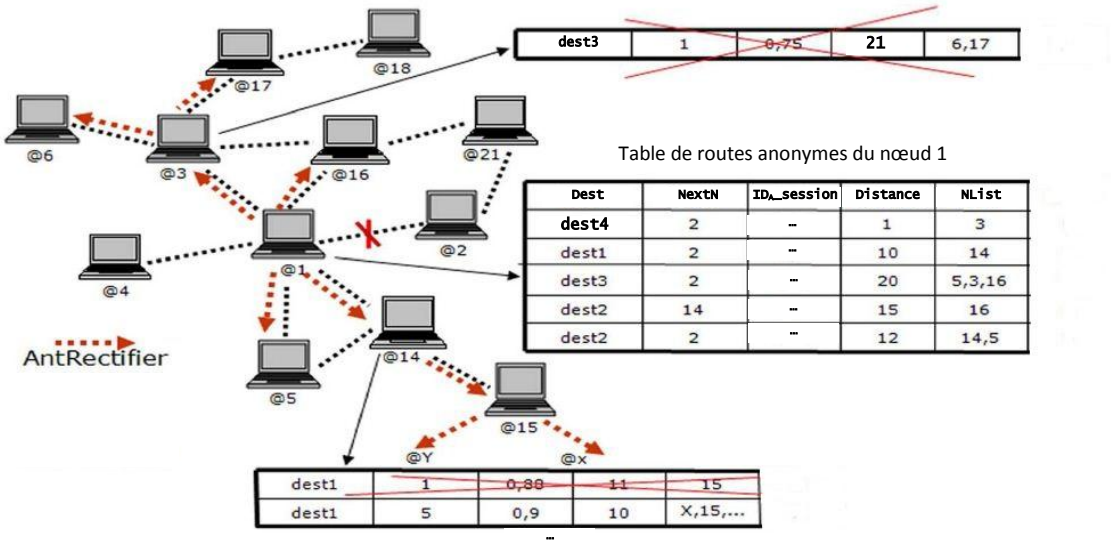
$E_{Kdf}(f_i,i,ID\_session)$ est la partie f_i de la ressource, son numéro d'ordre et <ID_session> qui sont chiffrés à l'aide du secret K_{df} échangé entre le demandeur et le fournisseur durant le processus de recherche et de réponse, et $sign_{Kdf}(f_i,ID\_session)$ est une signature de f_i et <ID_session> utilisant K_{df} pour permettre d'assurer l'intégrité du message envoyé. Le message construit est chiffré à son tour par la clé publique PK_{AV} où V représente le prochain nœud voisin qui se trouve sur le chemin 'anonyme' construit précédemment.

3.3.2.4. La gestion des paquets perdus

Il se peut qu'un nœud demandeur ne reçoive pas la totalité des parties de ressources envoyées par un nœud fournisseur (dû essentiellement à une perte de paquets dans le réseau). Dans ce cas, le protocole est doté d'un moyen qui permet au nœud de demander de façon 'anonyme' les messages manquants. Le nœud demandeur, au travers des routes 'anonymes', va envoyer la requête suivante : $E_{PKdv}(<ID_d>, <ID_f>, E_{Kdf}(<liste\_indices>, <ID\_session>))$, où $<ID_d>$ est l'adresse virtuelle (celle au moment de la demande) du demandeur des parties de la ressource, $<ID_f>$ est l'adresse virtuelle du nœud fournisseur de la ressource, *<liste_indices>* représente la liste des indices des messages manquants et *<ID_session>* est le numéro permettant d'identifier la ressource demandée. Les valeurs *<liste_indices> et <ID_session>* sont chiffrées à l'aide du secret K_{df}. Cette requête sera chiffrée par la clé publique PK_{AV} où V représente le prochain nœud voisin qui se trouve sur le chemin 'anonyme' qui mène vers le fournisseur. Dès que le nœud fournisseur aura reçu ce message, il va alors réémettre les f_i manquants en appliquant le même processus décrit plus haut.

3.3.2.5. La gestion des pertes de liens avec les voisins

Lorsqu'un nœud perd un lien entre un ou plusieurs de ses voisins, une mise à jour des tables de routes anonymes est nécessaire. Pour cela, il envoie un message de contrôle (l'agent AntRectifier dans la figure 3-9) 'rectificateur' à ses voisins (les voisins figurant dans le champ NList de la table de routage, cf. tableau 3-1) les informant de cette perte de liaison, cela permet aux nœuds voisins de mettre à jour leur tables de calcul de routes 'anonymes' et d'en informer à leur tour leurs propres voisins. De cette façon, les tables de routages 'anonymes' sont mises à jour et rafraichies à chaque changement de topologie du réseau (cf. figure 3-9).



desti: L'adresse virtuelle de la destination
NextN: L'adresse IP du prochain nœud vers desti
NList: La liste des voisins concernés par cette destination (les voisins utilisant le route en question)

Figure 3-9- La gestion des pertes de liens avec les voisins

3.3.3. Analyse des performances, de la sécurité et de l'anonymat des échanges dans AnonymP2P

Dans cette section, nous allons dans un premier temps discuter des caractéristiques originales de AnonymP2P puis analyser le protocole sous les aspects de la confidentialité et l'anonymat des échanges durant les phases de demandes et de transferts de ressources entre les différents nœuds du réseau. Nous montrerons aussi comment le protocole résiste à certaines attaques les plus utilisées dans les réseaux P2P.

3.3.3.1. Caractéristiques de AnonymP2P

Comme tout réseau P2P, les performances de notre protocole sont étroitement liées avec l'augmentation du nombre d'utilisateurs. Plus il y a de nœuds et de trafic et plus le protocole sera plus performant. Cependant, AnonymP2P possèdent quelques spécificités qui peuvent renforcer ses performances :

– Plusieurs routes sont construites (par les agents mobiles collaboratifs) entre un nœud X et un nœud Y et de ce fait les téléchargements seront plus rapides et le protocole sera plus robuste.

– Il n'y a pas de diffusion de recherches par le demandeur puisque la requête est déposée par un agent mobile sur les différents nœuds tout le long de sa route, il y aura donc moins de messages de contrôles et par conséquent moins de charge sur le réseau.

– Le nœud fournisseur peut posséder une route anonyme vers le demandeur avant même de recevoir la demande grâce aux agents mobiles collaboratifs qui travaillent d'une manière proactive. Dans ce cas, le fournisseur d'une ressource va envoyer directement la réponse sans attendre l'établissement des routes anonymes par le demandeur, ce qui permet d'avoir un délai de communication plus petit.

– Un agent peut transporter plusieurs requêtes et routes en même temps ce qui améliore le processus de recherche et d'établissement des routes 'anonymes' vers les demandeurs de ressources.

– Le protocole permet une topologie du réseau dynamique (changements de voisinage, disparition des nœuds, etc.), offre un système de gestion des pertes de paquets et des ruptures de liens entre les nœuds, ce qui permet la diminution des pertes de données dans le réseau.

3.3.3.2. Anonymat

Pour préserver l'anonymat des échanges entre les nœuds d'une manière générale, le protocole se base sur les points suivants :

– L'utilisation d'adresses virtuelles permet de cacher l'identité réelle (adresses IP) des nœuds demandeurs et fournisseurs de ressources.

– Le protocole empêche tout nœud attaquant d'être voisin avec tous les nœuds du réseau puisque de façon périodique et aléatoire chaque nœud du réseau peut changer de voisinage.

– Dans les protocoles existants, les transferts de données se font le long du chemin inverse de celui initié par la requête de demande (c'est le cas dans OneSwarm, Freenet, Mute et Ants). Dans le cas de AnonymP2P, il y a construction de plusieurs chemins anonymes de réponses qui sont généralement différents du chemin de la requête de demande. Cette originalité permet d'assurer en partie l'anonymat du demandeur et du fournisseur de ressources mais aussi de réduire l'attaque Man In The Middle lors de l'échange de secret qui permet de chiffrer la ressource et d'assurer ainsi l'intégrité de la ressource échangée.

– Nous avons vu dans la section 3.2.4 que le système OneSwarm [IPKA10] utilise des messages de maintien en vie pour rafraîchir les chemins qui expirent après trente secondes d'inactivité ; nous pensons que cette méthode peut permettre à un attaquant d'analyser le trafic en se basant sur cette caractéristique et déduire des sources de données. Dans notre cas, nous

utilisions les agents rectificateurs qui se déclenchent seulement lors des ruptures de liens entre les voisins. Cela permet aussi d'économiser la bande passante.

– Afin de renforcer la propriété d'anonymat, le protocole ne fera circuler dans le réseau que des paquets chiffrés qui ont sensiblement la même taille; de cette façon il est difficile aux attaquants qui 'écoutent' les paquets qui circulent de faire la distinction entre des paquets de contrôles et des paquets de données.

– Le changement de topologie évite à un attaquant d'entourer un nœud (dans le but de faire la correspondance entre son adresse réelle et son adresse virtuelle) quel que soit sa nature (demandeur ou fournisseur de ressources) en contrôlant ses voisins.

Dans ce qui suit, nous allons faire l'analyse de la propriété d'anonymat selon deux aspects : la protection de l'anonymat du demandeur et la protection de l'anonymat du fournisseur de ressources.

Anonymat du demandeur

Le protocole AnonymP2P assure l'anonymat du demandeur grâce aux propriétés suivantes :

– Les requêtes (demandes) ne sont pas diffusées par le nœud demandeur (ce qui est le cas de la majorité des protocoles P2P) mais diffusées grâce à la mobilité des agents (cf. figure 3-10). Il est difficile donc à un attaquant de déduire qui est à l'origine de la requête.

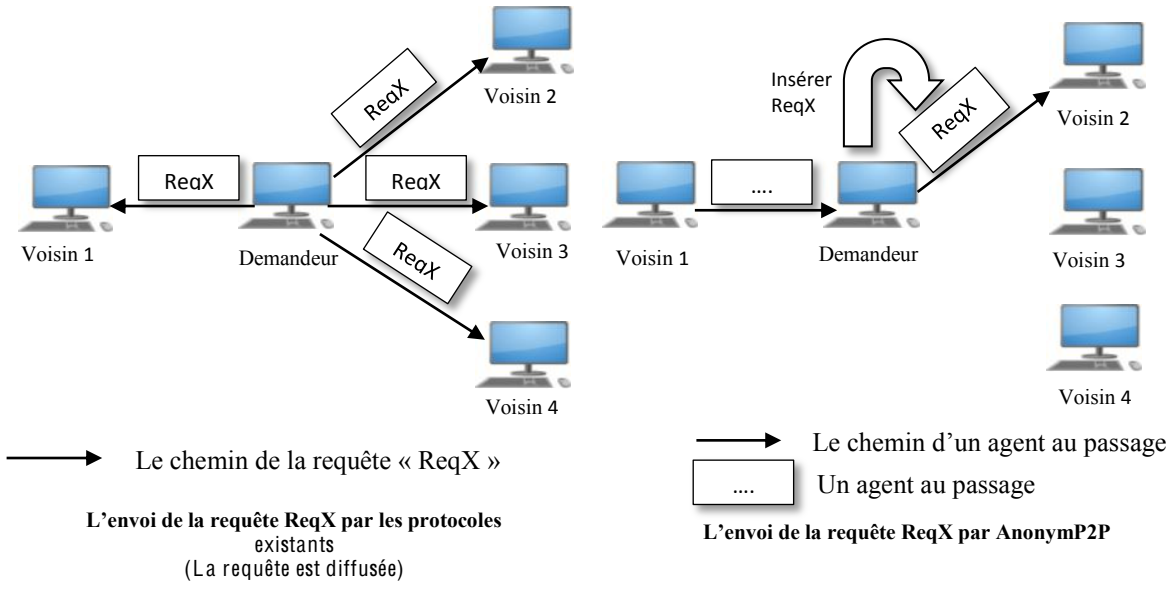


Figure 3-10- L'envoi de requêtes par AnonymP2P

– Dans OneSwarm, le client source d'une recherche peut mettre fin à cette recherche pour laquelle de nombreuses sources ont été découvertes ; cela se fait en envoyant un message dit « d'annulation de recherche » aux nœuds auxquels ils ont envoyé ou relayé le message de recherche. Nous pensons que cette méthode peut permettre à des attaquants de localiser le

demandeur par une étude de trafic. Dans AnonymP2P, chaque nœud possède des requêtes avec des TTLs qui correspondent aux nombres de fois que la requête peut être réexpédiée (forwardée) par le nœud courant (et non pas le nombre de sauts de cette requête), ce qui permet de perturber un attaquant qui essaie de déduire un demandeur en analysant les TTLs.

– Chaque nœud demandeur stocke et envoie les requêtes de demande des autres nœuds en plus des siennes. Par conséquent, un nœud voisin qui va recevoir une requête (à travers un agent) ne peut pas déduire à qui appartient cette requête, même s'il sait de quel nœud provient cet agent. Nous allons illustrer cela à travers l'exemple de la figure 3-11.

Exemple :

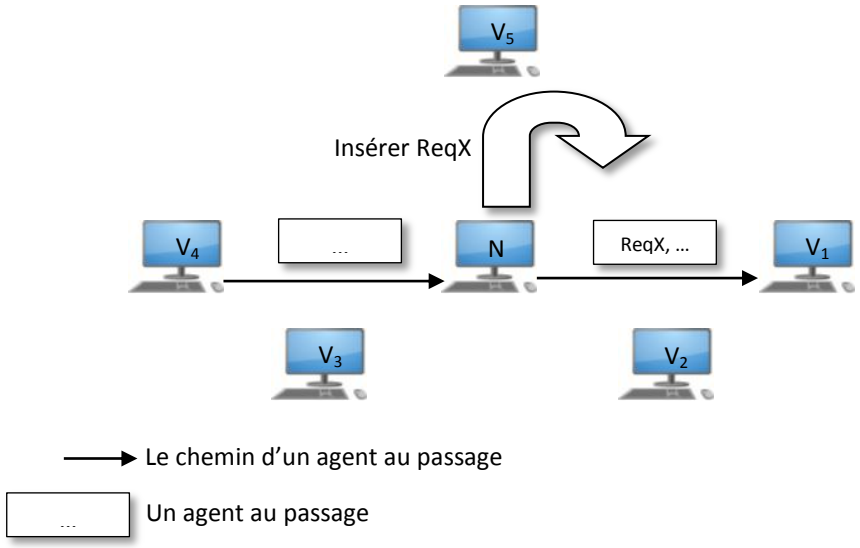


Figure 3-11- L'anonymat du demandeur

- Cas 1 **: Le nœud V1 est un attaquant**

Dans ce cas, le nœud V1 ne peut pas savoir si c'est le nœud N qui est la source de la recherche ou s'il s'agit juste d'un relais.

- Cas 2 **: Le nœud V1 et V4 sont des attaquants qui peuvent communiquer entre eux**

Dans ce cas, on suppose que le nœud V4 peut communiquer avec V1 et déduire que le nœud N a déposé une demande. Notre protocole résiste à cette attaque de la manière suivante :

Le nœud N n'enregistre pas seulement ses propres requêtes, mais il enregistre aussi dans son cache les requêtes des autres nœuds (qui peuvent être celles de V2, V3 ou V5 ou même une demande

qui vient d'un autre nœud du réseau) par conséquent rien ne prouve aux nœuds V1 et V4 que le nœud N soit l'initiateur de la requête.

- Cas 3 : Tous les voisins de N forment une « coalition » (ils peuvent communiquer entre eux)

Dans ce cas, tous les nœuds voisins de N peuvent communiquer entre eux et déduire que le nœud N a déposé une demande qui n'a jamais transité par ces voisins ; même si ce scenario est très peu probable, notre protocole résiste quand même à cette attaque de la manière suivante :

En réalité, les nœuds voisins ne peuvent pas déduire que c'est le nœud N qui est l'initiateur de la requête car il peut s'agir tout simplement d'une 'ancienne' requête qui existe dans le cache du nœud N et qui a été reçu via un 'ancien' voisin avant son changement de voisinage.

Anonymat du fournisseur

Le protocole AnonymP2P assure l'anonymat du fournisseur en résistant aux attaques suivantes:

– *Attaques basées sur le timing* [IPKA10]: l'attaquant mesure le temps total aller-retour (RTT) des couples requête/réponse dans le but d'estimer la proximité d'une source de données. L'attaquant peut être connecté à une source de données et peut ainsi déduire cette information ; cela est possible en se basant sur le faible RTT des paquets de réponse. Pour contrer cette attaque, OneSwarm utilise le principe d'augmentation artificielle des délais s'appliquant aux requêtes issues de pairs non fiables : toutes les réponses faites à des pairs non fiables subissent un délai aléatoire mais déterministe pour simuler le délai qui correspond à une requête propagée via un ou plusieurs intermédiaires de plus. Nous pensons que cette méthode peut causer une diminution de performances du protocole. Dans le cas du protocole AnonymP2P, la route de la demande est souvent différente des routes de la réponse, par conséquent un nœud qui tente cette attaque ne peut rien déduire. Dans le cas où le chemin de la requête soit le même que l'un de ceux de la réponse, on peut rajouter un petit temps d'attente aléatoire (pour simuler que la requête a été propagée via d'autres nœuds),

– *Collusion attaque* [IPKA10]: nous donnons un exemple de cette attaque en expliquant la figure 3-12 : Le nœud P envoie une recherche ciblée vers le nœud C ; il reçoit ensuite une réponse et vérifie si cette recherche a été relayée vers ses complices P1,...., Pk. Dans le cas où cette recherche n'a pas transité vers les nœuds P1,...., Pk alors le nœud P peut déduire qu'il s'agit d'une source de données.

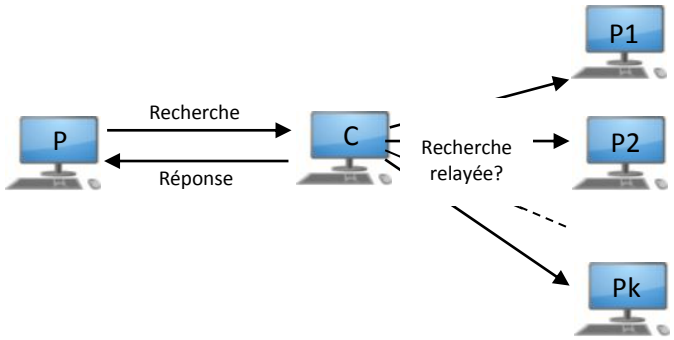


Figure 3-12- L'attaquant P avec ses complices P1,…Pk testent si le nœud C partage un fichier

L'efficacité de cette attaque exige que le nœud P et ses complice P1, P2,..Pk soient directement connectés à C. AnonymP2P peut résister à cette attaque car le changement de topologie évite de construire une topologie « fixe » comme celle de la figure3-12. Ainsi, dans AnanymP2P, une requête n'est pas systématiquement « diffusée» à tous les voisins mais envoyée juste à quelque uns choisis aléatoirement par les agents mobiles. Par conséquent, si les nœuds P1, P2,…Pk ne reçoivent pas la requête, le nœud A ne peut rien déduire car il se peut que les agents ne soient pas passés par les nœuds P1, P2,…Pk. Ainsi, en recevant une réponse, le nœud P ne peut pas déduire que le nœud C est une source de donnée car il peut s'agir d'une réponse générée par un voisin du nœud C avant son changement de voisinage.

– *Déconstruction des chemins* [IPKA10]: Cette attaque vise à localiser une source de données quelconque pour une ressource particulière. Le principe de cette attaque consiste à déconstruire le chemin menant à une source de donnée potentielle ensuite vérifier si elle partage la ressource. L'attaquant avec ses complices utilisent les mesures coordonnées de la propagation des messages de réponses (résultats de recherche) dans le but de :

- Déduire le prochain pair (le plus probable) le long d'un chemin ;
- Surveiller ou tenter de se connecter à ce pair ;
- Répéter l'opération.

Pour déconstruire un long chemin, l'attaquant doit effectuer plus d'itérations sachant que le chemin peut expirer avant la finalisation de la tache de l'attaquant (à cause de l'aspect dynamique du réseau). Ainsi, l'existence de plusieurs chemins crée une direction vers les sources en constante évolution, ce qui brouille la déduction basée sur le RTT des messages de réponses aux recherches. Cette attaque a donc une relation avec la diversité, la stabilité et la longueur des chemins vers la ressource.

Dans AnonymP2P, le changement de topologie obligatoire et la multitude de routes anonymes entre le demandeur et le fournisseur de ressources permettent de résister contre cette attaque. Les attaquants n'auront même pas le temps de faire leurs analyses statistiques avant que la topologie du réseau et par conséquence les chemins anonymes ne soient modifiés.

3.3.3.3. Confidentialité et intégrité des données échangées

Le protocole AnonymP2P assure la confidentialité et l'intégrité des données échangées en se basant sur les points suivants :

– Comme expliqué précédemment, le fournisseur d'une ressource envoie sa réponse chiffrée ainsi que la deuxième partie de la clef Diffie-Hellman, via tous les chemins 'anonymes'. Le demandeur peut authentifier cette deuxième partie de clé grâce à la diversité des chemins de transfert qui sont différents du chemin de la demande. Cette façon de faire permet au demandeur et au fournisseur d'échanger un secret et de résister à l'attaque Man In The Middle.

– L'intégrité des messages échangés entre le demandeur et le fournisseur est assurée grâce à la clé secrète échangée durant les phases requête/réponse et l'utilisation d'un protocole de signature électronique.

3.3.4. Simulation, tests et résultats

Nous avons testé les performances de notre protocole en utilisant le simulateur de réseau P2P PeerSim qui est un outil open source écrit en java et qui présente l'avantage d'être spécialisé pour l'étude des systèmes P2P. Ce simulateur a été conçu pour être à la fois dynamique et évolutif. PeerSim utilise une approche modulaire, ce qui offre la possibilité de réutiliser les modules existants. Il dispose d'une architecture ouverte, multi couches évolutive et maintenable qui permet de l'adapter et de le spécialiser. Parmi les couches implémentées dans le simulateur, citons : la couche recherche d'information, la couche réseau logique qui gère les relations de voisinage, la couche gestion des ressources locale (documents), etc.

Pour la simulation de notre protocole, nous avons dû modifier l'architecture du simulateur PeerSim pour prendre en charge l'aspect sécurité et surtout l'aspect messages de contrôles qui fonctionnent à la manière d'un système multi agent mobiles. Le simulateur peut donc prendre en charge dans le futur la simulation de tous les protocoles qui implémentent le concept d'agents mobiles. Nous avons ajouté aussi le module routage (gestion des tables de routage) et le module gestion d'adresses virtuelles comme illustré dans la figure 3-13.

Figure 3-13- La nouvelle architecture de PeerSim

3.3.4.1. Mesures de performances (recherche d'information)

L'étude suivante montre le comportement des deux protocoles AnonymP2P et Gnutella [Ber03]. Gnutella est un protocole P2P très populaire qui ne possède pas les propriétés d'anonymat et de confidentialité. Une comparaison de leurs performances en termes de recherche d'information est réalisée. Nous avons mis en place des jeux de test en faisant varier:

– Le nombre de nœuds du réseau.
– Le nombre de requêtes dans le réseau.

La métrique étudiée correspond aux requêtes satisfaites ; ceci correspond au pourcentage des requêtes satisfaites dans le réseau (requêtes qui ont au moins une réponse).

Parmi les objectifs du protocole AnonymP2P : réduire l'impact de l'anonymat et de la confidentialité sur les performances. Nos simulations (voir figure 3-14 et 3-15) montrent que les performances obtenues en terme de nombre de requêtes satisfaites sont meilleures que le Gnutella [Ber03] qui pourtant n'intègre pas d'anonymat et de confidentialité dans les échanges. Ces performances sont dues essentiellement à l'utilisation optimale des agents collaboratifs et aux systèmes de gestion des pertes de paquets et des ruptures de liens entre les nœuds.

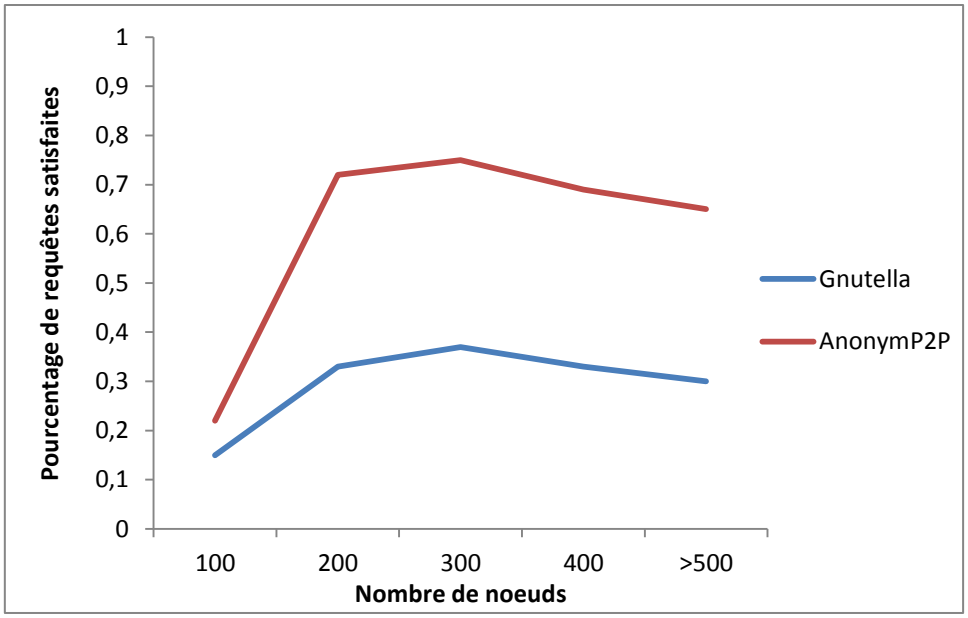


Figure 3-14- Pourcentage de requêtes satisfaites en fonction du nombre de nœuds

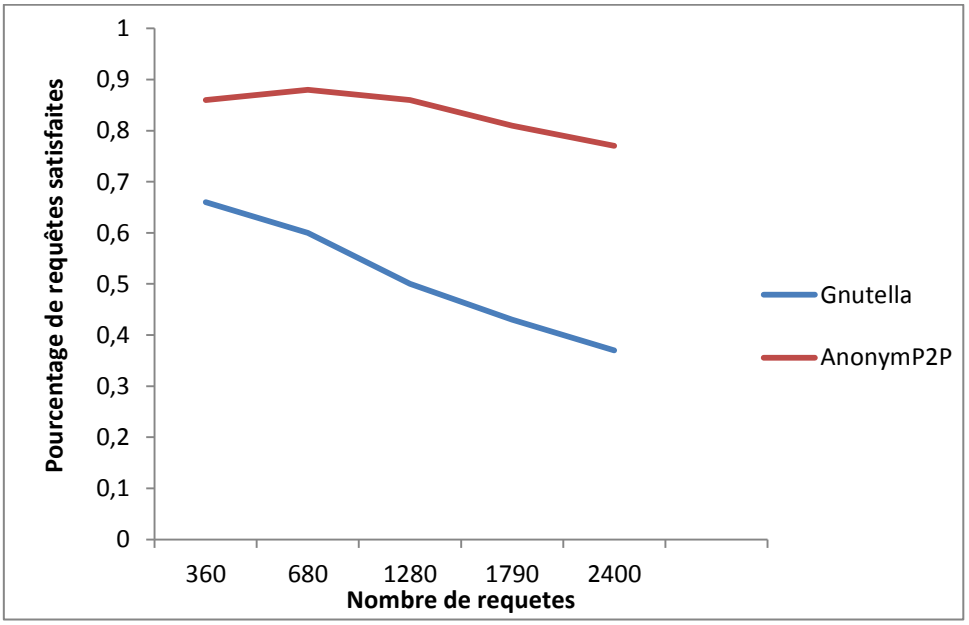


Figure 3-15- Pourcentage de requêtes satisfaites en fonction du nombre de requêtes

3.3.4.2. La diversité des routes

La figure suivante illustre la moyenne des nombres de routes 'anonymes' entre deux nœuds quelconques du réseau, en fonction du nombre de nœuds. Comme nous l'avons montré précédemment, la grande diversité des routes anonymes augmentent le facteur d'anonymat ainsi que l'intégrité des ressources échangées.

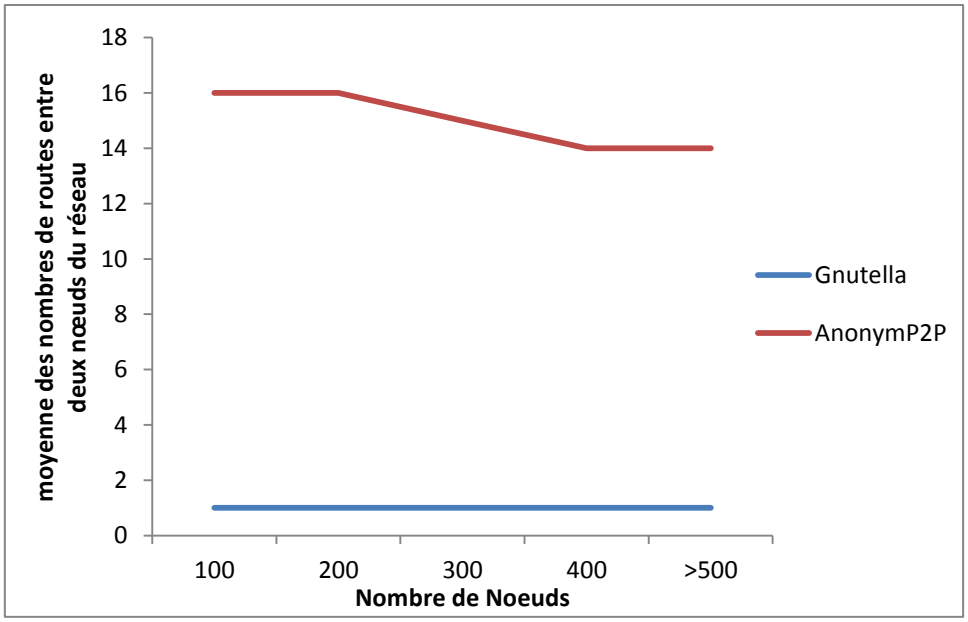


Figure 3-16- La moyenne des nombres de routes anonymes entre deux nœuds quelconques du réseau, en fonction du nombre de nœuds.

La figure 3-16 montre qu'il y a une multitude de routes 'anonymes' entre un nœud source et un nœud destination avec AnonymP2P ; ceci offre les avantages suivants :

– Les téléchargements (les transferts) d'une manière générale seront rapides, multi sources et tolérants aux pannes et/ou la congestion des nœuds intermédiaires.

– Il y a une meilleure résistance contre l'attaque « déduction de la topologie du réseau » (étudiée dans la section 3.3.3.2).

– Le protocole résiste mieux à l'attaque Man In The Middle lors de l'échange de la clé secrète entre le demandeur et le fournisseur (cf. section suivante).

3.3.4.3. Résistance du protocole à l'attaque Man In The Middle

Afin d'estimer la probabilité de réussite de l'attaque Man In The Middle lors de l'échange de secret entre le demandeur et le fournisseur, à l'aide du protocole Diffie-Hellman, nous considérons les éléments suivants:

– Nous supposons qu'il existe dans le réseau un attaquant qui peut contrôler un groupe composé de *nbmalicieux* nœuds dans le réseau.

– Nous considérons *nbnœuds* le nombre total de nœuds dans le réseau, *nbmoyen* le nombre moyen de nœuds contenus dans un chemin entre deux nœuds quelconques du réseau et *nbchemins* le nombre moyen de chemins reliant deux nœuds quelconques du réseau.

La probabilité que tous les nœuds d'un chemin quelconque ne soient pas compromis est donnée par la formule suivante:

$$P = \frac{C_{nbmoyen}^{(nbnœuds - nbmalicieux)}}{C_{nbmoyen}^{nbnœuds}} \qquad (1)$$

Par conséquent, la probabilité qu'il existe au moins un nœud compromis entre les nœuds d'un chemin quelconque reliant deux nœuds du réseau est:

$$(1-P) \qquad (2)$$

L'attaque Man In The Middle peut réussir dans notre cas s'il existe au moins un nœud compromis sur tous les chemins reliant deux nœuds quelconques du réseau. Donc la probabilité de réussite de cette attaque est donnée par la formule suivante :

$$P_{MAn\text{-}In\text{-}Middle} = (1-P)^{nbchemins} \qquad (3)$$

Pour estimer la valeur de probabilité décrite dans l'équation (3), nous considérons les valeurs suivantes:

– Selon la figure 3-16 et les simulations que nous avons effectuées, nous avons constaté qu'avec un nombre total de *nbnœuds* =600, le nombre moyen de chemins entre deux nœuds quelconques du réseau est stabilisé à *nbchemins* = 14.

– Les simulations ont montré aussi que le nombre moyen de nœuds intermédiaires dans un chemin reliant deux nœuds du réseau est *nbmoyen* = 18.

– Nous supposons qu'il existe un nœud qui contrôle 10% de tous les nœuds du réseau (ce qui est considérable pour un réseau P2P), donc *nbmalicieux* = 60.

L'équation (3) nous donne alors: $P_{MAn\text{-}In\text{-}Middle} = 0.11$

Ce résultat confirme ce que nous avons discuté précédemment.

3.3.4.4. Anonymat du fournisseur

Les graphes des Figures 3-17 et Figures 3-18 montrent le pourcentage de réponses qui sont envoyées (par le fournisseur d'une ressource) au nœud voisin qui a relayé la requête correspondante en fonction du nombre de nœuds et du nombre de requêtes. Nous remarquons et confirmons qu'avec

AnonymP2P le fournisseur envoie « rarement » la ressource sur le chemin inverse de la requête mais plus tôt par d'autres chemins, contrairement à Gnutella qui utilise le chemin inverse du chemin de la demande. Cette diversité des routes de transfert permet de protéger l'anonymat du fournisseur contre certaines attaques décrites précédemment. Cela confirme aussi notre analyse de sécurité (cf Attaques basées sur le timing).

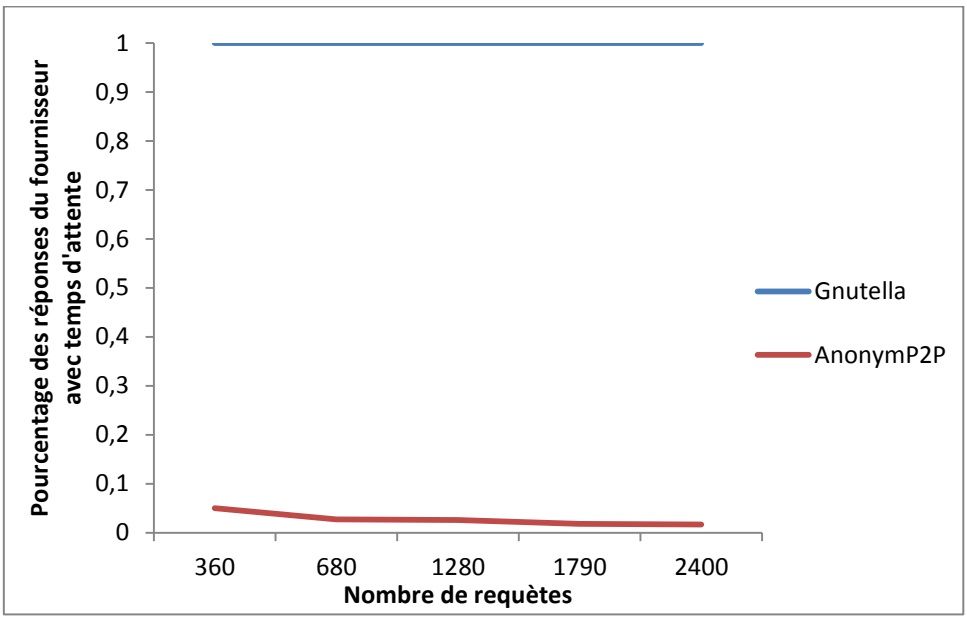


Figure 3-17- Le pourcentage de réponses qui sont envoyées (par le fournisseur d'une ressource) au nœud voisin qui a relayé la requête en fonction du nombre de requêtes

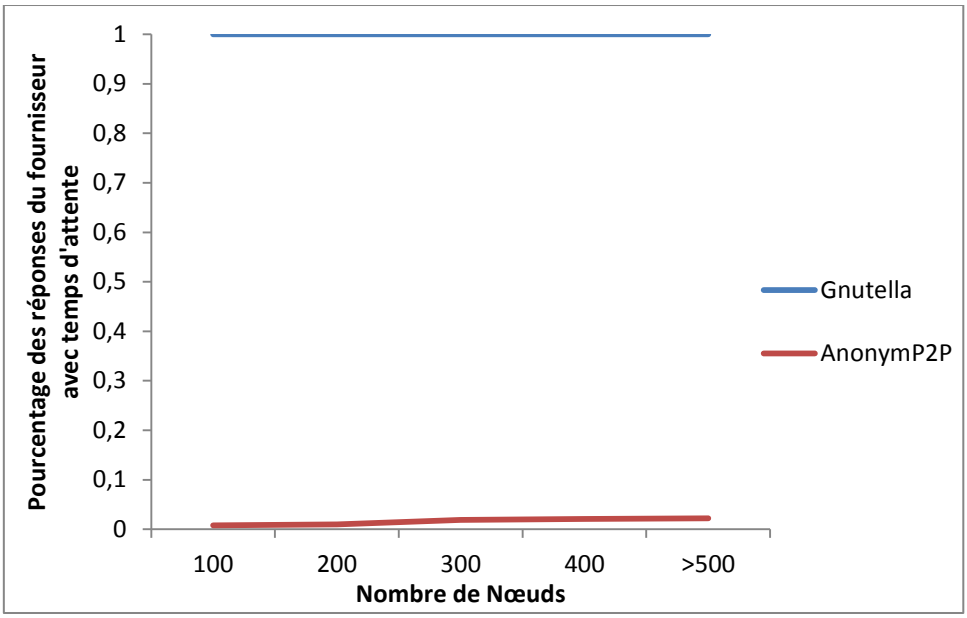


Figure 3-18-Le pourcentage de réponses qui sont envoyées (par le fournisseur d'une ressource) au nœud voisin qui a relayé la requête en fonction du nombre de nœuds

3.4. Conclusion

Dans ce chapitre nous avons traité le problème de la protection de la vie privée dans un réseau P2P distribué. A cet effet, nous avons opté pour la protection de l'anonymat des communicants (nœuds demandeurs et fournisseurs de ressources) ainsi que la confidentialité des ressources échangées. Les propriétés de préservation de l'anonymat et de confidentialité des échanges dans les réseaux P2P restent un vrai chalenge et peuvent, à l'avenir, devenir un enjeu de plus en plus important pour les utilisateurs de systèmes P2P qui ne souhaiteraient pas juste être des consommateurs mais aussi des producteurs de contenus. Ce sera aussi un moyen de se préserver contre certaines actions des fournisseurs d'accès à internet (FAIs) qui divulguent des informations sur les communications de leurs abonnés. L'augmentation du nombre d'utilisateurs, dû à l'apport de la préservation de la vie privée, aura comme effet immédiat l'amélioration des performances des systèmes P2P.

Nous avons présenté certains travaux proposés dans le cadre de la protection de l'anonymat des communicants dans un réseau P2P : il s'agit des systèmes Mute, Ant, Freenet et OneSwarm. Cette étude nous a permis de faire ressortir un certain nombre de problèmes et de vulnérabilités propres à ces systèmes.

Pour répondre en partie à ces problèmes, nous avons proposé AnonymP2P : un nouveau système de partage de ressources « adapté aux réseaux P2P non structurés» conçu pour assurer l'anonymat des utilisateurs, la confidentialité des échanges et qui comprend aussi des techniques avancées de recherche et de transfert de données. Ces techniques sont utilisées dans le but d'assurer l'efficacité, la robustesse, et la préservation de la sécurité des échanges. Parmi les idées originales du protocole, nous pouvons citer :

- *Le changement périodique de voisinage*, qui permet de donner au réseau un aspect plus dynamique en simulant la mobilité des pairs du réseau ce qui a pour effet de brouiller les pistes à un attaquant qui souhaite identifier les sources de données et/ou les demandeurs de ressources.
- *Les chemins suivis par les requêtes de demandes de ressources sont différents des chemins lors des transferts des ressources.* Cela permet de renforcer la propriété d'anonymat des nœuds et de résister aux attaques de type Man In The Middle durant l'échange de secret entre le demandeur et le fournisseur.

Les résultats d'analyse de sécurité et de simulation obtenus nous ont confirmé que le protocole AnonymP2P assure la confidentialité des échanges et permet des communications anonymes entre les demandeurs et les fournisseurs.

AnonymP2P utilise le concept d'agents mobiles car nous pensons que les réseaux de partage de ressources P2P basés sur l'approche orientée agents mobiles ouvrent de nouvelles perspectives pour s'attaquer à des problèmes plus complexes de sécurité, et de robustesse.

Deuxième partie : Routage dans les réseaux ad hoc

CHAPITRE 4

4. Les réseaux ad hoc

L'évolution que connait la technologie sans fil promet un avenir florissant pour les domaines de télécommunication. L'environnement résultant appelé environnement mobile occupe de plus en plus de place dans les communications personnelles et au sein des entreprises. Cela est dû, au fait qu'il ne présente aucune restriction sur la localisation des usagers en leur permettant une grande mobilité. De plus, il offre aux utilisateurs une grande flexibilité d'emploi, surtout lorsque il s'agit d'un déploiement dans des endroits où il est parfois impossible de le faire avec les réseaux filaires. Les réseaux informatiques sans fil, en pleine expansion, suscitent un grand intérêt de par les facilités qu'ils apportent : Actuellement, cette technologie est devenue indispensable pour assurer le fonctionnement de nombreux services informatiques et des télécommunications tels que la téléphonie, la messagerie électronique et l'accès à Internet.

Dans ce qui suit, nous présentons d'abord le concept de base des réseaux sans fils qui sont classés en deux grandes familles : La famille des réseaux sans fils « avec infrastructure » qui sont constitués d'un ensemble de stations de bases fixes connectées par un réseau filaire, et des hôtes mobiles qui communiquent entre eux via le réseau des stations de base et la famille des réseaux sans fils « sans infrastructure ». Ces derniers sont formés par un ensemble d'unités mobiles communiquant entre elles sans l'aide d'infrastructure fixe. Les principaux avantages de cette famille de réseaux restent la flexibilité et le support de la mobilité ainsi que la rapidité et le coût faible de déploiement, ce qui permet une large variété d'applications dans cette famille de réseau. Dans le reste de la section nous présentons les domaines d'application, les spécificités ainsi que les problèmes de sécurité propres aux réseaux ad hoc.

4.1. Les réseaux avec infrastructure

En mode infrastructure, le réseau sans fil est constitué au minimum d'un médium de communication sans fil (une infrastructure qui peut être un point d'accès ou une station de base) connecté au réseau filaire, et d'un ensemble de postes réseaux sans fil capables d'échanger des informations par propagation électromagnétique (figure 4-1). En effet, il se pose toujours le problème d'interférences pendant la transmission des ondes électromagnétiques (lors des échanges de données entre les terminaux mobiles) ; ce phénomène peut être produit lorsque plusieurs transmissions sont entamées en même temps en utilisant un même canal de transmission. Les solutions qui ont été proposées pour remédier à ce problème consistent à utiliser des protocoles qui gèrent l'accès et le

partage du médium sans fil [KRD06] ; dans ce contexte, on peut citer les protocoles les plus importants comme: le protocole FDMA (Frequency Division Multiple Access), le protocole TDMA (Time Division Multiple Access), et le protocole CSMA (Carrier Sense Multiple Access).

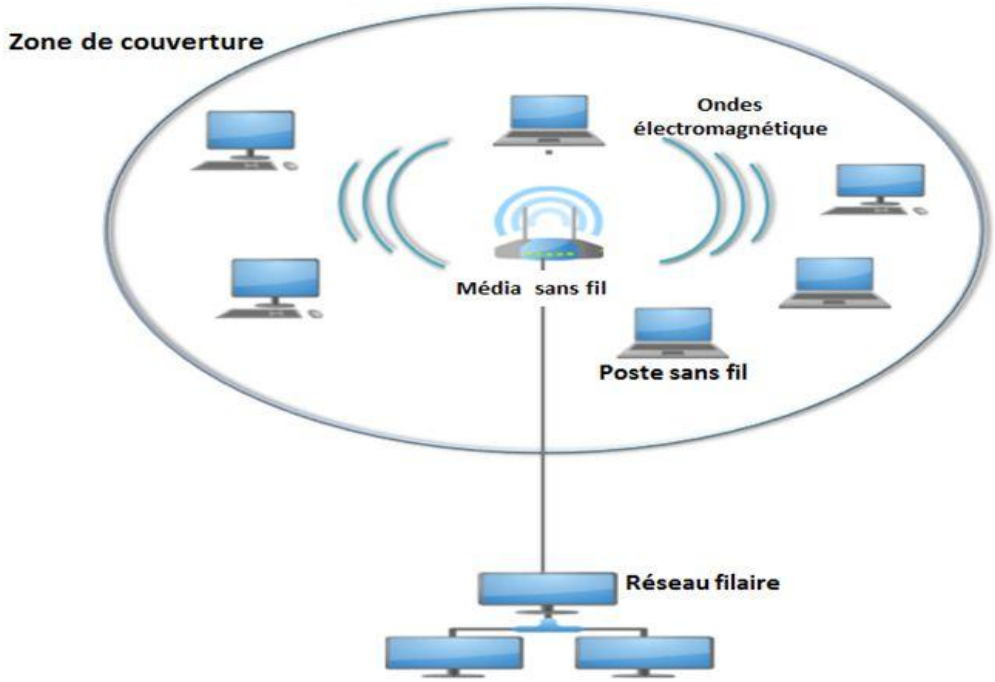


Figure 4-1- Les réseaux sans fils avec infrastructure

Un exemple de réseau sans fil avec infrastructure très connu et utilisé dans notre vie quotidienne est le réseau de la téléphonie mobile dit GSM [Hei99] (Global System for Mobile Communications). Cette famille de réseau, appelée aussi réseau cellulaire, est basée sur le principe de division du réseau en plusieurs cellules où chaque cellule dispose d'un émetteur-récepteur central nommé « station de base » muni d'antennes installées sur un point haut (château d'eau, immeuble ...). Les utilisateurs du réseau GSM sont des personnes équipées d'une entité mobile (un appareil téléphonique et une carte SIM) capables de communiquer avec les plus proches stations de base qui se trouvent dans des endroits géographiques distincts. La figure 4-2 montre un exemple d'une architecture de réseau GSM.

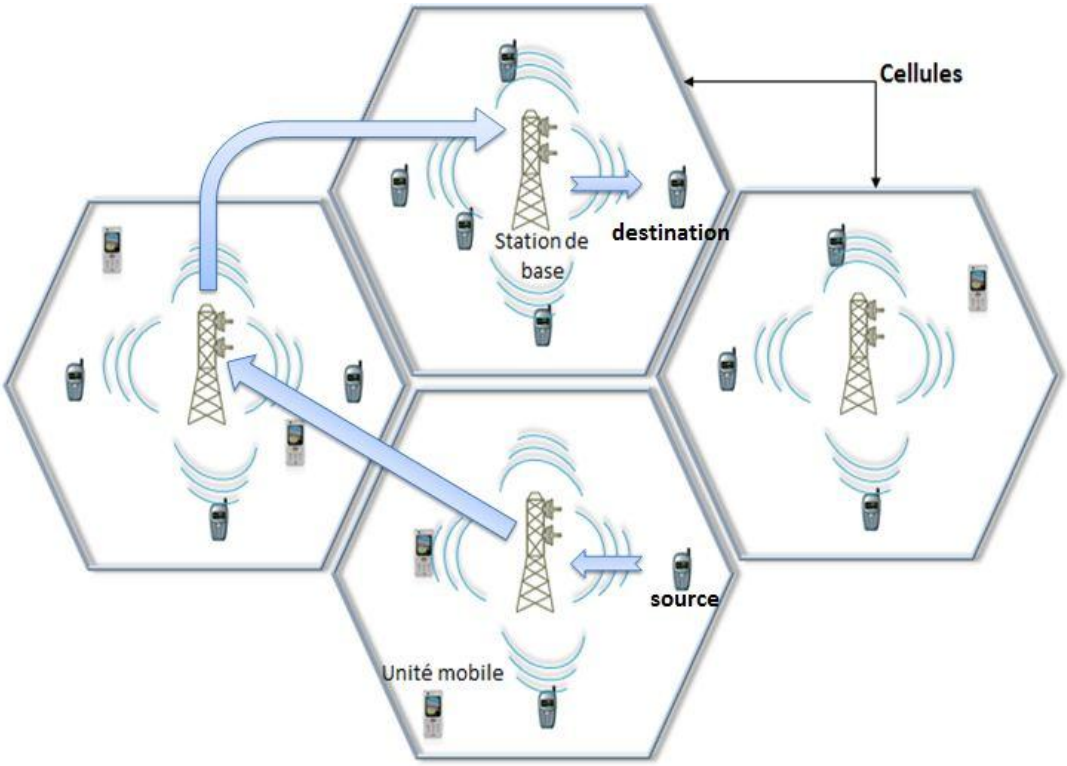


Figure 4-2- L'architecture d'un réseau GSM

4.2. Les réseaux sans infrastructure : le concept des réseaux ad hoc

Dans un réseau ad hoc sans fil, les nœuds, éventuellement mobiles, peuvent communiquer sans requérir la présence d'une infrastructure particulière lorsqu'ils sont à portée radio l'un de l'autre (cf. figure 4-3). Les réseaux ad hoc représentent donc une alternative intéressante aux réseaux à infrastructures, notamment pour les applications distribuées qui sont dynamiquement déployées, comme par exemple les applications dans le domaine militaire. Cependant, de nombreux défis doivent être relevés pour que les réseaux ad hoc puissent être effectivement exploités par les utilisateurs, pour le partage et l'accès aux ressources numériques offertes par les nœuds composant le réseau. Un des défis est lié aux caractéristiques intrinsèques des réseaux sans fil : une communication dans un réseau ad hoc est immédiatement interrompue dès lors que les terminaux impliqués se trouvent hors de leur portée radio respective, ce qui peut être fréquent dans le cas de terminaux fortement mobiles. Cette dynamique du réseau soulève le problème de la conception des protocoles de communication en général et des algorithmes de routage en particulier dans ce type de réseaux.

Figure 4-3- Les réseaux ad hoc

4.2.1. **Domaines d'application** des réseaux ad hoc

Les premières applications des réseaux ad hoc concernaient les communications et les opérations dans le domaine militaire. Cependant, avec l'avancement des recherches dans le domaine des réseaux et l'émergence des technologies sans fil (ex : Bluetooth, IEEE 802.11 et Hiperlan) ; d'autres applications civiles sont apparues. On distingue : Les services d'urgence, Le travail collaboratif et les communications dans des entreprises, les Applications commerciales, les réseaux de senseurs, les réseaux en mouvement (véhicules), etc.

4.2.2. Propriétés et spécificités des réseaux ad hoc

Des caractéristiques spécifiques aux réseaux ad hoc doivent être prises en compte lors de la conception des algorithmes et des protocoles de communication spécifiques à ce type de réseaux. Les plus pertinentes sont :

L'absence d'une infrastructure centralisée : chaque nœud travaille de façon totalement autonome, et agit en tant que routeur pour relayer des communications. La gestion du réseau ainsi que les protocoles d'échanges de services sont tous distribuée sur l'ensemble des éléments du réseau.

La mobilité des nœuds et maintenance des routes : Dans un réseau ad hoc les nœuds sont mobiles. Un nœud peut joindre un réseau, changer de position ou quitter le réseau de façon totalement aléatoire.

Ceci permet un changement dynamique de la topologie du réseau. Ce déplacement a un impact sur le comportement complexe des protocoles de communication. Les algorithmes de routage doivent ainsi résoudre ces problèmes et supporter la maintenance et prendre en charge en un temps limité la reconstruction des routes tout en minimisant la surcharge générée par les messages de contrôle.

La contrainte d'énergie : Les équipements mobiles disposent de batteries limitées, et dans certains cas très limités, et par conséquent d'une durée de traitement réduite. Sachant qu'une partie de l'énergie est déjà consommée par la fonctionnalité du routage, cela limite les services et les applications supportées par chaque nœud.

4.3. La sécurité dans les réseaux ad hoc

Dans les réseaux classiques, les utilisateurs sont sécurisés par plusieurs couches de défenses telles que les par feu et les systèmes de détection d'intrusions. Par conséquent, il est très difficile pour un attaquant d'effectuer une attaque car il doit franchir plusieurs barrières pour avoir un accès à la machine victime. Contrairement aux réseaux filaires, l'utilisation des liaisons sans fils dans les réseaux ad hoc a conduit inévitablement à l'apparition de nombreux problèmes de sécurité. En effet, les communications entre les stations mobiles ainsi que la transmission des données via des ondes radios ont facilité considérablement les attaques contre les réseaux ad hoc. La nature complètement distribuée et les caractéristiques des réseaux ad hoc posent ainsi des problèmes liés à la sécurité des échanges entre les nœuds, et le réseau devient plus vulnérable face aux différentes attaques. Comme tous les nœuds sont équivalents et potentiellement nécessaires au fonctionnement du réseau, il est facile pour un attaquant de s'insérer dans le réseau, ce qui lui permet par exemple de supprimer ou de modifier les informations de routage qu'il reçoit de ses voisins, ou bien d'injecter de fausses informations de routage. Ce comportement peut perturber le bon fonctionnement du réseau dans la mesure où le routage est considéré comme une tâche principale dans les réseaux informatiques. D'un autre côté, l'équivalence des nœuds et l'absence de la centralisation dans un réseau ad hoc rend la collecte d'informations pour la détection d'intrusions ou d'un déni de service plus délicate et plus complexe. De manière générale, les réseaux mobiles ad hoc sont considérés comme étant très fragiles en matière d'attaques en tout genre.

4.3.1. **Modèles d'a**ttaquant

Comme la majorité des réseaux informatiques, les réseaux ad hoc sont menacés par plusieurs types d'attaquants qui tentent d'effectuer des actions malveillantes, celles-ci pouvant paralyser le bon fonctionnement du réseau ou parfois causer des dégâts au niveau des ressources matérielles et logicielles des entités connectées au réseau.

Les **Nœud malicieux** : Dans un réseau ad hoc autonomes et auto-organisé les nœuds peuvent apparaitre ou disparaitre volontairement. Parmi ces entités mobiles qui constituent le réseau, il existe parfois des nœuds malveillants dont le seul objectif est d '« écouter » secrètement les trafics circulants entre les nœuds (on dit qu'ils sont passifs) ou parfois modifient les paquets qui transitent par ces nœuds (dans ce cas ils sont actifs). Les attaquants malicieux en général visent parfois un objectif très sensible qui est l'analyse des flux. Cette analyse est lancée lorsque les données échangées entre les nœuds sont masquées. L'analyse du trafic leur permet par exemple de déterminer la source et l'identité des nœuds en cours de communication ou la nature des communications échangées.

Les nœuds égoïstes : Un nœud peut avoir un mauvais comportement qui lui permet de ne pas effectuer correctement sa part de gestion du réseau (tel que le routage par exemple). Ce comportement non autorisé peut dégrader sévèrement les performances d'un réseau MANET ou causer involontairement des dégâts aux autres nœuds. Les nœuds ayant un mauvais comportement sont généralement des nœuds égoïstes et non collaboratifs qui souhaitent obtenir des avantages par rapport aux autres nœuds. Ils visent généralement à économiser leurs batteries d'énergie, ou bien élargir leur bande passante.

Attaquant interne-externe : Un attaquant interne est un nœud malveillant qui peut contrôler d'autres nœuds membres dans un réseau. Il peut disposer des informations relatives à un nœud ciblé pour effectuer une attaque donnée. Par exemple, plusieurs attaques peuvent être lancées en se basant sur la connaissance des informations contenues dans les tables de routage ou des clés secrètes de chiffrement. Par contre un attaquant externe n'est pas en possession de ces informations et par conséquent les actions possibles pour ce type d'attaquant vont se limiter à détecter la présence ou pas de communications et éventuellement à supprimer des paquets ce qui limite l'éventail des attaques qu'il pourra réaliser.

4.3.2. Description des principales attaques spécifiques aux réseaux ad hoc

Dans cette section, nous décrivons quelques attaques génériques spécifiques aux réseaux ad hoc. Cette partie sera complétée au chapitre 6 par la présentation d'attaques spécifiques aux protocoles de routages dans les réseaux ad hoc.

Attaque par déni de service : Dans les réseaux ad hoc, les attaques de type déni de service peuvent avoir plusieurs formes. Elles peuvent viser la bande passante du réseau, le trafic circulant entre les nœuds mobiles, les ressources énergétiques des nœuds ou la qualité de la transmission. Les attaquants sont capables par exemple de faire un brouillage au niveau du canal radio pour empêcher toute communication dans le réseau. Ils tentent aussi de déborder ou falsifier des tables de routages des

nœuds servant de relais ainsi qu'ils peuvent disperser, modifier ou supprimer le trafic en jouant sur les mécanismes de routage.

Privation de sommeil : Cette attaque est rencontrée dans les réseaux ad hoc où les nœuds sont caractérisés par des ressources énergétiques limitées. L'attaque s'appuie sur l'idée de demander de façon permanente des services à un nœud victime, cela influence négativement sur sa puissance et ses ressources système d'une part, et d'autre part, ça empêche d'autres nœuds du réseau d'échanger des données, des services ou des informations avec le nœud ciblé.

Spoofing attacks ou attaque Sybil **(usurpation d'identité)** : Dans un réseau ad hoc les nœuds sont identifiés de façon unique par leurs adresse MAC ou adresse IP. Un attaquant peut usurper l'identité d'un autre nœud du réseau en modifiant son adresse dans les champs des messages qu'il délivre. Il peut ainsi recevoir les paquets destinés aux nœuds légitimes, ou former une boucle de routage entre les nœuds.

Attaque par trou de ver : Ce type d'attaque suscite l'intérêt de la communauté scientifique car elle est particulièrement difficile à contrer. Elle peut être lancée par une collaboration d'au moins deux attaquants externes situés sur des zones géographiques distinctes. Le but de cette attaque est de former un tunnel qui permet de router des messages entre deux nœuds corrompus. Ici nous allons présenter une version de cette attaque, qui peut être lancée pour falsifier la longueur des routes établies par un protocole donné. Cette attaque fonctionne sur la base d'un protocole de routage réactif dont le fonctionnement sera présenté en détail dans le chapitre suivant.

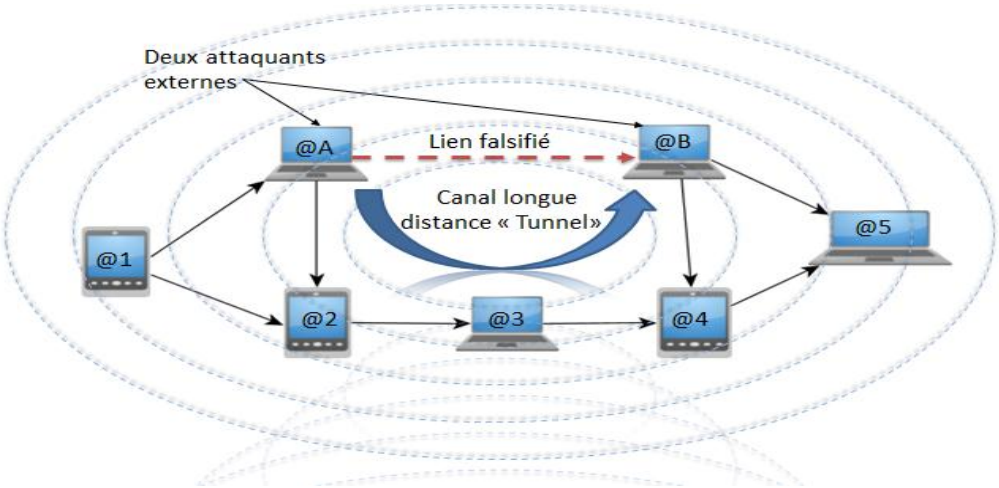


Figure 4-4- Attaque par trou de ver

Dans le cas de la figure 4-4 ci-dessus, les attaquants A et B se sont insérés dans le réseau et capturent les paquets circulant entre les nœuds. Admettons que ces deux intrus font partie d'une

opération de découverte d'une route reliant les nœuds (1) et (5), et le protocole utilisé s'appuie sur le nombre de sauts comme une métrique d'évaluation des routes. Lors d'initiation du processus de découverte de route, le nœud (1) propage un paquet RouteRequest via le réseau. Une fois que l'attaquant A capture ce paquet, il l'encapsule et l'envoi à son complice B via le tunnel (A, 2, 3, 4, B). B à son tour transmet le paquet à la destination sans avoir effectué la mise à jour du nombre de sauts qui indique que le message a traversé les nœuds (2, 3, 4). Le même mécanisme est suivi pour transmettre le paquet de réponse. De cette façon, le nœud (1) choisira aveuglement la route qui passe par le trou de ver car le nombre de sauts de cette route est très inférieur par rapport à celui passant par le nœud (2). Cela permet donc aux attaquants de créer une fausse route qui leur permet de garder le trafic qu'ils souhaitent.

4.4. Conclusion

Nous avons présenté dans ce chapitre les concepts de base des réseaux ad hoc en faisant ressortir leurs propriétés les plus importantes à savoir le manque de ressources matérielles, l'utilisation de lien sans fil, et les changements fréquents de topologie. Ces propriétés engendrent de nouveaux problèmes qui ne sont pas rencontrées dans les réseaux classiques et représentent des contraintes supplémentaires dans la conception des services de base du réseau. Parmi ces services figure celui du routage des données dans le réseau.

Les algorithmes de routage dédiés aux environnements ad hoc doivent tenir compte des changements fréquents dans la topologie (qui peuvent engendrer des pertes de paquets dans le réseau). En effet, un haut degré de mobilité des nœuds conduit à des calculs de routes et des échanges de paquets de contrôle fréquents, ce qui engendre par conséquence une grande consommation de la bande-passante, d'énergie et de puissance de calcul ainsi que l'augmentation des délais de communication. Les protocoles de routage conçus pour les réseaux ad hoc doivent donc prendre en considération ces contraintes et par conséquent les techniques de routage des réseaux classiques, basées sur des routes préétablies par des équipements spécialisés et dédiés (routeurs), ne peuvent plus fonctionner correctement sur un réseau ad hoc vu l'absence de ces équipements. Nous présentons dans le prochain chapitre, les différents protocoles de routage dédiés aux réseaux ad hoc en mettant en avant leur classification ainsi que leurs avantages et leurs inconvénients.

La majorité des protocoles de routages dédiés aux réseaux ad hoc font l'hypothèse du « bon comportement » des nœuds lors du routage des paquets. Dans le cas des réseaux filaires, ce problème ne se pose pas, puisque le routage est assuré par des entités de confiance appelées routeurs, ce qui n'est pas le cas des réseaux ad hoc où la tâche du routage est assurée par les nœuds du réseau. Un

nœud peut en effet rejeter les paquets en transit qu'il reçoit dans le but d'économiser son énergie (nœud égoïste).

Nous présentons en détail dans les chapitres suivants notre contribution: la proposition d'un nouveau protocole de routage dédié aux réseaux ad hoc en tenant compte des spécificités de ce type de réseau et d'autre part d'un modèle de renforcement de la coopération des nœuds intégré au nouveau protocole de routage afin d'inciter chaque nœud à « router » les paquets des autres nœuds.

CHAPITRE 5

5. Le routage dans les réseaux ad hoc

5.1. Le routage dans les réseaux ad hoc : Etat de l'art

La fonctionnalité principale que doit garantir un réseau de communication est l'acheminement des données entre les expéditeurs et les destinataires ; les réseaux mobiles ad hoc doivent donc fournir un mécanisme d'établissement de chemins entre les nœuds distants leurs permettant d'établir des communications et d'échanger des données via le réseau.

La sélection des routes entre les nœuds mobiles doit être réalisée de façon optimale ; cette condition devra prendre en considération toutes les limitations imposées par les caractéristiques des réseaux ad hoc. En raison de ces contraintes, les nœuds doivent être dotés d'un protocole de routage basé sur une stratégie de gestion prédéfinie.

Vu les limitations engendrées par les caractéristiques des réseaux ad hoc qui les différencient des réseaux classiques (grande mobilité des nœuds, faible puissance de calcul, ressources énergétiques limitées), les protocoles de routage proposés pour les réseaux classiques ne peuvent pas être appliqués aux réseaux ad hoc. Pour cela, plusieurs protocoles de routage pour les réseaux ad hoc ont été proposés.

Il existe fondamentalement deux grandes catégories de protocoles de routage pour les réseaux ad hoc : les protocoles uniformes et les protocoles non uniformes (figure 5-1). Les protocoles uniformes se distinguent par le fait que les nœuds jouent tous le même rôle. Les protocoles sont de type non uniforme dans le cas où certains nœuds ont des fonctionnalités particulières dans la gestion du réseau.

Dans la classification uniforme, les protocoles peuvent être classés en deux sous catégories suivant la manière dont les informations de routage sont maintenues et échangées entre les nœuds. Les deux approches de routage qui se distinguent sont :

– Routage par vecteur distance.
– Routage par état des liens.

Dans l'approche du routage par vecteur distance, chaque nœud du réseau maintient localement une structure de données appelée table de routage. Les tables contiennent des informations sur les

différentes routes disponibles vers toutes destinations dans le réseau. Des mises à jour sont effectuées entre les nœuds pour sauvegarder des nouvelles informations obtenues lors des changements topologiques, ou bien pour mettre à jour des entrées déjà existantes dans les tables de routage. Le routage par vecteur distance offre quelques avantages tels que la simplicité de la mise en œuvre de ses algorithmes, et la décentralisation du réseau qui correspond exactement aux caractéristiques des réseaux ad hoc. Cependant, nous pouvons citer quelques inconvénients :

– Les échanges de mises à jours fréquents,
– La taille des informations de routage qui est proportionnelle au nombre des nœuds disponibles,
– Le problème de bouclage qui conduit vers une saturation dans la bande passante du réseau.

Un exemple d'un protocole de routage qui utilise des algorithmes de type vecteur distance est le protocole DSDV [PB94] qui sera présenté plus en détail par la suite.

Les protocoles à état de liens ont été conçus pour pallier les limitations des protocoles de routage à vecteur de distance. Dans cette approche, chaque nœud dispose d'une vue locale sur l'état de liens[1] avec ses voisins adjacents; cette information est délivrée à l'aide des messages hello échangés périodiquement entre les nœuds. Lorsqu'un nœud souhaite émettre ses paquets à une destination, il sera capable de reconstituer la topologie globale du réseau ; cette opération est effectuée à l'aide de la diffusion des messages de contrôle topologiques (contenant les états des liens locaux des nœuds) par chacun des nœuds à la totalité du réseau, ce qui facilite ainsi le calcul des routes appropriées. Les protocoles de routage par état des liens ont pour avantage de répondre rapidement aux moindres changements de topologie dans le réseau. Cependant le maintien des états des liens de la totalité du réseau, requière une utilisation significative de la mémoire et la puissance de traitement des nœuds, ce qui peut ralentir les performances de cette famille de protocoles. OLSR [OLS03] est un exemple type de protocoles utilisant le principe de routage par état des liens et sera présenté plus en détail dans ce qui suit.

Les protocoles de routage non uniformes sont divisés en deux catégories, on distingue des protocoles à sélection de voisins, où chaque nœud décharge la fonction de routage à un sous ensemble de voisins directs, et les protocoles à partitionnement où le réseau est découpé en zones dans lesquelles le routage est assuré par un unique nœud maître.

[1] L'état de connectivité avec ses nœuds voisins

D'autres critères permettent de classifier les protocoles de routage dans les réseaux ad hoc ; ils s'appuient principalement sur le déclenchement des mises à jour des informations de routage. On distingue principalement trois approches différentes : les protocoles proactifs, réactifs et hybrides.

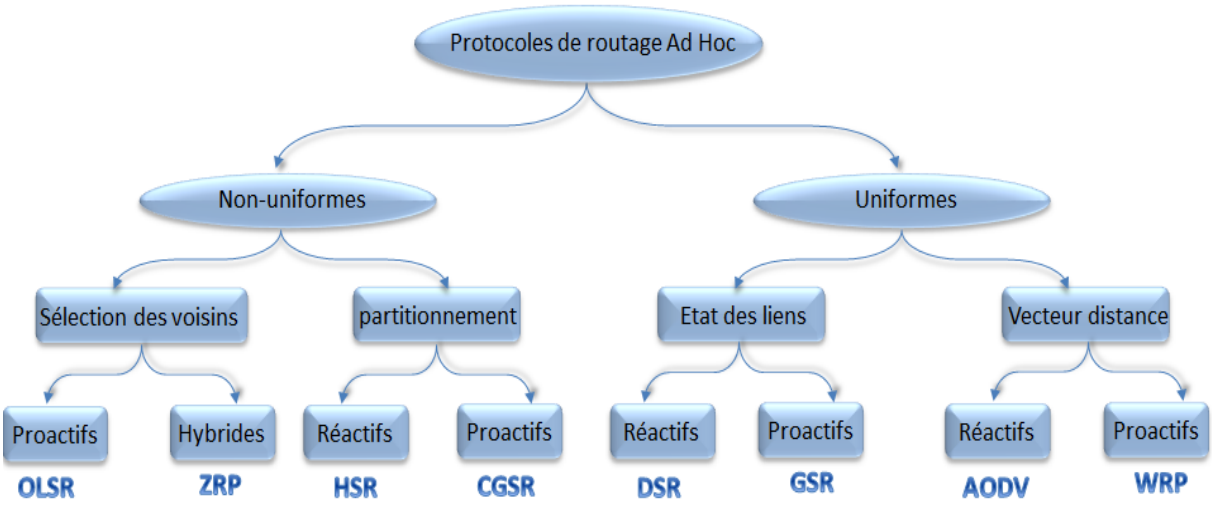


Figure 5-1- Les familles de protocoles de routage dans les réseaux ad hoc

5.1.1. Le routage proactif

Les protocoles proactifs sont basés sur le principe de l'échange régulier des messages de contrôles entre tous les nœuds du réseau, afin de maintenir au niveau de chaque nœud des tables de routage contenant des chemins valides vers tous les participants. Les routes sont donc préétablies de façon continue même si elles ne sont pas utilisées dans l'immédiat. Cette particularité permet d'avoir de bonnes aptitudes en termes de temps de réponse: les nœuds peuvent trouver à tout moment un chemin vers chaque destination atteignable via le réseau. Cependant, les diffusions périodiques des messages de contrôles (cf. figure 5-2) engendrent un surcoût significatif en matière de consommation de bande passante et de ressources énergétiques. Nous allons détailler par la suite deux exemples de protocole de routage qui utilisent l'approche proactive à savoir DSDV et OLSR.

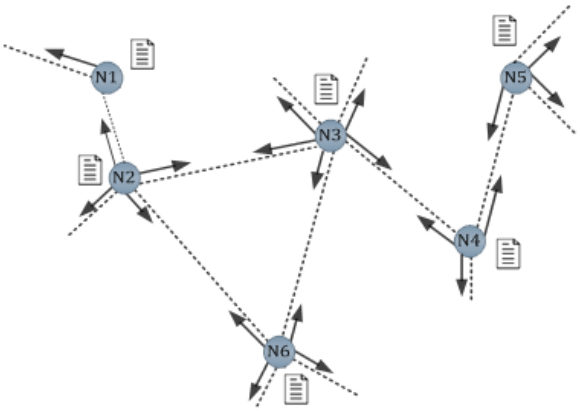


Figure 5-2- Le routage proactif

5.1.1.1. Le protocole DSDV

DSDV (Destination Sequenced Distance Vector Protocol) [PB94] est un protocole proactif basé sur l'algorithme de routage par vecteur distance.

Dans ce protocole, chaque nœud mobile maintient localement une table de routage dont les champs sont structurés comme suit: *< Destination>, <Nœud suivant>, <Nombre de sauts> <Numéro de séquence>* où le champ numéro de séquence est utilisé pour faire la distinction entre les anciennes et les nouvelles routes. Dans le but de maintenir la cohérence des routes, chaque nœud diffuse périodiquement à ses voisins directs (ou lors d'un changement topologique) sa table de routage encapsulée dans un paquet de mise à jour. Un nœud recevant ce paquet, compare les nouvelles informations des routes reçues avec celles dont il dispose localement avant de transmettre à son tour un paquet de mise à jour à ses voisins. S'il reçoit une route (ou des routes) qui existent déjà dans sa table de routage locale, il prend alors en considération la route ayant le plus grand numéro de séquence. Dans le cas d'existence de collision, (plusieurs routes ayant le même numéro de séquence) c'est alors la route de plus courte distance (en nombre de sauts) qui sera choisie. Ce processus est exécuté périodiquement par tous les nœuds du réseau. Apres chaque diffusion, le nœud incrémente le numéro de séquence pour la prochaine diffusion. Grace à ces numéros de séquences, DSDV élimine aussi la formation des boucles de routage dans le réseau. En plus de la forte consommation de la bande passante par les paquets de mises à jour, l'inconvénient de ce protocole réside dans le fait que les nœuds doivent attendre la réception du prochain paquet de mise à jour, avant de mettre à jour la route défaillante dans leurs tables de routage.

5.1.1.2. Le protocole OLSR

OLSR (Optimized link State Routing Protocol for Ad-Hoc Networks) [OLS03] est un protocole de routage considéré comme une amélioration des protocoles par état des liens dédiés aux réseaux filaires. Le protocole utilise un échange périodique des messages pour maintenir les informations sur la topologie du réseau. Son innovation réside dans sa nouvelle technique d'optimisation qui sert à limiter le nombre de retransmissions lors de la diffusion des paquets de contrôle. Pour cela, OLSR emploie des nœuds particuliers appelés les relais multipoint ou MPRs (multipoint relays). Ces relais sont un sous ensemble de voisins à un saut d'un nœud quelconque dans le réseau et sont sélectionnés par chaque nœud X du réseau de façon qu'il puisse atteindre un voisinage situé à deux sauts. Aucun nœud voisin n'a le droit de retransmettre un paquet OLSR sauf s'il est un relais multipoint (voir figure 5-3), ils peuvent seulement lire et traiter les paquets reçus. La nouvelle technique d'optimisation des diffusions du protocole OLSR permet d'économiser les ressources radio.

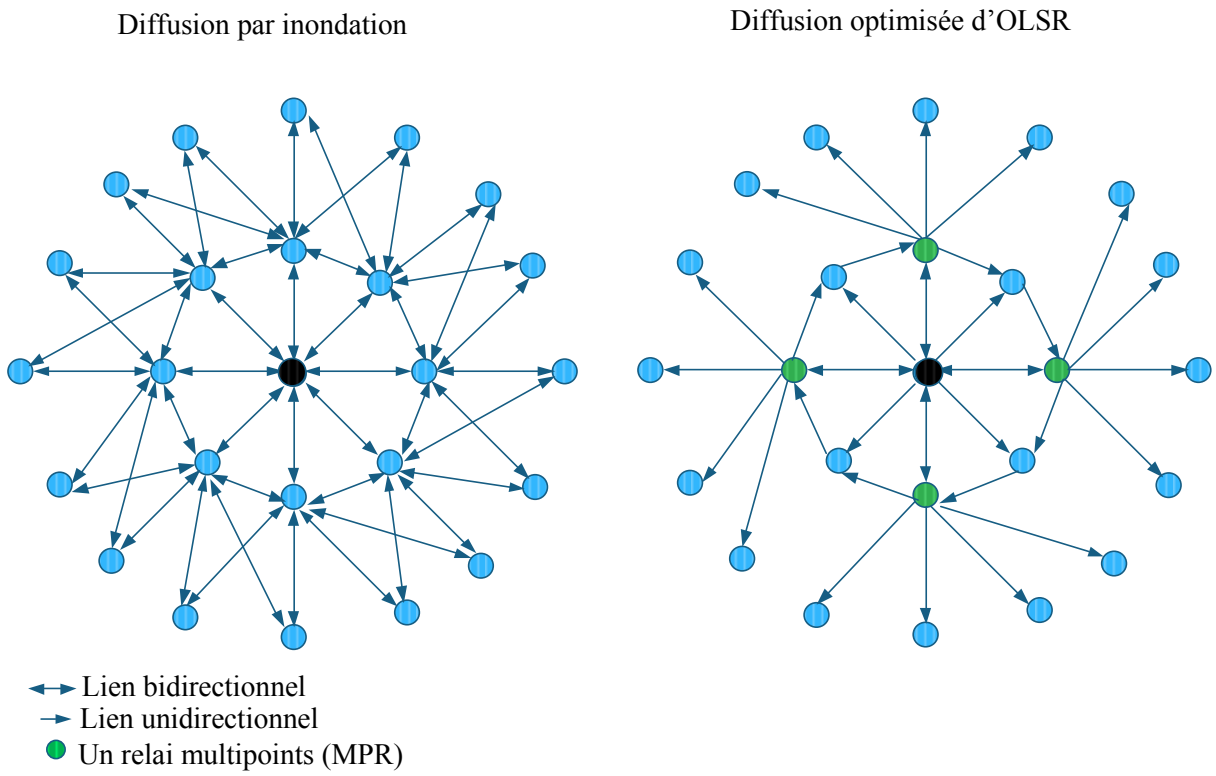


Figure 5-3- La diffusion optimisée de OLSR

5.1.2. Le routage réactif

Dans cette approche, les routes ne sont pas connues à l'avance mais sont établies et maintenues à la demande. A chaque fois qu'un nœud souhaite atteindre une destination dans le réseau, un processus de découverte de route est lancé ; cette procédure se déroule comme suit :

– Le nœud demandeur crée une requête de demande d'une route contenant les informations nécessaires à la recherche (adresses destination, nombre de sauts, etc.)

– La requête de demande va être relayée et propagée d'un nœud vers un autre, ce qui permet d'avoir une collaboration entre les nœuds du réseau dans l'opération de découverte de route.

– La dernière étape consiste à créer une réponse à la précédente requête ; cette réponse est envoyée par le nœud destination au nœud demandeur de la route, ou bien par un nœud intermédiaire connaissant une route vers cette destination.

Les informations obtenues par le processus de découverte de route peuvent être sauvegardées dans les mémoires caches des nœuds intermédiaires pour une utilisation ultérieure. L'avantage principal du routage réactif consiste à minimiser le flux engendré par les échanges périodiques de messages de contrôle. Par conséquent, il n'y aura pas de consommation de la bande passante comme

dans le cas du routage proactif. En revanche, cette approche de routage augmente le temps de réponse avant d'établir une communication entre deux nœuds et envoyer des messages. Ainsi, dans un réseau dense, un protocole réactif entraîne la diffusion excessive de paquets de demande de route qui risquent de saturer le réseau. Dans ce qui suit, nous allons détailler deux exemples de protocoles de routage réactifs appelés: DSR et AODV.

5.1.2.1. Le protocole DSR

DSR (Routage à Source Dynamique) [DBDAJ01] est un protocole réactif basé sur une méthode de routage dite par source.

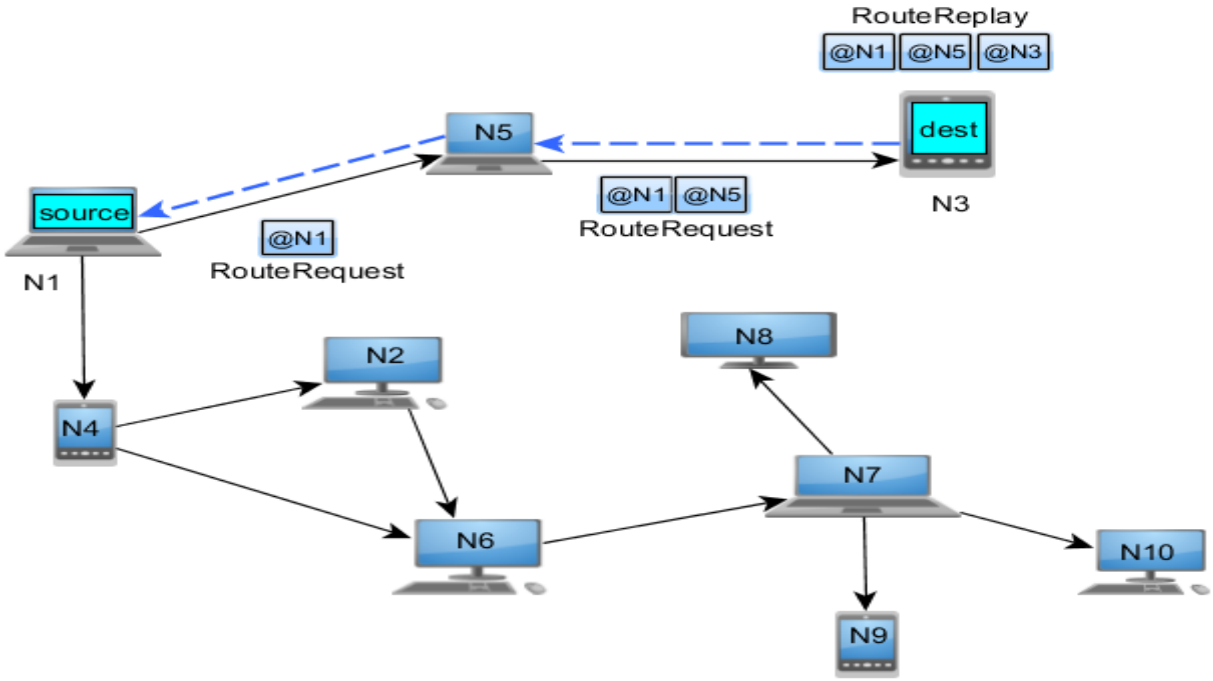


Figure 5-4- Le principe de fonctionnement de DSR

Lorsqu'un nœud veut communiquer avec une destination, il lance d'abord une opération de découverte de route. Pour se faire, il inonde le réseau par un paquet de demande de route appelé « RouteRequest ». La propagation de cette requête dans tout le réseau permet d'avoir une multitude de chemin entre la source de donnée et la destination. Le paquet « RouteRequest » contient un champ initialement vide ; il sert à sauvegarder les adresses des nœuds relayant le paquet jusqu'à ce qu'il atteigne la destination demandée (cf. figure 5-4) ou bien qu'un nœud intermédiaire possédant au moins une route vers cette destination. Dès qu'une route est trouvée vers la destination ; DSR applique une méthode de calcul pour choisir la route optimale en terme de nombre de sauts. Lorsque la route optimale est calculée, le nœud destination envoie une réponse au nœud demandeur par un

paquet appelé «RouteReplay » ; ce paquet contient des informations sur les nœuds intermédiaires qui construisent la route sélectionnée.

Le protocole DSR fournit un système de maintenance de routes tout au long de son utilisation. Ce mécanisme est basé sur la propagation d'un message d'erreur appelé « RouteError », qui est envoyé lors de la détection d'une défaillance de lien ou bien si aucune émission n'est reçue à partir d'un nœud pendant un moment donné. La création du message d'erreur est effectuée par le nœud qui détecte un problème lors d'une transmission. Ce nœud, met son adresse dans le message avec l'adresse du prochain nœud qui le suit dans le chemin puis propage le paquet vers le nœud source de la transmission. Quand celui-ci reçoit le message d'erreur, il supprime le nœud concerné par la route défaillante et tous les chemins qui contiennent ce nœud sont tronqués à ce niveau-là. Après cela, il lance une nouvelle opération de découverte de routes.

Comme tous les protocoles réactifs, DSR minimise le nombre de messages de contrôles échangés entre les nœuds du réseau. Cela est bien observé dans les réseaux qui sont caractérisés par un nombre réduit de nœuds sources communiquant avec des destinations rarement interrogées. Un autre avantage de ce protocole est l'existence des chemins multiples. Cette propriété permet d'acheminer les paquets correctement vers la destination même en cas des changements topologiques. L'inconvénient principal de ce protocole est observé lorsque le chemin reliant la source et la destination est long car dans ce cas-là le message « RouteRequest » devient grand puisqu'il doit porter une grande séquence d'adresses des nœuds intermédiaires, ce qui augmente la consommation de la bande passante du réseau.

5.1.2.2. Le protocole AODV

AODV (ad hoc On demand Distance Vector) [PBRD03] est un protocole réactif basé sur des algorithmes de routage par vecteur distance. Son fonctionnement est inspiré du protocole proactif DSDV combiné avec le protocole réactif DSR ; il hérite d'une part le principe des numéros de séquence et des tables de routage du protocole DSDV, et d'une autre part, il applique les mécanismes de découverte et de maintenance des routes utilisés par le protocole DSR (pour réduire le nombre de messages de contrôles dans le réseau).

Le processus de découverte de routes dans AODV est exécuté à chaque fois qu'un nœud cherche une nouvelle route vers une destination ou bien lorsque la durée de vie du chemin existant vers la destination a expiré. Pour lancer le processus de recherche de route, AODV utilise un paquet de demande de route dit « RouteRequest » (dont le format est illustré dans la figure 5-5) qui permet de créer un chemin vers une destination donnée. Le champ « numéro de séquence de la source » se

trouvant dans ce paquet est identique à celui de son nœud créateur, la valeur du champ « numéro de séquence de la destination » contient le dernier numéro associé à cette destination qu'a sauvegardé le nœud demandeur (dans sa table de routage) ; si cette valeur n'est pas connue alors c'est la valeur 0 qui sera prise par défaut. La requête « RouteRequest » transite d'un nœud vers un autre jusqu'au nœud destination. Au fur et à mesure, le protocole met à jour les tables de routage de chacun des nœuds traversés lors de la recherche.

@ source	Numéro de séquence de la source	Numéro de diffusion	@ destination	Numéro de séquence de la destination	Nombre de sauts

Figure 5-5- Le format du message RouteRequest

Chaque nœud recevant un message « RouteRequest» procèdera comme suit :

– En premier lieu, il conserve dans sa table de routage l'adresse du nœud qui lui a transmis la requête ; ce nœud sera considéré comme étant le prochain saut vers le nœud source du message « RouteRequest ». Cette information permettra par la suite d'orienter le paquet de réponse jusqu'au nœud source.

– S'il possède la route demandée dans sa table de routage, il envoie une réponse au nœud demandeur dans un paquet « RouteReplay » (le format du paquet est illustré dans la figure 5-6). Ce paquet emprunte le chemin inverse de la requête jusqu'au nœud source. Tous les nœuds situés sur le parcours de cette réponse, conservent dans leur table de routage le chemin vers le nœud destination.

– Sinon, il incrémente le nombre de sauts dans le paquet « RouteRequest », et propage cette requête à ses nœuds voisins afin de poursuivre la recherche.

@ source	@ destination	Numéro de séquence de la destination	Nombre de sauts	Durée de vie

Figure 5-6- Le format du message RouteReplay

Afin de maintenir la validité et la cohérence des routes, AODV utilise le principe de transmissions périodiques des messages hello ; un changement de topologie du réseau est déclaré si trois messages hello consécutifs ne sont pas reçus à partir d'un nœud voisin ; dans ce cas, le lien est considéré invalide. L'ensemble des nœuds qui utilisent le lien défaillant seront informés grâce à la propagation des messages d'erreurs « RERR ». À la réception de ces messages, les nœuds sources concernés par la route défaillante vont mettre à jours leurs tables de routage et initialisent une nouvelle opération de découverte de routes.

Le protocole AODV présente une convergence rapide lors des changements topologiques ; il élimine les boucles de routages grâce à l'utilisation du principe des numéros de séquence. Cependant, AODV hérite les inconvénients du routage réactif expliqués précédemment.

5.1.3. Les protocoles de routage hybrides

Les protocoles de routage hybrides sont basés sur le fonctionnement des deux modes de routage proactif et réactif étudiés précédemment. Cette fusion a pour objectif de proposer une nouvelle famille de protocoles regroupant les avantages des deux approches. Le routage hybride utilise la méthode proactive pour faire découvrir à chaque nœud X une « zone » limité à une certaine distance (en nombre de sauts). Le nœud X possède alors les routes vers tous les nœuds se trouvant au niveau de cette « zone » ; au-delà de cette zone prédéfinie, les protocoles hybrides font appel aux techniques de routages réactives pour chercher les routes vers des nœuds distants. Nous présentons dans ce qui suit le protocole ZRP [Haa97] qui est basé sur le principe de clustérisassions (ou découpage en zones) et le protocole AntHotNet [DCDG04] dont le fonctionnement est inspiré d'un algorithme de type colonie de fourmis.

5.1.3.1. Le protocole ZRP

ZRP (Zone Routing Protocol) [Haa97] est un protocole de routage hybride. Ce protocole est basé sur la notion de découpage du réseau en plusieurs zones. Chaque nœud définit autour de lui une zone de routage qui regroupe un ensemble de nœuds situés à une distance limitée en nombre de sauts par rapport à ce nœud. Les nœuds situés à la frontière d'une zone (c.-à-d. ayant exactement la distance limitée par la zone) sont appelés nœuds périphériques. Le routage des paquets dans ZRP est effectué par deux types de protocoles, le premier est un protocole proactif nommé IARP (Intrazone Routing Protocol), qui est employé uniquement à l'intérieur des zones, et le deuxième dit IERP (Interzone Routing Protocol), qui est un protocole réactif qui opère en dehors des zones pour la recherche des routes vers des destinations extérieures. La technique proactive utilisée dans ZRP est fondée sur l'approche du routage par vecteur distance et permet d'identifier tous les chemins possibles entre les nœuds dans une zone de routage alors que le protocole réactif quant à lui, permet de lancer des opérations de découverte de routes vers des destinations lointaines. De manière générale, la recherche des chemins dans ZRP s'effectue de la manière suivante :

– Lorsqu'un nœud souhaite transmettre des paquets, il vérifie si le nœud destination se trouve dans sa « zone ».

– Si le nœud destination se trouve dans sa table de routage, alors les paquets sont transmis.

– Sinon (le nœud destination est située en dehors de la « zone » du nœud source), il lance le protocole IERP. Ce dernier envoie une demande d'établissement de route « RouteRequest » à l'ensemble des nœuds périphériques de la zone.

– si un nœud périphérique a connaissance de la route demandée, alors le nœud demandeur recevra de ce nœud un paquet « RouteReplay » contenant le chemin menant à la destination. Dans le cas contraire, le protocole se poursuit récursivement jusqu'à obtention d'une réponse.

La figure 5-7 illustre le fonctionnement du protocole ZRP avec une zone de routage limitée à deux sauts.

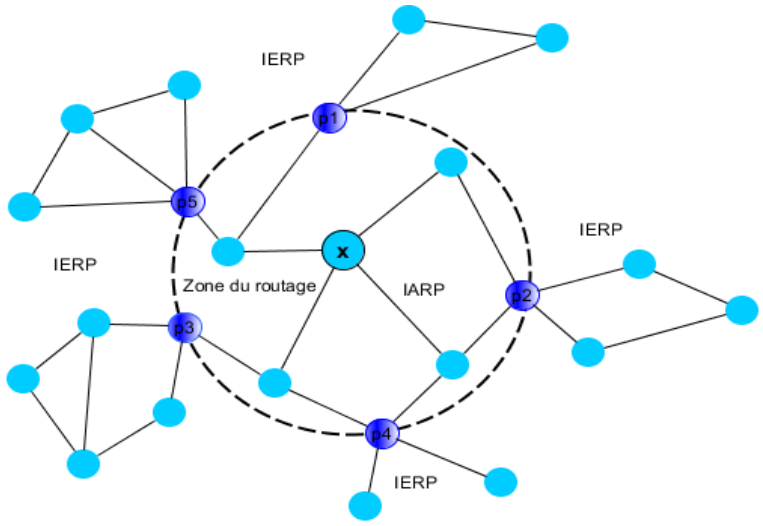


Figure 5-7- Le principe de fonctionnement de ZRP

ZRP possède un troisième protocole interne appelé BRP (Bordercast Routing Protocol) et dont le rôle est de définir les frontière des différentes « zones » et de construire des listes des nœuds périphériques aux différentes « zones » de routage.

L'avantage du protocole ZRP est qu'il réduit de manière significative le temps de transmission des paquets par rapport aux protocoles proactifs et réactifs purs. En effet, chaque nœud ne connaît que la topologie de sa propre « zone » et non pas la totalité du réseau comme le cas des protocoles proactifs. En outre, l'emploi des nœuds périphériques facilite et optimise les opérations de découverte de routes mieux que dans les protocoles réactifs puisque chaque nœud périphérique qui reçoit une requête de recherche réactive peut indiquer immédiatement si la destination est dans son voisinage ou non, et par conséquent savoir s'il faut aiguiller la dite requête vers les autres zones sans « perturber » les nœuds de sa propre zone. L'inconvénient principal du protocole ZRP réside d'une part au niveau du calcul difficile des distances qui limitent les zones, et d'autre part il doit supporter plusieurs types de protocoles internes et externes pour assurer les communications et l'envoi de paquets.

5.1.3.2. Protocoles hybrides basés sur les colonies de fourmis

5.1.3.2.1. Fonctionnement des colonies de fourmis

Les études fondées sur le comportement des insectes font apparaitre une nouvelle catégorie de protocoles de routage dans les réseaux ad hoc. En faisant une projection du comportement de ces insectes sur les caractéristiques des réseaux ad hoc, nous remarquons en effet que le comportement des fourmis est bien adapté aux caractéristiques des réseaux ad hoc à savoir le changement de topologie et le caractère autonome de ces réseaux. L'algorithme basé sur les colonies de fourmis est inspiré du comportement d'une colonie de fourmis réelles lorsque ces dernières cherchent leur nourriture. En se déplaçant de son nid vers une source de nourriture, la fourmi dépose durant son parcours une substance chimique odorante appelée phéromone ; cette opération permet ainsi une communication indirecte entre les fourmis et dont la finalité, après plusieurs aller/retour entre le nid et la nourriture, permet de renforcer collectivement la trace du plus court chemin entre le nid et la source de nourriture.

Pour mieux comprendre la dynamique du processus de recherche de nourriture chez les fourmis, nous allons détailler une expérience qui permet d'observer ces insectes de prêt. Le comportement des fourmis est observé sur un pont à double branches de longueurs différentes, ce pont est placé entre le nid d'une colonie de fourmis et une source de nourriture (voir figure 5-8). Les fourmis au départ se déplacent aléatoirement sur les deux branches du pont. Dès que les fourmis ont atteint la source de nourriture, elles empruntent le chemin inverse en déposant une certaine quantité de phéromone. Au fur et à mesure, elles renforcent la trace odorante existante le long des différents chemins. On note l'existence d'un phénomène d'évaporation qui permet de diminuer la concentration des phéromones ; ce phénomène permet justement aux fourmis de s'adapter aux changements de leur environnement (l'apparition d'un obstacle par exemple). C'est justement ce phénomène d'évaporation qui est repris dans les protocoles de routage basés sur le fonctionnement d'une colonie de fourmis pour gérer les changements de topologie dans les réseaux ad hoc. A chaque intersection de la figure 5-8, les fourmis choisissent une direction de façon aléatoire et proportionnelle à la quantité de phéromone déposées par les fourmis durant leur recherche. C'est ainsi qu'après plusieurs itérations (plusieurs aller/retour des fourmis de leur nid vers la nourriture), la quantité de phéromone va diminuer le long du chemin le plus long et augmenter le long du chemin le plus court et cela à cause du phénomène d'évaporation plus important sur le chemins le plus long, et de la concentration de la phéromone plus importante le long du plus court chemin : Les fourmis ont réussi à calculer le chemin le plus court de façon autonome et coopérative.

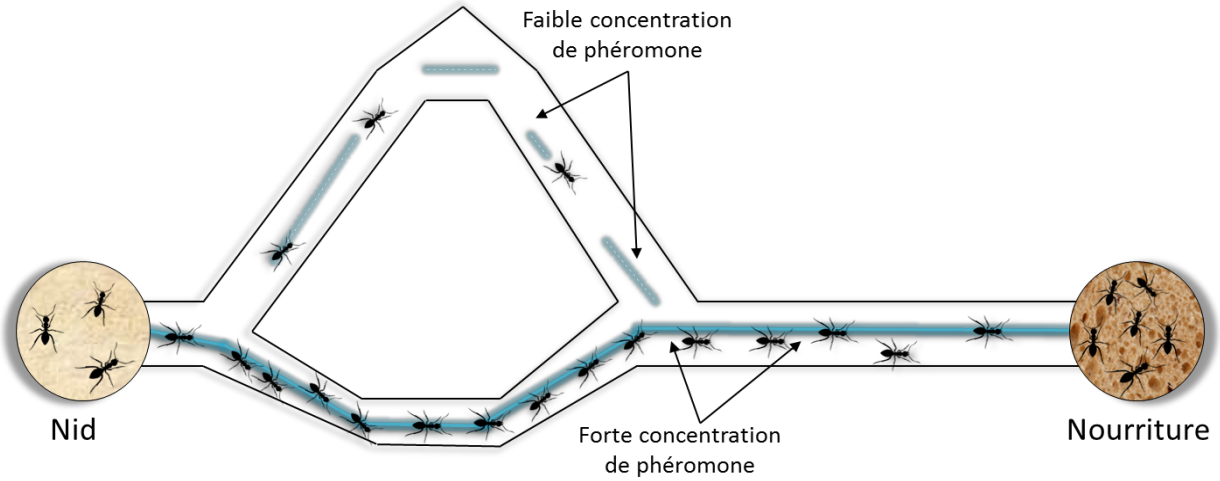


Figure 5-8- Le principe de fonctionnement d'une colonie de fourmis

5.1.3.2.2. Protocoles de routage basés sur les colonies de fourmis

Cette famille de protocoles de routage s'est inspirée du fonctionnement des colonies de fourmis. Elle s'appuie sur des algorithmes fondés sur la capacité de simples fourmis (agents) à résoudre des problèmes complexes par coopération. Les algorithmes basés sur les colonies de fourmis constituent un sous ensemble de ce qu'on appelle l'intelligence par essaim [BDT99]. L'idée principale de ces protocoles est de construire des chemins entre des nœuds source et destination en imitant les fourmis dans leur tâche de recherche de nourriture. Une caractéristique importante de ces protocoles est que les fourmis (agents) n'ont pas besoin de communiquer directement pour accomplir leur tâche, mais échangent de façon indirecte à travers la modification de leur environnement par dépôt de phéromones. De plus, il a été montré que ce type d'algorithmes s'adapte bien au caractère autonome des réseaux ad hoc. De plus, ces algorithmes ont la particularité de réagir rapidement face aux changements fréquents dans la topologie des réseaux ad hoc, ce qui facilite par conséquent la maintenance des routes. Un autre avantage de ces protocoles est qu'ils génèrent peu de trafic réseau (paquets de contrôle) comparés aux protocoles traditionnels tels que AODV ou OLSR.

Le point commun entre les différents protocoles de routage ad hoc basés sur le fonctionnement des colonies de fourmis est qu'ils utilisent tous une adaptation de l'heuristique ACO [DDC99] détaillée ci-dessous.

Le réseau ad hoc peut être modélisé à l'aide d'un graphe connexe G = (V, E) tel que |V| représente le nombre de nœuds du réseau du graphe qui représentent les nœuds du réseau ad hoc et E est l'ensemble des arcs du graphe qui symbolise les liens ou les connexions entre les nœuds. La méthode ACO est utilisée pour trouver le plus court chemin entre un nœud source v_s et un nœud destination v_d.

Chaque fourmi construit son chemin en se déplaçant à partir du sommet initial v_s, à travers un ensemble d'arcs du graphe G, en direction du nœud v_d. Lors de son déplacement, elle dépose une quantité de phéromone sur l'ensemble des arcs faisant partie de son chemin. Par conséquent tous les arcs de G possèdent une quantité de phéromones ($\forall e (i, j) \in E$ reliant v_i et v_j, e(i, j) <-- $\psi_{i,j}$) modifiée à chaque passage d'une fourmi ; cette opération a pour objectif de guider les autres fourmis lors de leurs prochains parcours.

Les fourmis choisissent le prochain sommet à atteindre suivant une règle de décision probabiliste en fonction des pistes de phéromone déposées sur les arcs. Une fourmi qui est arrivée au nœud v_i utilise la phéromone $\psi_{i,j}$ associée au nœud $v_j \in N_i$ (l'ensemble des voisins à un saut de v_i) pour calculer la probabilité de choisir v_j comme nœud suivant. La loi de probabilité utilisée par ACO est exprimée dans la formule suivante:

$$P_{i,j} = \frac{\psi_{i,j}}{\sum_{v_j \in N_i} \psi_{i,j}} \quad si\ v_j \in N_i\ ;\ 0\ sinon, \Sigma\ p_{i,j} = 1 \quad (1)$$

Dès qu'une fourmi quitte le nœud v_i pour aller vers le nœud v_j, elle dépose une quantité de phéromone sur l'arc e(i, j) ; cette modification est effectuée comme suit :

$$\psi_{i,j} \leftarrow \psi_{i,j} + \Delta\psi \quad (2)$$

Dans le cas des fourmis réelles la quantité de phéromone s'évapore avec le temps. L'algorithme ACO tient compte de ce phénomène d'évaporation. Après chaque itération d'algorithme, la quantité de phéromone sur les arcs diminue selon la règle suivante :

$$\psi_{i,j} \leftarrow (1 - q)\, \psi_{i,j} \quad q \in [0,1] \quad (3)$$

Il est difficile de mener des analyses théoriques précises sur les protocoles de routage reposant sur l'heuristique ACO, à cause des nombreux paramètres aléatoires qui dépendent les uns des autres et dont la distribution change à chaque itération. Cependant, ces protocoles donnent des résultats intéressants dans la pratique et en particulier en termes de réduction des coûts de communication et de réactivité aux changements de topologie. L'heuristique ACO représente donc une approche intéressante à étudier.

Il existe de nombreux protocoles de routage s'inspirant du fonctionnant des colonies de fourmis et qui utilisent l'heuristique ACO parmi lesquels nous pouvons citer les protocoles AntHocNet (An Adaptive Nature-Inspired Algorithm for Routing in Mobile ad hoc Networks)

[DCDG04] et ARA(The Ant-Colony Based Routing Algorithm for MANETs) [GSB02] [Bou02]. Dans ce qui suit nous donnons une présentation détaillée du protocole AntHocNet.

5.1.3.2.3. Le protocole AntHocNet

AntHocNet (An Adaptive Nature-Inspired Algorithm for Routing in Mobile ad hoc Networks) [DCDG04] est un protocole de routage hybride pour réseaux ad hoc, il est apparenté aux algorithmes ACOs et fortement inspiré du protocole AntNet [DD98]. La particularité de ce protocole est qu'il ne maintient pas les routes entre tous les nœuds du réseau à tout instant comme dans le cas du protocole AntNet, mais il maintient seulement les routes vers un ensemble de stations quand une demande d'échange de données est initiée. Afin de comprendre la manière avec laquelle l'heuristique ACO est employée par AntHocNet, nous allons détailler le fonctionnement de ce protocole selon trois phases principales :

La découverte de routes : Durant cette phase plusieurs chemins sont créés et cela requiert l'utilisation de fourmis réactives. Quand un nœud souhaite initier une communication avec un nœud destination, et qu'il ne procède pas de route vers cette destination dans sa table de routage, il crée une « fourmi réactive aller » et la diffuse vers tous ses voisins à un saut. La fourmi se déplace dans le réseau et cherche au niveau de chacun des nœuds traversés la destination demandée. Si un nœud courant dispose dans sa table de routage une entrée correspondante à sa requête, la fourmi sera dirigée vers le prochain saut menant à la destination. Sinon, elle sera clonée et diffusée vers tous les voisins de ce nœud. Un ensemble de fourmis sont dites de même génération si elles sont clonées à partir de la même fourmi. Elles sont caractérisées par un nombre de sauts limité pour éviter les recherches inutiles. Chaque nœud qui reçoit plusieurs fourmis de même génération dans une même période, interprète la distance parcourue par chaque fourmi et permet ensuite à une seule fourmi ayant parcouru le plus court chemin de continuer sa recherche et détruira les autres. En accord avec le mécanisme des algorithmes ACOs, les fourmis se déplacent d'un nœud vers un autre de manière aléatoire et proportionnelle aux pistes de phéromone locales des nœuds. La probabilité de choisir le voisin v parmi plusieurs voisins du nœud courant i pour atteindre la destination d vaut :

$$P_{vd} = \frac{\left(\varphi_{vd}^{i}\right)^{\beta 1}}{\sum_{j \in N_{jd}^{i}} \left(\varphi_{jd}^{i}\right)^{\beta 1}}$$

φ_{vd}^{i} *: La quantité de phéromones sur le lien (i ,v) concernant la destination d.*

N_{jd}^{i} *: L'ensemble des voisins de i à travers lesquels un chemin vers d est connu.*

β_1 **:** *Une constante supérieure à 1 qui modifie l'importance des pistes de phéromone.*

Dès qu'une fourmi trouve le chemin vers la destination d, celle-ci emprunte le chemin inverse (en marquant les tables de routage locales) afin de transporter la route trouvée à son nœud créateur. Cette méthode permet à un nœud demandeur de route d'avoir un ou plusieurs chemins vers la destination demandée. Ce nœud peut ensuite lancer le processus de routage des paquets. Les données envoyées utilisent le même mécanisme aléatoire que les fourmis pour choisir les prochains nœuds. La formule utilisée est identique à l'équation précédente sauf que la constante β_1 est remplacée par une constante β_2 tel que : $\beta_2 > \beta_1$ de sorte à réduire la recherche (exploration) au profit de l'exploitation.

Le maintien des routes : Lors du transfert des données, le nœud envoi en parallèle des fourmis ayant un comportement proactif ; le rôle de ce type de fourmis est de maintenir les routes en mettant à jour les informations contenues dans les tables de routage des nœuds traversées. A la création des fourmis proactives, ces dernières sont par défaut dans leur phase aller et se propagent d'un nœud vers un autre en suivant la même loi de probabilité basée sur les pistes de phéromones. Une phase retour est déclenchée par les fourmis proactives dès que celles-ci atteignent le nœud destination, Durant cette phase, les fourmis se chargent de la mise à jour des informations relatives aux chemins tels que le temps de parcours.

Dans le but de renforcer l'exploration des chemins dans le réseau, et apporter des variations et des améliorations autour des routes établies, il est possible qu'une fourmi « proactive aller » soit clonée et diffusée dans le voisinage des nœuds appartenant à son parcours. Notons que cette procédure est limitée (deux fois dans les expériences) et peu probable (sa probabilité est égale à 0.1).

Le protocole AntHocNet met en place un mécanisme de maintien de voisinage entre les nœuds. Durant la phase aller, ce mécanisme aide les fourmis proactives dans leurs déplacements. Pour se faire, chaque nœud diffuse périodiquement dans son voisinage son propre identifiant ; cette opération permet d'insérer les nouveaux nœuds dans les tables de routage des voisins.

Traitement des ruptures de liens : La dernière phase est le traitement des ruptures de liens détectées en l'absence d'informations reçues par l'un des voisins. Dès qu'un nœud quitte le réseau, chaque nœud voisin cherche s'il dispose d'une route alternative qui mène vers toute destination que pouvait atteindre le nœud ayant quitté le réseau. Si la route existe, il met à jour sa table de routage et informe ses voisins par diffusion de la modification, dans le cas contraire il crée une fourmi réactive qui se charge de la recherche de la destination. Si la fourmi a atteint son objectif, le nœud met à jours sa table de routage puis diffuse aux autres voisins la nouvelle route trouvée sinon l'entrée pour cette destination sera supprimée et toute requête de transfert de données vers le nœud destination va retourner un message d'erreur à son émetteur.

5.2. Notre proposition : Un nouveau protocole de routage hybride pour réseaux ad hoc

Nous avons présenté dans le chapitre précédent plusieurs architectures de réseau ad hoc en faisant ressortir leurs caractéristiques communes à savoir le manque de ressources matérielles, la mobilité et le caractère autonome et spontané de ce type de réseaux. Ces caractéristiques engendrent de nouvelles problématiques qui n'existaient pas dans les réseaux filaires traditionnels. Une des problématiques est le routage des données. Nous avons présenté dans ce chapitre l'essentiel des protocoles de routage utilisés dans les réseaux ad hoc. Nous pouvons généralement distinguer trois catégories de protocoles de routage : les protocoles utilisant une approche réactive, ceux utilisant une approche proactive et ceux qui utilisent aux mieux les avantages du réactif et du proactif : les protocoles hybrides. Quel que soit l'approche utilisée pour le routage des données, les différents protocoles de routage souffrent de problèmes récurrents aux communications dans les réseaux ad hoc. Le premier problème est celui du surcoût de communication causé par l'activité du protocole. Ce surcoût est essentiellement dû à la technique de broadcast utilisée lors de la diffusion des paquets de contrôle ou aux échanges périodiques des tables de routage. Le deuxième problème est que l'introduction de nombreux paquets de contrôle lors du calcul de chemins, peut entrainer une congestion du réseau ce qui peut causer une diminution des performances du réseau qui peut se dégrader en partie ou en totalité. L'augmentation du trafic peut faire perdre à certains nœuds leur pouvoir de transmission et certains paquets peuvent être ainsi perdus.

Afin d'apporter une réponse partielle à tous ces problèmes, nous proposons un nouveau protocole de routage pour les réseaux ad hoc. Ce protocole est hybride dans le sens où il combine une construction périodique de routes et une construction de route à la demande. L'aspect proactif de notre protocole est assuré par des agents moins volumineux que les tables de routages des protocoles classiques. Le protocole est construit à la façon d'un système Multi-Agent mobile qui fonctionne sur le principe d'un algorithme basé sur une colonie de fourmi mais qui diffère totalement du principe de l'algorithme ACO défini précédemment. Chaque nœud du réseau crée un certain nombre d'agents (fourmis) qu'il met à la disposition du réseau. Chaque agent permet, d'une part, une construction stochastique de routes vers le nœud qui l'a généré (aspect proactif du protocole) et, d'autre part, de collaborer avec les autres nœuds durant leurs processus respectifs d'établissement de routes (état réactif de l'agent). Grâce à l'état proactif des agents, une partie importante des routes sont construites. L'idée est d'utiliser cette quantité d'informations afin de construire, au besoin, les routes restantes. Une première idée serait d'utiliser la technique de diffusion comme c'est le cas dans les protocoles réactifs. Mais vu le nombre important des routes déjà construites durant la phase proactive, l'exploration totale de tout le réseau est une solution qui peut s'avérer non nécessaire. L'idée serait de faire une « exploration intelligente » du réseau en s'appuyant sur le principe de fonctionnement d'une

colonie de fourmis. Au lieu de diffuser la demande de route sur tout le réseau, comme c'est le cas dans tous les protocoles de routage hybrides basés sur les colonies de fourmis, la demande se fera localement au niveau de chaque nœud demandeur ; ce sont les agents transitant par ce nœud durant leur phase proactive qui se chargeront alors de diffuser cette demande de route dans le réseau. De cette façon, nous proposons un nouveau schéma de découverte de route moins couteux, qui s'adapte au caractère dynamique des réseaux ad hoc et à l'activité du réseau tout en étant plus efficace dans la pratique que le broadcast. Notre protocole ne permet pas d'établir systématiquement les routes les plus courtes en nombre de sauts par contre il permet à chaque nœud d'avoir plusieurs routes vers une même destination. Ceci a plusieurs avantages comme une meilleure robustesse face aux changements de topologie et la possibilité de choisir une route optimale par rapport à différents paramètres comme la qualité des liens ou le degré de fiabilité des nœuds intermédiaires (confiance ou réputation des nœuds).

Afin de valider notre protocole et d'évaluer ses performances, nous l'avons comparé à d'autres protocoles de routages (AODV, DSDV et AntHotNet) en tenant compte du changement de topologie (mobilité des nœuds), du nombre de paquets perdus, des délais de communication et enfin du nombre et de la taille des paquets de contrôle.

Notre protocole est constitué de trois principaux modules :

Le premier comprend la découverte de voisinage entre les nœuds mobiles. Cette tâche est primordiale dans un réseau caractérisé par une haute mobilité des nœuds et permet à chaque nœud de connaître l'identité de ses voisins à un saut.

Le second module gère la découverte des routes ; il permet d'établir des connexions entres des nœuds sources et des nœuds destinations. Cette opération est effectuée selon deux modes : le mode proactif et le mode réactif.

Le dernier module implémenté dans notre protocole prend en charge la gestion des ruptures de liens détectés entre les nœuds (suite à la mobilité ou à la disparition des nœuds du réseau). Une illustration de l'idée principale du protocole est donnée dans la figure 5-9. Nous décrivons dans ce qui suit le fonctionnement détaillé de notre protocole.

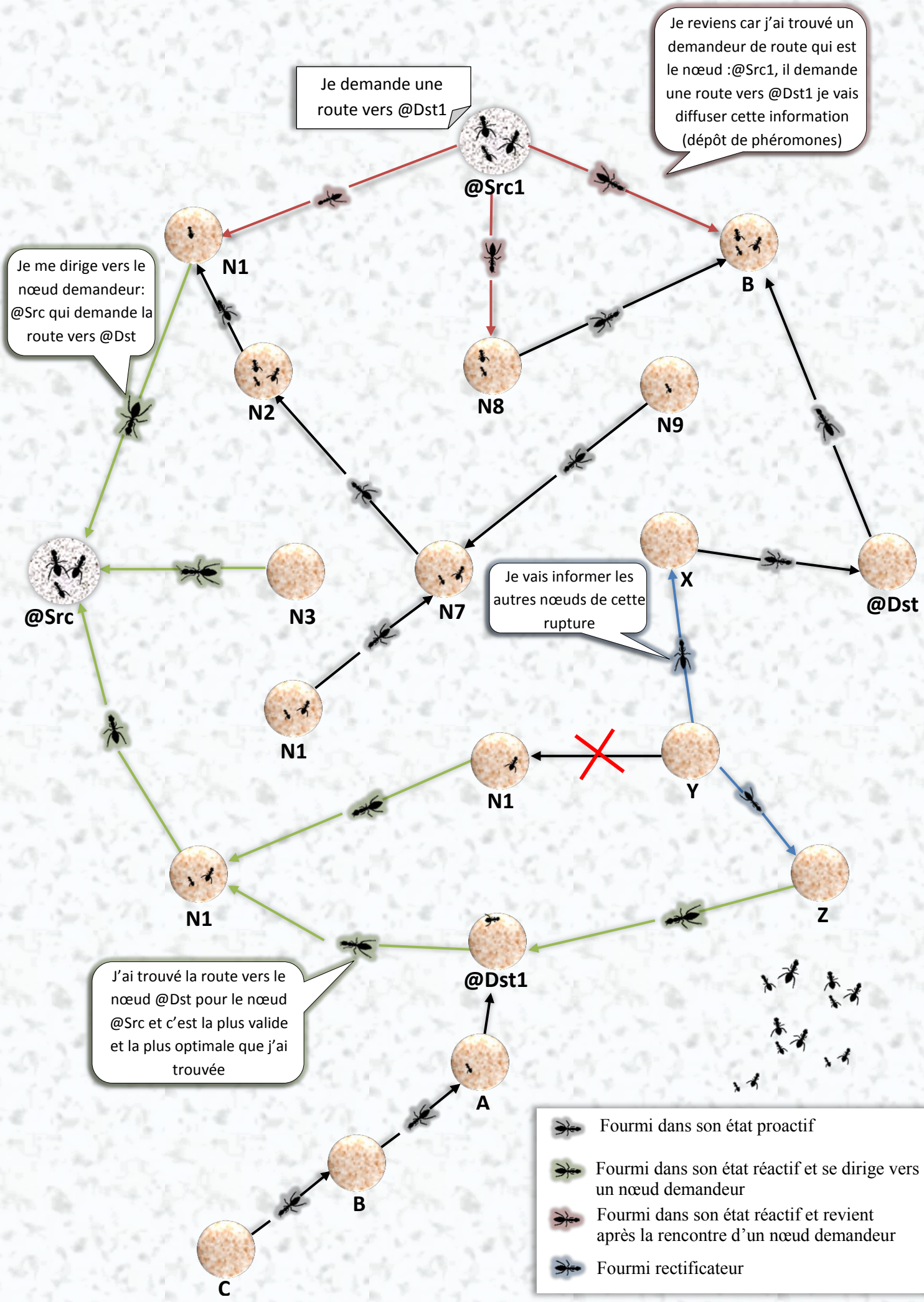


Figure 5-9- L'idée de base de notre protocole

5.2.1. La découverte de voisinage

Notre protocole effectue la découverte de voisinage et maintient les relations entre les voisins en utilisant des messages appelés hello. Chaque nœud diffuse périodiquement ses paquets hello aux nœuds situés dans sa portée radio. Les messages hello caractérisent aussi le type de lien entre les voisins puisqu'ils permettent d'identifier l'ensemble de voisins à un saut ayant un lien bidirectionnel avec d'autres nœuds. Cela s'effectue de la façon suivante: Si deux voisins diffusent leurs messages hello et que chacun d'entre eux figure dans la liste de l'autre (liste envoyée dans le message hello), cela confirme que le lien entre eux est bidirectionnel. Dans le cadre de notre protocole, nous faisons l'hypothèse que les liens entre les nœuds du réseau sont tous bidirectionnels.

Chaque nœud du réseau maintient localement une structure de donnée dite table de voisinage, en sauvegardant les informations récupérées à travers les messages hello circulant entre les nœuds. Cette table regroupe l'ensemble des voisins à un saut ainsi que quelques informations qui les concernent. Chaque entrée de la table de voisinage implémentée dans notre protocole est définie par: *<Adresse du voisin><Etat du lien><Date expiration de l'entrée>* ; où *<Adresse du voisin>* représente l'adresse du voisin à un saut, *<Etat du lien>* indique si le lien avec un voisin directe est bidirectionnel ou non et *<Date expiration de l'entrée>* présente la durée de validité de l'entrée.

5.2.2. La découverte de routes

Notre protocole effectue une découverte de route selon un modèle inspiré des colonies de fourmis. Les routes sont établies grâce à un ensemble d'agents que nous appellerons *Ant*. Ces agents *Ant* se déplacent en fonction des pistes de phéromones déposées au niveau des nœuds traversés. L'opération de découverte de routes comprend deux phases différentes qui reflètent le comportement des agents : une phase proactive et une phase réactive.

5.2.2.1. La phase proactive

Pour établir et maintenir les routes au sein du réseau, chaque nœud crée périodiquement un agent *Ant* et lui affecte deux paramètres différents: un identifiant unique *<ID_Ant>* et un numéro de séquence. L'identifiant est incrémenté à chaque création d'un nouvel agent et sert à éviter la formation des boucles de routage. En effet, si un nœud reçoit plusieurs agents avec un même identifiant alors il tient compte des informations délivrées par le premier agent et ignore les autres. Le numéro de séquence permet aux nœuds qui reçoivent plusieurs agents, de sélectionner les informations les plus récentes à sauvegarder pour mettre à jour leurs tables de routage (ceci sera détaillé par la suite). L'agent appartient à un seul nœud qui est son créateur appelé « Nœud Origine ». Durant son cycle de

vie, l'agent *Ant* proactif passe par deux phases différentes : une phase aller et une phase retour (voir figures 5-10 et 5-11). Lors de sa création, l'agent *Ant* est par défaut dans sa phase aller ; il se déplace aléatoirement d'un nœud vers un autre. Au fur et à mesure de son déplacement, il établit une route vers son nœud d'origine. Afin de contrôler le nombre d'agents dans le réseau, suivre leurs activités et surtout ne pas surcharger la bande passante, chaque agent Ant est doté d'une valeur qui limite sa durées de vie; cette valeur est exprimée en nombre de sauts maximum que l'agent doit traverser appelée TTL (Time To Live). La valeur TTL est proportionnelle à la taille du réseau pour mieux contrôler les déplacements des agents et mieux explorer les différentes zones autour des nœuds d'origines et garder un nombre d'agents dans le réseau approximativement constant.

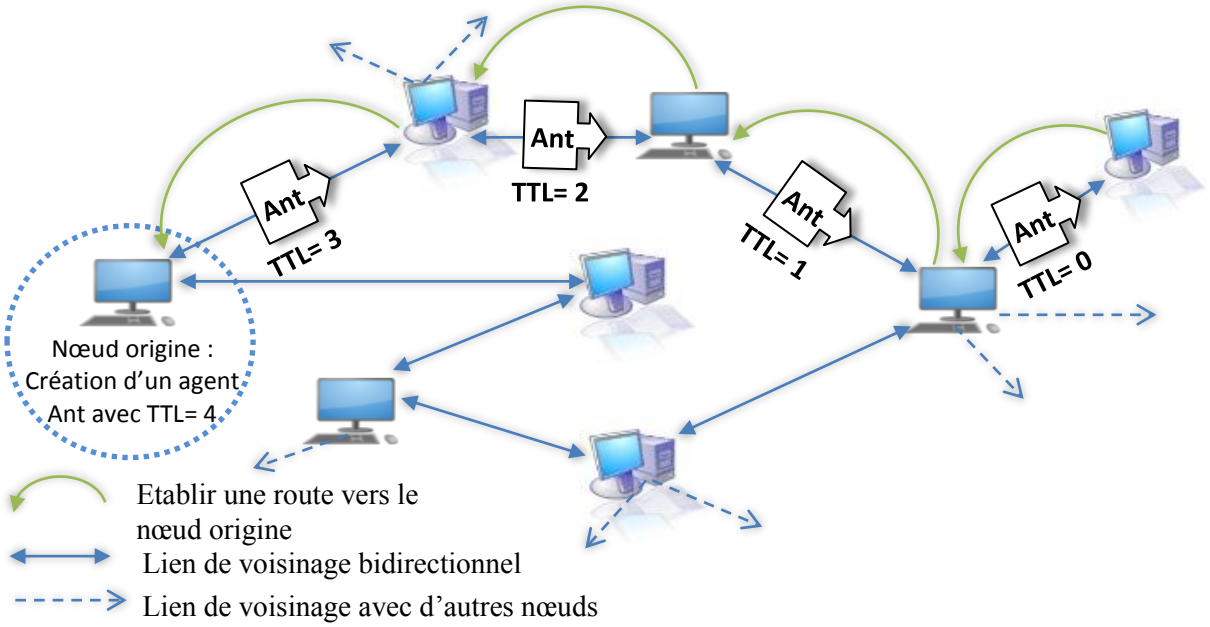


Figure 5-10- La phase aller d'un agent *Ant* proactif

La phase retour de l'agent est déclenchée dès que la valeur de son TTL vaut 0. L'agent emprunte alors le chemin inverse construit durant la phase aller, et construit une route à partir des nœuds traversés vers le dernier nœud atteint durant la phase aller.

Les routes découvertes par les agents pendant leurs phases aller et retour ne sont pas nécessairement les meilleures routes dans le réseau. L'objectif de notre protocole est de construire une multitude de routes pour une même destination. Cela permet de remédier aux changements de topologiques du réseau. Ainsi, l'existence de plusieurs chemins vers une même destination, permet aux participants de choisir certains chemins par rapport aux autres en se basant sur certains critères (nombre de sauts, confiance attribuée aux voisins ou toute autre métrique).

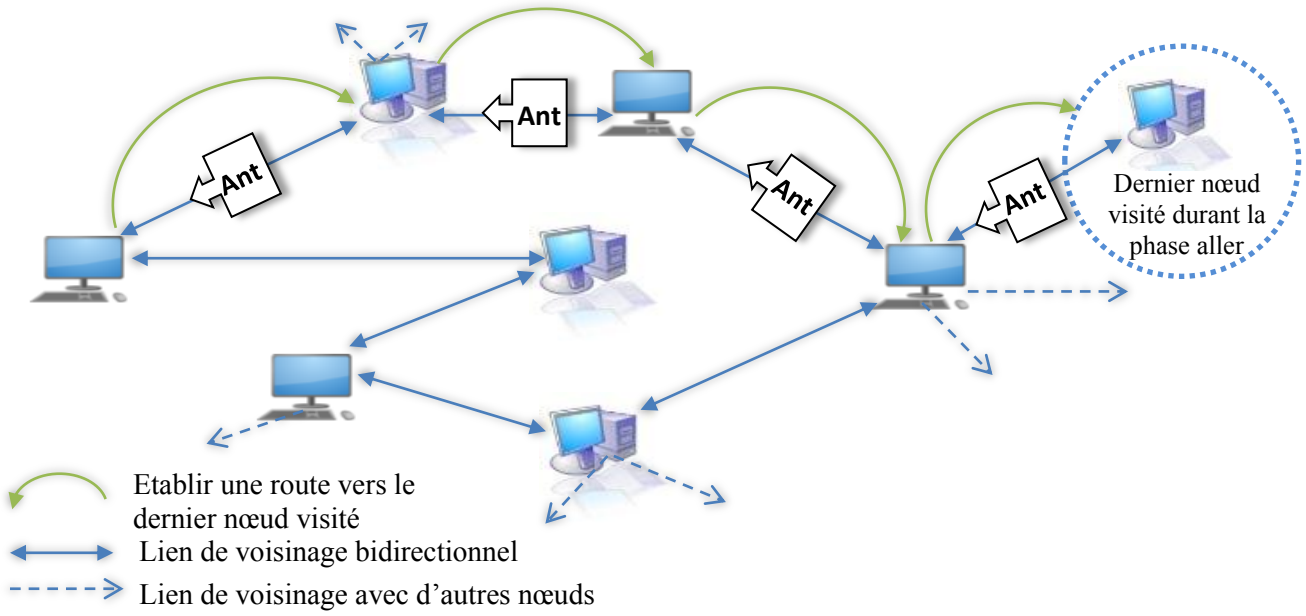


Figure 5-11- La phase retour d'un agent *Ant* proactif

Comme dans le cas des autres protocoles de routage, notre protocole maintient au niveau de chaque nœud une table de routage permettant de sauvegarder les routes établies. Pendant leur cycle de vie, les agents *Ant*, qu'ils soient en mode proactifs ou réactifs, effectuent des mises à jours des entrées des tables de routage des nœuds traversés pour maintenir les chemin empruntés lors des phases aller et retour. Chaque entrée d'une table de routage est définie par les champs <*dest*> <*n-suiv*><*d*><*Listv*> <*SeqNumber*><*E*><*Stat*> où <*dest*> est le nœud destination, <*n-suiv*> est le prochain nœud pour atteindre la destination, <*d*> est la distance qui sépare le nœud source de sa destination, c'est une métrique d'évaluation de la longueur des routes, <*Listv*> est la liste des voisins qui utilisent la route vers la destination, <*SeqNumber*> représente le numéro de séquence de la route, <*E*> indique la date d'expiration d'une entrée et <*Stat*> est une information sur la validité de la route et qui sert à indiquer l'état des liens entre les voisins.

5.2.2.2. La phase réactive

La phase réactive consiste à l'établissement des routes à la demande. Lorsqu'un nœud veut relayer des paquets de données vers une destination dont la route n'est pas établie durant la phase proactive, il initie alors une opération de découverte de route dans le réseau. L'approche réactive utilisée par les protocoles existants est basée sur une technique de diffusion traditionnelle qui consiste à inonder le réseau par des paquets de contrôles afin d'établir un chemin. Le nombre de messages de contrôle utilisés dans cette approche augmentent de façon exponentielle avec le temps, ce qui

provoque une saturation au niveau de la bande passante, et engendre un surcout de communication surtout dans les réseaux denses. Ce type de diffusions influence négativement sur les ressources énergétiques des nœuds mobiles. Dans le cas de notre protocole, nous avons pensé à une nouvelle stratégie plus originale pour la découverte de routes. L'idée proposée consiste à sauvegarder les demandes de routes au niveau des nœuds demandeurs (stockage local), et c'est aux agents *Ant* circulant dans le réseau de collaborer pour propager intelligemment la demande au sein du réseau et par conséquent trouver des chemins vers la destination recherchée. Pour cela, chaque nœud N maintient localement une structure de données appelée « table de demandes de routes », mise à jour de façon périodique et dont le but est de sauvegarder localement les demandes de route du nœud N.

Les demandes de routes sont prises en compte par les agents *Ant* durant leur phase proactive. Pendant leur phase aller, les agents Ant peuvent passer vers un état réactif. Le changement d'état est effectué automatiquement dès qu'un agent traverse un nœud qui fait localement une demande de route ; l'agent bascule alors dans sa phase retour vers son nœud d'origine (même si son TTL ne vaut pas 0) et informe les nœuds qu'il traverse de cette demande de route. La technique utilisée pour informer les nœuds intermédiaires de cette demande de route est fortement inspirée du fonctionnement des colonies de fourmis. En effet, chaque agent, durant sa phase retour, dépose une quantité de phéromone au niveau des nœuds du chemin vers son nœud origine. Ce dépôt de phéromones va contribuer à marquer les routes vers les nœuds demandeurs, ce qui permettra d'attirer plus d'agents qui vont de manière coopérative aider à l'établissement des routes demandées. Les quantités de phéromones déposées par les agents sont enregistrées dans une table de phéromone maintenue au niveau de chacun des nœuds du réseau. Chaque entrée de cette table est définie par :

<*Demandeur*><*Destination demandée*><*Liste des voisins*><*Quantité de phéromone*> ; où <*Demandeur*> est l'adresse du nœud demandeur, <*Destination demandée*> est l'adresse du nœud destination que le demandeur souhaite atteindre, <*Liste des voisins*> est la liste des voisins qui mènent vers le nœud demandeur et <*Quantité de phéromone*> est la quantité de phéromone relative à cette demande.

La table de phéromone est mise à jour à chaque passage d'un agent qui transporte une demande de route. La quantité de phéromone déposée par chaque Agent est définie par l'équation (1) suivante:

$$Q_{it} = Q_{i(t-1)} + q \qquad (1)$$

Où, Q_{it} est le niveau de phéromones dans le nœud n_i au temps t, et q est une constante positive.

Cette diffusion de phéromones dans le réseau indique l'état de demandes de routes initiées par certains nœuds du réseau à un moment donné (voir figure 5-12). Durant la phase proactive et afin de satisfaire globalement toutes ces demandes de routes, les agents se déplacent dans le réseau de manière stochastique et proportionnellement à la quantité de phéromones se trouvant sur chaque nœud du réseau. En effet, chaque nœud, durant sa phase aller, choisit le nœud suivant à visiter parmi ses voisins de façon aléatoire et en donnant plus de « chance » aux voisins menant vers des nœuds demandeurs de routes. Ce processus augmente les chances de choisir une direction vers un nœud demandeur tout en ne négligeant pas d'autres directions.

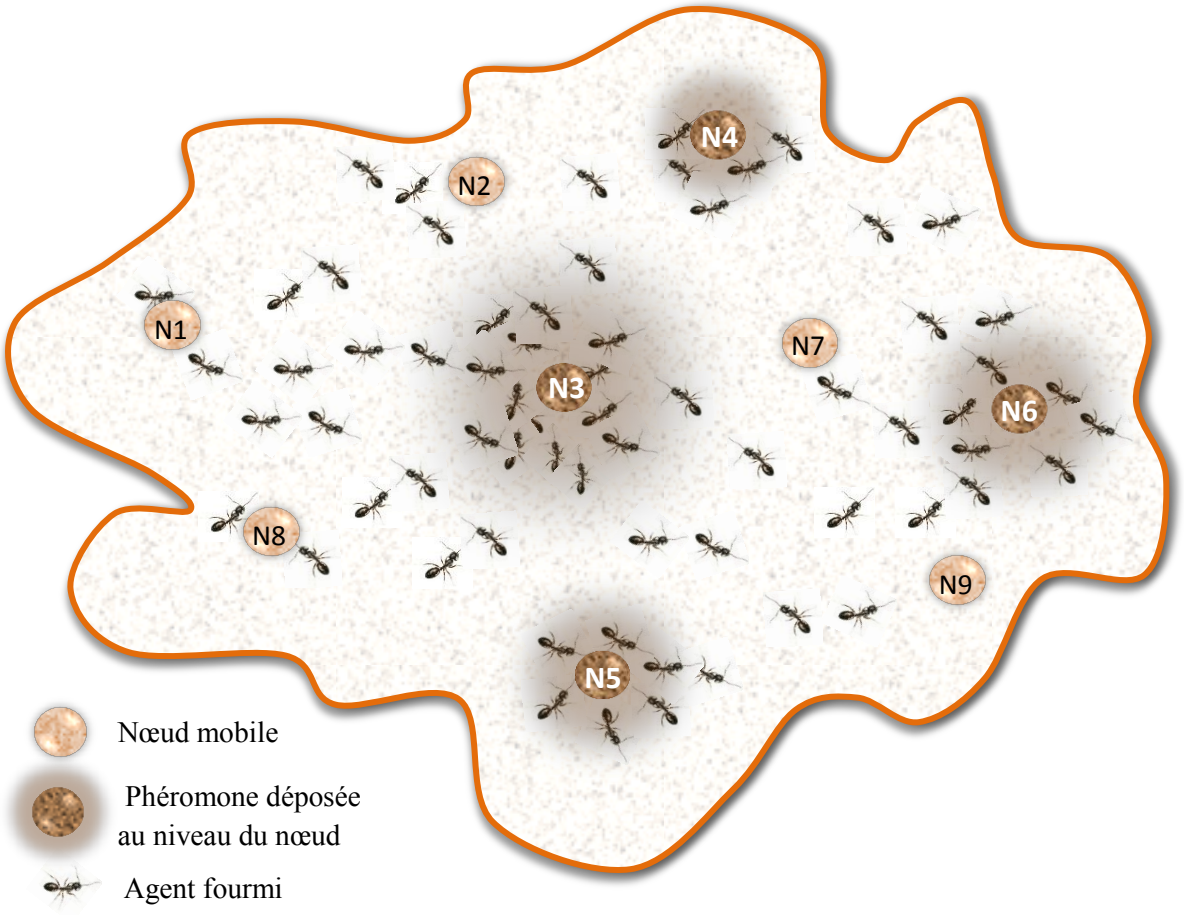


Figure 5-12- Influence des phéromones sur le déplacement des agents *Ant*

L'expression qui calcule la probabilité de choisir le nœud n_j comme prochain saut qui mène vers la destination d parmi plusieurs voisins du nœud n_i est définie comme suit :

$$p\,(n_j, d) = \frac{Q_{(n_j,d)}}{\sum_{n_k \in V(n_i)} Q_{(n_k,d)}} \qquad (2)$$

Avec : $V\,(n_i) = \{n_j \text{ voisin de } n_i\}$ et : $Q\,(n_k, d)$ correspond à la quantité de phéromone associée au voisin n_k de n_i pour la demande de route d.

Cette approche augmente d'une part les chances d'avoir un agent *Ant* provenant de la destination demandée, et d'autre part, permet d'attirer rapidement les agents qui ont traversé des nœuds qui ont déjà établi des chemins vers cette même destination.

Lorsqu'un agent visite un nœud *X* qui indique dans sa table de phéromones une demande de route d'un nœud *Y* vers une destination *d*, l'agent vérifie alors si le nœud X possède une route vers la destination d. Si cette route existe, il étend alors cette route jusqu'au nœud demandeur ce qui lui permet de satisfaire sa demande route. Dans le cas contraire, il continue son déplacement de façon déterministe vers le nœud demandeur de route. Il effectue la même vérification sur chaque nœud intermédiaire traversé. Dès que l'agent arrive vers le nœud demandeur de route deux cas se présentent : si la route vers la destination d a été trouvée sur un nœud intermédiaire il dépose les informations concernant cette route et retourne à son nœud d'origine en empruntant le chemin inverse (phase retour), sinon il vérifie si la demande de route est toujours en cours, et si c'est le cas, il retourne à son nœud d'origine en incrémentant la quantité de phéromones associées au nœud demandeur le long du chemin inverse (phase retour) sinon, il se dirige seulement vers son nœud origine sans renforcer la trace de phéromone. La figure 5-13 montre un exemple type du rôle d'un agent *Ant* réactif.

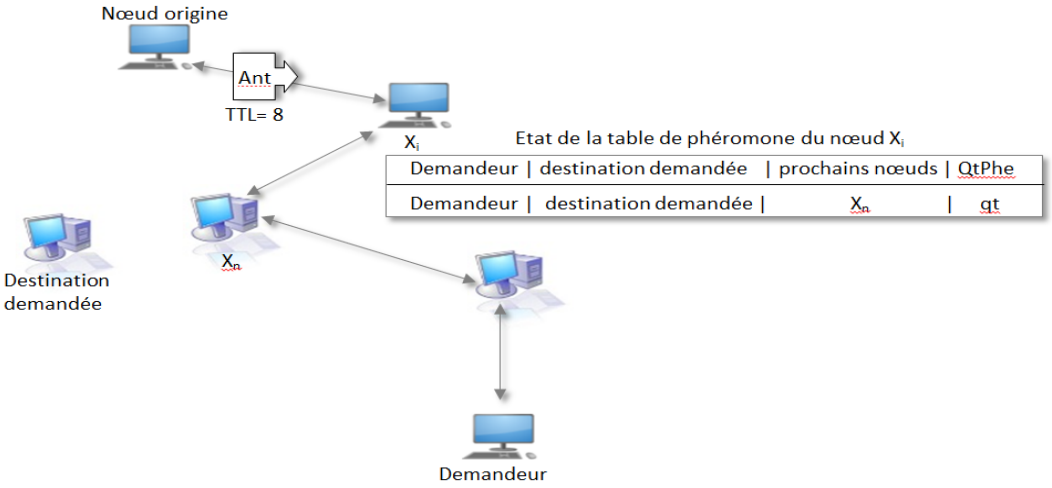


Figure 5-13- Rôle d'un agent *Ant* réactif

Afin de permettre aux demandes de routes d'être prises en compte de manières équitable en tenant compte des changements de topologie du réseau, un processus d'évaporation est mis en place. A chaque intervalle de temps, la quantité de phéromones correspondant à chaque demande de route est décrémentée suivant la formule de l'équation (3) :

$$Q_{it} = (1-\alpha) * Q_{i(t-1)} \qquad (3)$$

Ou, Q_{it} est la quantité de phéromone liée à un nœud demandeur, stockée dans le nœud n_i au temps t, et α est un réel ($0 < \alpha < 1$).

Comme nous l'avons déjà présenté, le déplacement des agents dépend fortement de la quantité globale de phéromones diffusée dans le réseau. Cette quantité de phéromones indique l'état des demandes globales de routes dans le réseau. Ceci a pour conséquence un déplacement intelligent des agents dans le réseau suivant les besoins dans chaque zone. Cette propriété est très intéressante car elle permet au protocole de s'adapter à l'activité des nœuds et de ce fait exploiter au mieux les capacités des agents.

Rappelons que le protocole ne permet pas d'établir systématiquement les routes les plus courtes, par contre il permet à chaque nœud d'avoir plusieurs routes vers une même destination et donc de pouvoir choisir la meilleure route parmi celles trouvées. Lorsqu'un agent arrive sur un nœud, il met à jour les informations de routage locales puis rafraichit en même temps ses propres informations de routage par l'intermédiaire du nœud visité. La sélection d'une route optimale parmi celles découvertes, se fait, tout d'abord sur la base du plus grand numéro de séquence qui correspond aux routes les plus récentes et ensuite sur la base de la plus courte distance en nombre de sauts.

5.2.2.3. Fonctionnement détaillé des agents *Ant*

Apres avoir présenté le comportement et le fonctionnement général des agents *Ant* pendant les deux phases proactive et réactive, nous présentons dans ce qui suit, l'ensemble des fonctionnalités allouées aux agents à travers quelques algorithmes. Avant de présenter ces algorithmes, nous donnons quelques notations et terminologie que nous utiliserons tout au long de cette partie ainsi qu'une description des champs transportés par les agents *Ant* et ceux qui interviennent lors des changements de topologies durant leur parcours.

5.2.2.3.1. Notations et terminologie

Soit $Route(a, b) = (a, i_1, i_2, ..., b)$ une route entre le nœud *a* et la destination *b* avec:
$(i_1, i_2, ..)$ représentent les nœuds intermédiaires.
Init (Route(a, b))=a.
Terminal (Route(a, b))=b.
$suiv (i_k) = i_{k+1}$ représente le nœud suivant du nœud i_k dans *Route(a, b).*

Soit deux routes $Route_i(a, b)$ et $Route_j(a, b)$:
$Route_i(a, b) != Route_j(a, b) => suiv(init(Route_i(a, b))) != suiv(init(Route_j(a, b)))$.

Routes(a, b) :	L'ensemble des routes reliant le nœud *a* au nœud *b*.
RouteOptimale(a, b):	La meilleure route d'un nœud *a* vers un nœud *b*.
$TABROUT_n[Dest, suiv]$:	L'ensemble des routes disponibles dans le nœud *n* pour la destination *« Dest»* qui ont comme prochain saut le nœud *« Suiv»*.
$TABV_n$:	La table de voisinage du nœud *n*.
$TABROUT_n$:	La table de routage du nœud *n*.
$TABPH_n$:	La table de phéromone du nœud *n*.

5.2.2.3.2. Les champs transportés par l'agent *Ant*

Dès leur création et pendant leur déplacement (état réactif ou proactif), les agents sont porteurs d'un ensemble de champs spécifiques. C'est grâce à ces informations échangées avec les nœuds visités que le protocole aura le comportement décrit précédemment. Ces informations sont décrites par les attributs suivants :

« *NoeudOrigine* » :	L'adresse du nœud créateur de l'agent.
« *Direction* » :	Variable qui fait la distinction entre la phase aller et la phase retour des agents.
« *seqNumber* » :	Un numéro de séquence incrémenté à chaque déplacement du nœud origine.
« *NoeudPreced* » :	L'adresse du nœud précédemment visité. Initialement, c'est l'adresse associée au nœud créateur de l'agent « ***NœudOrigine*** ».
« *TTL* » :	La durée de vie de l'agent. Sa valeur est décrémentée à chaque saut.
« *ListV* » :	La liste des nœuds déjà visités.
« *NoeudCible* » :	Ce champ sauvegarde l'adresse d'un nœud qui fait une demande de route.
« *DestDemand* » :	Une variable utilisée par l'agent pour sauvegarder la destination demandée par « *NoeudCible* ».
« *NoeudEtabRoute* » :	Contient l'adresse du nœud vers lequel l'agent est entrain d'établir une route. Il est initialisé à « *NoeudOrigine* ».
« valide » :	Une information booléenne initialisée à vrai et qui indique la validité de la route entre « *NoeudPreced* » et « *NoeudEtabRoute* ».
« *NS* » :	Le nombre de sauts pour atteindre « *NoeudEtabRoute* ». Il est initialisé à zéro.
« *NoeudSuiv* » :	Le prochain nœud que l'agent va visiter.
« *t* » :	Le temps écoulé pour se déplacer du « *NoeudEtabRoute* » vers le nœud courant.
« *ID* » :	L'identifiant de chaque agent. Il est incrémenté à chaque émission d'un nouvel agent.
« *Reserved* » :	Des champs réservés pour une utilisation ultérieure.

Remarque : afin d'apporter une solution au problème de formation des boucles de routage, nous avons pensé à combiner l'identifiant de chaque agent avec l'adresse de son nœud origine. De cette façon, les nœuds qui reçoivent deux fois de suite le même agent, ne tiennent comptent que des informations reçues lors du premier passage de l'agent pour mettre à jours leurs tables de routage.

5.2.2.3.3. Les algorithmes décrivant quelques comportements des agents Ant

Nous décrivons dans ce qui suit, les principales tâches allouées aux agents *Ant*, parmi lesquelles :

– La Mise à jour de la table de routage du nœud courant visité.
– L'actualisation des informations de l'agent.
– Le lancement de la phase retour.
– La mise à jour des tables de phéromone du nœud visité par l'agent.

Avant de détailler les algorithmes qui correspondent à ces activités, nous commençons d'abord par donner l'algorithme qui décrit la notion de route optimale telle qu'elle est implémentée dans notre protocole.

Le choix des routes optimales est effectué par chaque agent Ant selon l'algorithme suivant :

Algorithme 1 Le choix d'une route optimale

Soient *:*

- *n* le nœud courant
- *NoeudSuiv* le prochain nœud que l'agent *Ant* va visiter et lui ramener une route optimale vers la destination *NoeudEtabRoute* (route contenue dans le nœud courant *n*)
- $E = \{Route_k(n,NoeudEtabRoute) \in Routes(n,NoeudEtabRoute)$ tq *:* $TABROUT_n[NoeudEtabRout,suiv(init(Route_k(n,NoeudEtabRout)))].SeqNumber$ est le maximum$\}$

Si$(RouteOptimale(n,NoeudEtabRoute) \in E$ existe *et* $\forall\ Route_k(n,NoeudEtabRoute) \in E$, $suiv(init(Route_k(n,NoeudEtabRoute))) != NoeudSuiv)\{$

Ant.NS/Ant.t $= TABROUT_n[NoeudEtabRout, suiv(init(RouteOptimale(n,NoeudEtabRoute)))].d/t$

Ant.seqNumber $= TABROUT_n[NoeudEtabRoute,$ $suiv(init(RouteOptimale(n,NoeudEtabRoute)))].SeqNumber$

Ant.valide =vrai

$\}$ Sinon *Ant.valide =faux*

La procédure de mise à jour d'une table de routage effectuée par un agent *Ant* est décrite suivant l'algorithme suivant :

Algorithme 2 Mise à jour d'une table de routage d'un nœud

Soient *:*

- *Ant* un agent qui se déplace du nœud précèdent *n-1* pour mettre à jour la table de routage du nœud courant *n*
- *N* l'ensemble des nœuds du réseau
- *NoeudEtabRoute* l'adresse du nœud vers lequel l'agent est en train de créer une route

//**Vérifier la validité de la route transportée par l'agent** *Ant* et effectuer un test de connectivité entre les //**nœuds** *n-1* et *n* :

Si *(Ant.valide=vrai et Ant.NoeudPreced* $\in$ *$TABV_n$){*

*Ant.t =Ant.t+*temps estimé par l'agent pour aller du nœud *n-1*vers le nœud *n*

Ant.NS++

//Mise à jour de la table de routage

Si *$TABROUT_n$[Ant.NoeudEtabRoute, Ant.NoeudPreced]*n'existe pas*{*

//Créer une entrée dans la table de routage avec :

destination =Ant.NoeudEtabRoute

nœud suivant =Ant.NoeudPreced

distance =Ant.NS //ou bien *t =Ant.t*

numéro de séquence =Ant.seqNumber

ListV =NULL //**cette liste sera remplie initialement par le prochain nœud que l'agent** *Ant* //**va visiter puisque ce nœud va probablement utiliser cette route**

}

Sinon *{*

Si *[Ant.seqNumber>$TABROUT_n$[Ant.NoeudEtabRoute,Ant.NoeudPreced].SeqNumber ou (Ant.seqNumber=$TABROUT_n$[Ant.NoeudEtabRoute, Ant.NoeudPreced].SeqNumber et Ant.NS/Ant.t<$TABROUT_n$[Ant.NoeudEtabRoute, Ant.NoeudPreced].(d/t))] {*

//**Remplacer la table existante par les informations amenées par l'agent**

$TABROUT_n$[Ant.NoeudEtabRoute, Ant.NoeudPreced].(d/t) :=Ant.NS/Ant.t

$TABROUT_n$[Ant.NoeudEtabRoute, Ant.NoeudPreced].SeqNumber =Ant.seqNumber

//Envoyer un agent rectificateur vers tous

*//***nœud** *x* $\in$ *$TABROUT_n$[Ant.NoeudEtabRoute, Ant.NoeudPreced].ListV*

}

}

}

Dès qu'un agent effectue une mise à jour d'une table de routage, il s'occupe de la mise à jour de ses propres informations puis se déplace aléatoirement vers le prochain nœud à visiter. Les détails de cette étape sont décrits dans l'algorithme suivant :

Algorithme 3 Actualisations des informations de l'agent *Ant*

Soient *:*

- *Ant* un agent situé au niveau du nœud courant *n* et venant du nœud précèdent *n-1*
- *TTL* durée de vie de l'agent *Ant*

//Actualiser les paramètres

Ant.NoeudPreced = n

Ant.TTL- -

Si (l'agent *Ant* est dans sa phase aller) *Ant.listNoeudVisit+n*

Algorithme 4 **La phase retour d'un agent ant**

Soient *:*

- *Ant* un agent qui lance sa phase retour au niveau du nœud courant *n* et se dirige vers le nœud *n+1*
- *n.SeqNumber* est le numéro de séquence du nœud courant *n*
- *NoeudEtabRoute* l'adresse du nœud vers lequel l'agent est entrain de créer une route

//Initialisation des paramètres **de l'agent** *Ant* **au niveau du nœud** *n*

Ant.NoeudEtabRoute = n

Ant. NoeudPreced= n

Ant.NS =0

Ant.valide =vrai

Ant.seqNumber=n.SeqNumber

Ant.NoeudSuiv= Ant.ListV.suivant() //parcourir *ListV* et choisir le prochain saut, qui est *n+1* dans notre cas

Si (le nœud *n* est un demandeur de route)

ant.DestDem =destination demandée par le nœud courant

//**Taches à effectuer par l'agent** A*nt* **à son arrivée au nœud** *n+1*

Mettre à jour la table de routage du nœud *n+1*(voir algorithme 2)

Mettre à jour ses propres informations (voir algorithme 3)

Si *(Ant.DestDem !=null) {*

Mette à jour la table de phéromone du nœud *n+1* (voir algorithme 5)

Si *(n+1=Ant.NoeudOrigine)* fin du parcours de l'agent

*}*Sinon*{*

*Ant.NoeudSuiv=Ant.ListV.suivant() // **NoeudSuiv** $\in$ **TABV**$_n$*

Choisir *RouteOptimale(n+1,NoeudEtabRoute)* (voir algorithme 1)

Si *(Ant.valide=vrai)*

$TABROUT_{n+1}$*[Ant.NoeudEtabRoute, suiv(init(RouteOptimale(n+1 ,NoeudEtabRoute)))].ListV*

+ Ant.NoeudSuiv

}

Comme on l'a détaillé précédemment, l'agent *Ant* (dès sa création) est par défaut dans sa phase aller. La phase retour de l'agent est activée dès que le TTL de l'agent vaut 0 ou lorsque l'agent se trouve sur un nœud qui fait une demande locale de route vers une destination. La procédure du lancement de la phase retour d'un agent *Ant* est décrite suivant l'algorithme précédent (algorithme 4).

Une tâche importante des agents *Ant* réactifs (durant leurs phases retour) est la mise à jour des tables de phéromone. Comme on l'a défini précédemment, cette procédure est effectuée pour attirer le maximum d'agents vers un demandeur de route et augmenter ses chances d'avoir une réponse à sa requête. Le déroulement de cette tâche est décrit dans l'algorithme suivant :

Algorithme 5 **Mise à jour d'une table de phéromone**

Soient *:*

- *Ant* un agent réactif traversant le nœud *n* pendant sa phase retour
- $TABPH_n$ *[Demandeur, Destination demandée] :* la table de phéromone du nœud *n*
- *ListV* le champ *<Liste des voisins>* d'une entrée de la table $TABPH_n$
- *qtPhe* une quantité de phéromone associée à une entrée de la table de phéromone
- Δqt est la quantité de phéromone initiale à déposer dans une nouvelle entrée de la table de phéromone
- Ψqt la valeur de phéromone à ajouter par un agent A*nt*

//Mise à jour de la table de phéromone du nœud intermédiaire *n*

Si $TABPH_n$*[Ant.NoeudEtabRoute, Ant.DestDem]*existe*{*

$TABPH_n$*[Ant.NoeudEtabRoute, Ant.DestDem].qtPhe* + ψqt

Si *Ant.NoeudPreced* $\notin$ $TABPH_n$*[Ant.NoeudEtabRoute, Ant.DestDem].ListV*

$TABPH_n$*[Ant.NoeudEtabRoute, Ant.DestDem]. ListV + Ant.NoeudPreced*

*}*Sinon*{*

//Créer une nouvelle entrée dans la table de phéromone avec :

Demandeur =Ant.NoeudEtabRoute

Destination demandée =Ant.DestDemand

Liste des voisins = Ant.NoeudPreced

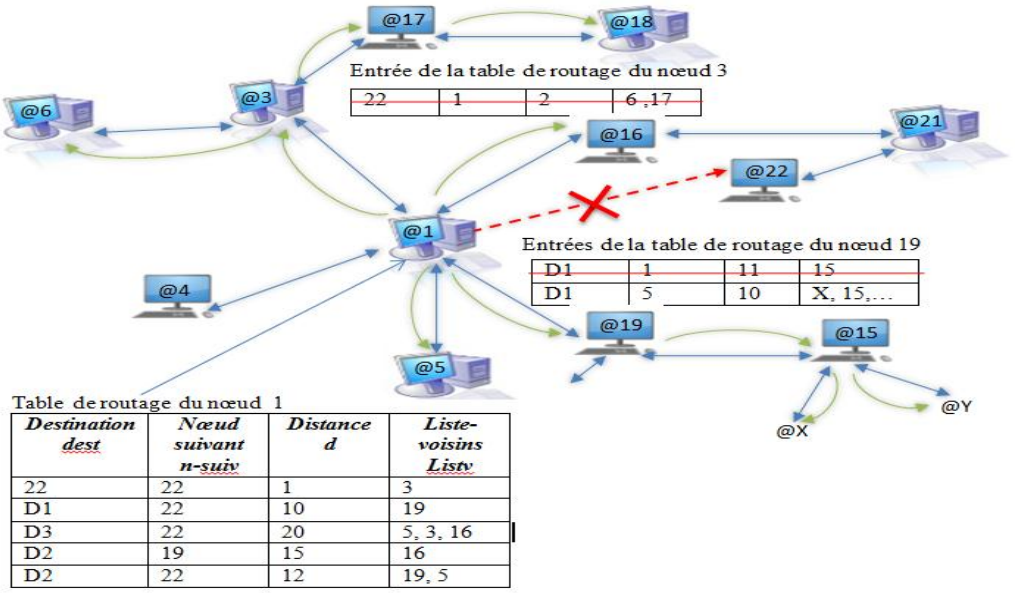
Quantité de phéromone =Δqt

}

5.2.3. La maintenance des routes

Les changements fréquents de la topologie des réseaux ad hoc provoquent souvent des ruptures de liens entre les nœuds mobiles. A cet effet, notre protocole fournit un mécanisme de maintenance des ruptures liens permettant d'assurer la robustesse du système proposé. La gestion des

ruptures de liens est effectuée au niveau de chaque nœud mobile grâce à des agents appelés rectificateurs ou « *RectifierAnt*». Ces agents spéciaux sont créés par chaque nœud dès qu'un lien avec un nœud voisin est rompu, et que ce voisin est sauvegardé dans la table de routage du nœud comme étant le prochain saut vers au moins une destination dans le réseau. Les agents rectificateurs sont envoyés vers tous les voisins sauvegardés dans le champ « Listv» des entrées correspondantes dans la table de routage du nœud afin d'actualiser les paramètres de routage de ses voisins. Chaque nœud mobile recevant un agent *RectifierAnt*, et après avoir mis à jour ses informations de routage, génère à son tour un agent de même type qu'il enverra aux voisins concernés par la route défaillante et ainsi de suite. Grace à cette diffusion intelligente, tous les nœuds concernés seront informés sans avoir divulgué l'information dans tout le réseau. Dans la figure 5-14, nous allons montrer un exemple de maintenance de lien entre deux nœuds dans un réseau ad hoc.



Destination dest	*Nœud suivant n-suiv*	*Distance d*	*Liste-voisins Listv*
22	22	1	3
D1	22	10	19
D3	22	20	5, 3, 16
D2	19	15	16
D2	22	12	19, 5

Agent rectificateur « RectifierAnt »
Lien bidirectionnel entre deux nœuds mobiles
Lien rompu

Figure 5-14- La maintenance de routes dans notre protocole

5.2.3.1. Fonctionnement détaillé de l'agent « *RectifierAnt* »

Les agents « *RectifierAnt* » ont un rôle très important dans le cas de notre protocole, ils permettent en effet d'assurer la stabilité et la robustesse du réseau en assurant la validité des liens entres les nœuds mobiles. Les agents rectificateurs sont appelés afin de remédier à des changements de topologie du réseau. A cet effet, ces agents interviennent dans les deux cas suivants :

– Défaillance de lien avec un nœud voisin.
– Modification des entrées dans les tables de routage.

Dans le premier cas, la défaillance peut être repérée selon plusieurs méthodes. Notre protocole s'appuie en premier lieu sur le mécanisme des messages hello (utilisés dans la découverte de voisinage) pour assurer la validité des liens entre les nœuds voisins. Un nœud considère que le lien avec son voisin est rompu s'il ne reçoit pas de sa part un paquet hello dans un intervalle de temps. Une seconde méthode de détection de liens rompus est basée sur la non réception des acquittements lors de la transmission des paquets de données. Cependant, cette méthode exige une couche « liaison de données » employant ce type d'acquittements. Dès qu'un nœud détecte une défaillance de lien, il envoie immédiatement un agent rectificateur aux nœuds voisins concernés (voir figure 5-14). Ensuite, il supprime de sa table de routage toutes les entrées utilisant le lien cassé. Dans le deuxième cas, l'agent *RectifierAnt* est utilisé pour modifier des entrées dans les tables de routage des nœuds voisins suite à un changement dans les informations de routage du nœud courant. Lorsque un nœud subit une mise à jour d'une route dans sa table de routage (ex : l'agent *Ant* insère une route optimale à la place d'une ancienne), il transmet les nouvelles informations aux voisins concernés par cette route via des agents rectificateurs. Dans ce qui suit, nous allons présenter un algorithme dans lequel nous décrivons un exemple d'initialisation d'un agent rectificateur lors de la détection d'un lien rompu avec un voisin. Nous présentons ensuite, à travers un autre algorithme, les différentes opérations effectuées par l'agent rectificateur dès son arrivée au nœud vers lequel il est envoyé.

Algorithme 6 Initialisation d'un agent rectificateur

Soient

- *n* un nœud qui détecte une rupture de lien avec un nœud voisin *m*
- *Ant* est l'agent rectificateur

Supprimer l'entrée correspondante dans la table de voisinage
pour toute destination *dest* $\in TABROUT_n$ tq *suiv(init(Route(n, dest))) = m* faire
    Supprimer cette entrée
    pour tous nœud *x* $\in TABROUT_n$*[dest, m].ListV et x* $\in TABV_n$ *faire{*
        //Lui envoyer un agent rectificateur contenant les informations suivantes :
        Ant.NoeudSuiv =x
        Ant.NoeudEtabRoute =dest
        Ant.NoeudPreced =n
        choisir RouteOptimale(n, dest)
        Si *(Ant.valide=vrai)*
            $TABROUT_n$ *[Ant.NoeudEtabRoute, suiv(init(RouteOptimale(n,dest)))].ListV*
        +ant.NoeudSuiv
}

L'algorithme suivant permet de décrire les activités effectuées par un agent *RectifierAnt* lors de son arrivée sur un nœud voisin :

Algorithme 7 Taches d'un agent rectificateur

Soit

- *Ant* un agent rectificateur provenant du nœud *n-1* afin de mettre à jour la table de routage du nœud *n*

Si $TABROUT_n[Ant.NoeudEtabRoute, Ant.NoeudPreced]$ existe{
    Si *(Ant.valide=vrai)*{
        *Ant.t =ant.t+*temps estimé par l'agent pour aller du nœud *n-1* vers le nœud *n*
        Ant.NS++
        $TABROUT_n[Ant.NoeudEtabRoute, Ant.NoeudPreced].(d/t) := ant.NS/ant.t$
        $TABROUT_n[Ant.NoeudEtabRoute, Ant.NoeudPreced].SeqNumber := Ant.seqNumber$
        Envoyer un agent rectificateur vers tous nœud $x \in TABROUT_n[Ant.NoeudEtabRoute, Ant.NoeudPreced].ListV$
    }Sinon{
        Supprimer $TABROUT_n[Ant.NoeudEtabRoute, Ant.NoeudPreced]$
        Envoyer un agent rectificateur vers tous nœud $x \in TABROUT_n[Ant.NoeudEtabRoute, Ant.NoeudPreced].ListV$
    }
}

5.2.4. Simulations

Nous avons évalué notre protocole à travers une série de tests de simulations. Nous avons comparé ses performances avec celles des protocoles AODV [PBRD03] (protocole réactif), DSDV [PB94] (protocole proactif) et AntHocNet [DCDG04] (protocole hybride inspiré du principe des colonies de fourmis). Nous présentons dans ce qui suit l'environnement de simulation, les scénarios des tests et nous terminons par une analyse des résultats obtenus.

5.2.4.1. Environnement de simulation

La simulation du protocole que nous avons proposé est réalisée à l'aide du simulateur NS2 [AH99]. Le protocole a été développé totalement avec le langage C++ et intégré dans NS2 en respectant la même méthodologie de développement que les autres protocoles existants dans ce simulateur. La vitesse maximale des nœuds dans un scénario est fixée à 30m/s, la fréquence d'envoi des agents *Ant* est de 0.5s, la fréquence d'évaporation de la phéromone est de 0.5s, le taux d'évaporation « alpha » est de 0.1 (équation (3)) et la valeur de mise à jour de la phéromone « q » est

de 0.1 (équation(1)). Le trafic a été généré automatiquement à l'aide d'un script TCL que nous avons développé. Les scénarios sont produits par l'outil SETDEST de NS2. Le trafic est généré aléatoirement de la façon suivante: n communications sont établies en choisissant n couples de nœuds au hasard. Une communication consiste à l'envoie d'un paquet de 512 octets au moyen du protocole UDP. Enfin, le nombre total de paquets de données varie de 700 à 1000 par simulation.

Rappelons que nous avons comparé notre protocole avec trois autres protocoles qui sont de familles différentes : réactive, proactive et hybride et qui sont respectivement : AODV, DSDV et AntHocNet. Pour cela, nous avons choisi un réseau à densité moyenne (ce choix est justifié par le fait que ce type de réseau est le plus répandu dans la pratique). Les métriques d'évaluation que nous avons choisies sont :

Le nombre de paquets perdus (en pourcentage) : cette métrique mesure le nombre de messages de données qui n'ont pas atteint leur destination ; elle permet de montrer la robustesse et l'efficacité du protocole ainsi que des informations concernant la congestion du réseau.

Les délais de transmission : Cette métrique mesure le délai moyen entre le moment d'envoi d'un paquet de données et le moment de sa réception.

La taille totale des messages de contrôles générés par le protocole

Pour chacune de ces métriques, nous avons fait varier d'une simulation à une autre, le nombre de nœuds dans le réseau (entre 60 nœuds et 100 nœuds), le nombre total de paquets de données (entre 700 et 1000 paquets) ainsi que la vitesse de déplacement des nœuds du réseau (entre 5m/s et 30m/s).

5.2.4.2. Les résultats de simulation

Dans ce qui suit, nous présentons les résultats de simulation de notre protocole ainsi qu'une comparaison de ses performances avec les protocoles cités précédemment.

5.2.4.2.1. Le taux de perte de paquet

Cette métrique représente l'un des indicateurs les plus intéressants pour évaluer un protocole de routage.

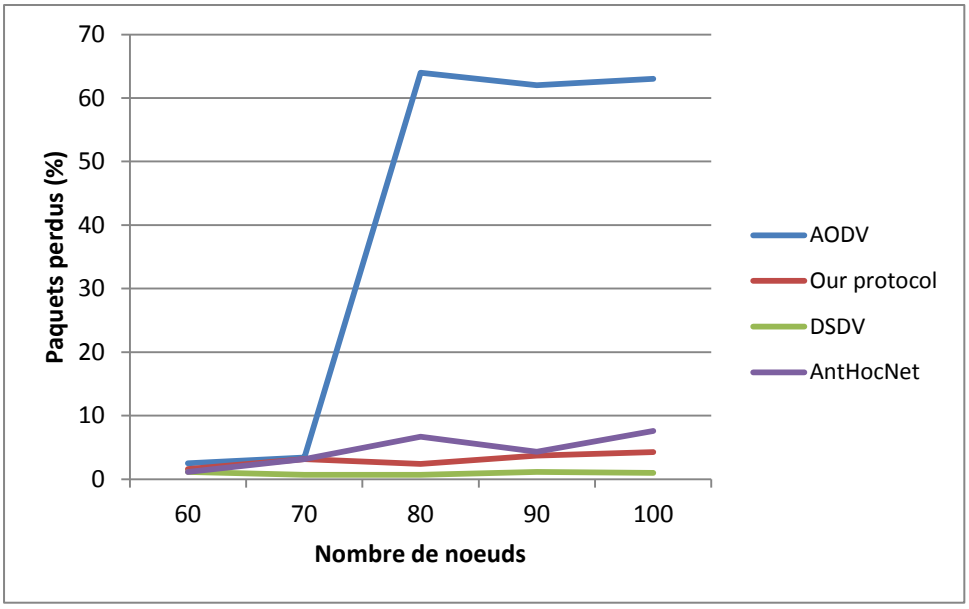


Figure 5-15- Pourcentage des paquets perdus en fonction du nombre de nœuds dans le réseau

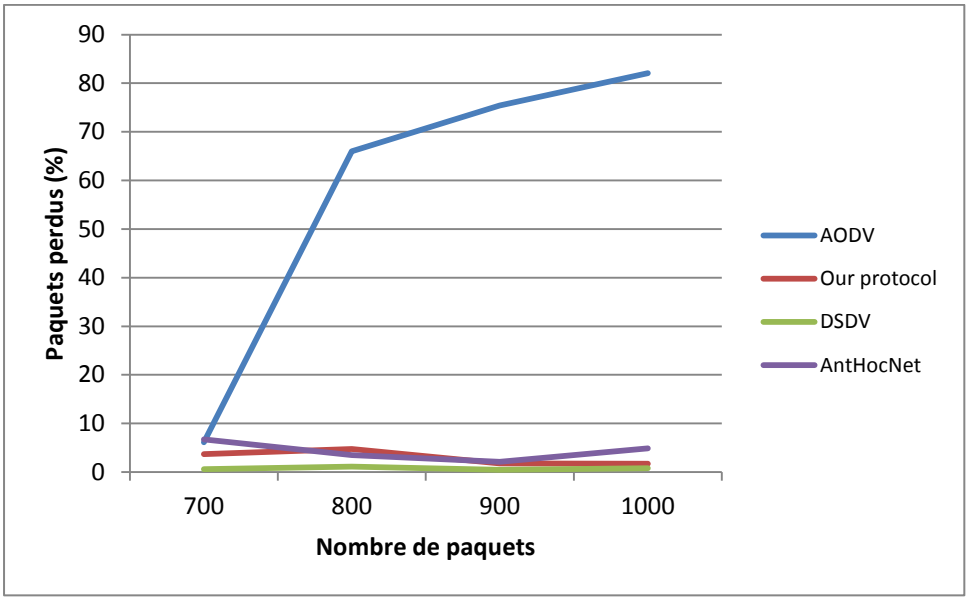


Figure 5-16- Pourcentage des paquets perdus en fonction du nombre total de paquets de données

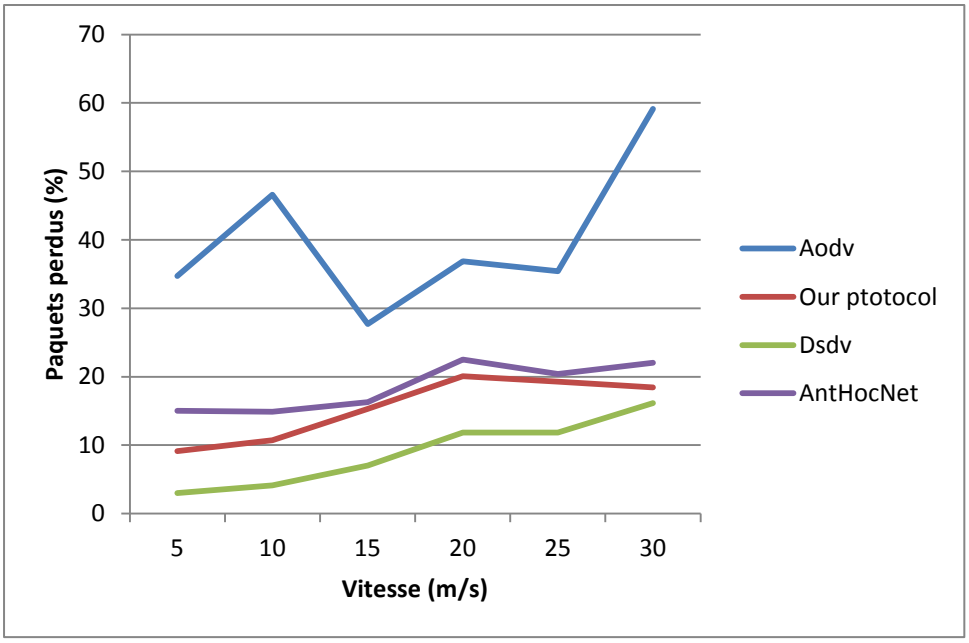


Figure 5-17- Pourcentage des paquets perdus en fonction de la vitesse (m/s) des nœuds

Les deux premières figures (5-15 et 5-16) montrent la variation du pourcentage des paquets perdus en fonction de la variation du nombre de nœuds et du nombre total de paquet de données. On remarque que notre protocole, le protocole DSDV ainsi que le protocole AntHocNet ont beaucoup moins de pertes de paquets par rapport à AODV, où le nombre de paquets perdus augmente rapidement dans les réseaux dont le nombre de nœuds dépasse 70 et ceux qui contiennent un fort trafic ; cela est dû au fait que AODV utilise le principe de diffusion qui génère trop de messages de contrôle et par conséquent augmente les possibilités de collisions et surcharge le réseau en engendrant beaucoup de paquets de données non reçus par leurs destinataires. Par rapport à AODV, notre protocole permet donc un plus grand taux de livraison de paquets et garde cette propriété avec l'augmentation de la densité du réseau ; cela peut être expliqué par le fait que notre protocole utilise une approche de découverte de routes réactive efficace qui diffère de celles utilisant le mécanisme de diffusion. La troisième figure, qui représente la variation du pourcentage des paquets perdus en fonction de la variation de la vitesse des nœuds, montre que notre proposition a moins de paquets perdus par rapport à AODV et AntHocNet en faisant varier la vitesse des nœuds ; cela est justifié en partie par le fait que, dans le cas de notre protocole, les pannes de liaison sont en partie prises en charge par les agents rectificateurs. La diversité des chemins permet aussi au protocole d'être plus résistant face aux changements topologiques.

5.2.4.2.2. Délai de communication

La rapidité et le temps de réponse d'un protocole de routage peuvent être mesurés par cette métrique. Avoir des délais courts permet d'avoir un protocole adapté pour plusieurs applications.

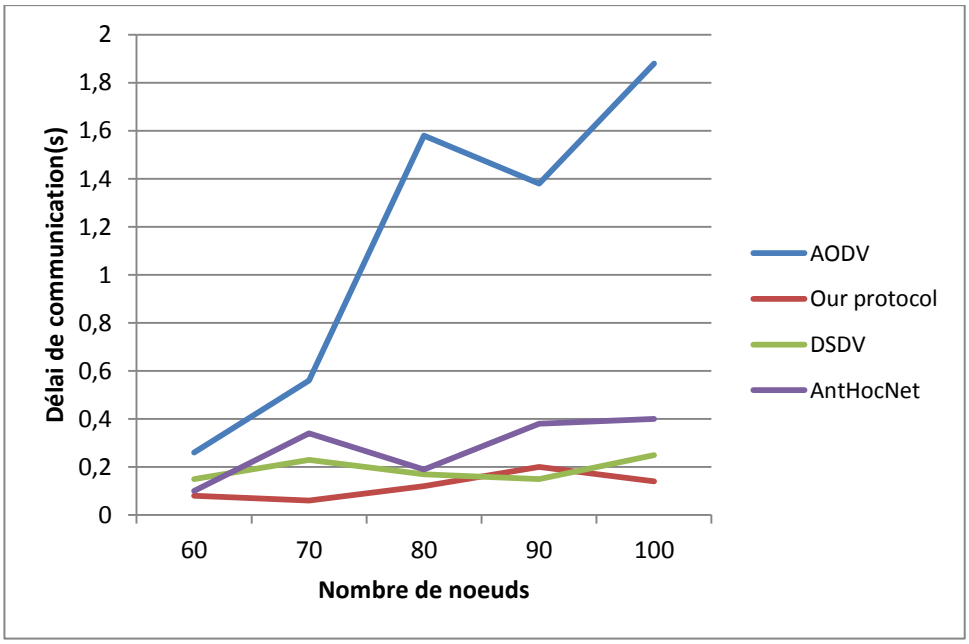


Figure 5-18- Délai de transmission (s) en fonction du nombre de nœuds dans le réseau

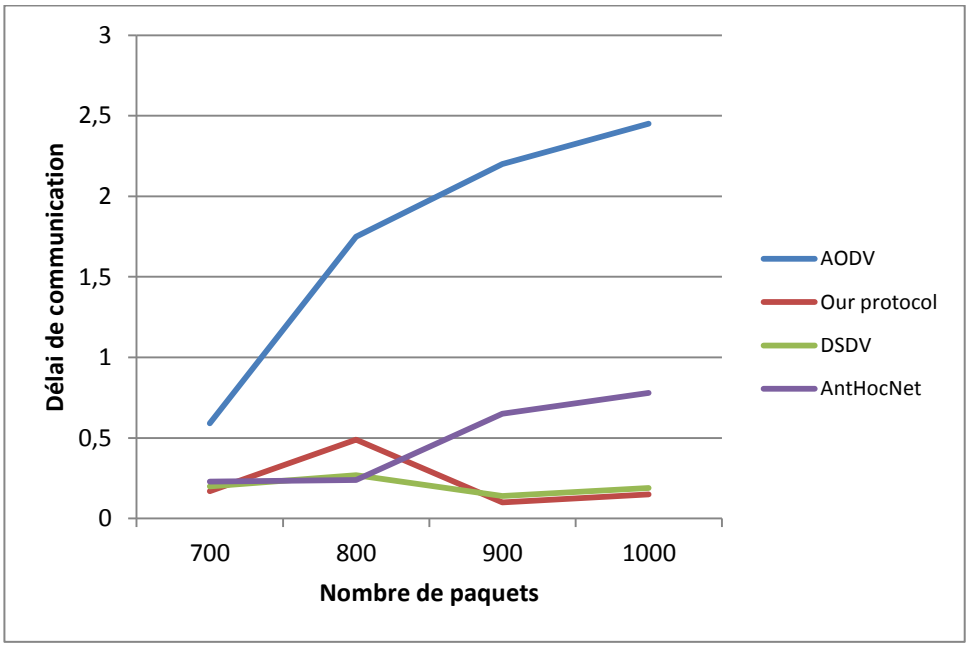


Figure 5-19- Délai de transmission (s) en fonction du nombre total de paquets de données

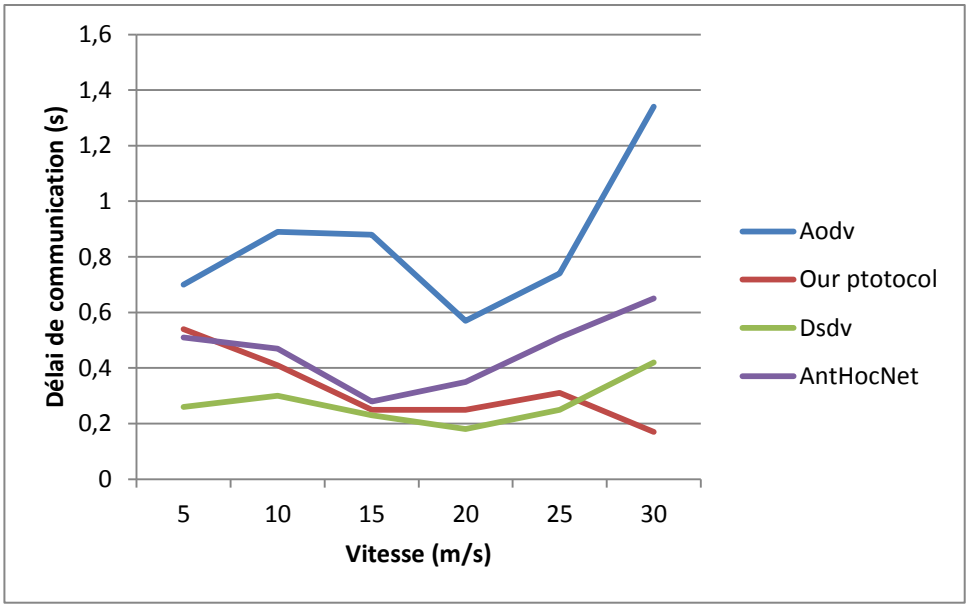


Figure 5-20- Délai de transmission (s) en fonction de la vitesse (m/s) des nœuds

Les trois figures (5-18, 5-19 et 5-20) montrent la variation du délai moyen de transmission en fonction de la variation du nombre de nœuds dans le réseau, du nombre total de paquet de données et de la vitesse des nœuds dû à la mobilité. D'après les figures, nous remarquons que notre protocole ainsi que le protocole DSDV présentent des délais de communication plus petits que ceux de AODV et AntHocNet, Notre protocole et DSDV sont donc plus efficace et possèdent des procédures de routage plus rapide que celles des protocoles AODV et AntHocNet, en particulier dans le cas d'un grand nombre de nœuds et de paquets de données, et plus particulièrement dans le cas de vitesses élevées des nœuds. Concernant notre protocole, cette efficacité se justifie par les points suivants :

– Le caractère proactif de notre protocole: par définition du routage proactif, le nœud souhaitant envoyer un paquet de donnée n'aura pas besoin de procéder à une demande de route (qui consomme du temps et génère beaucoup de trafic réseau), puisque les routes sont déjà existantes dans sa table de routage. Vu que notre protocole est hybride, donc une bonne partie de paquets sera routée avec un temps d'établissement de route=0. Cela permet de réduire considérablement les délais de transmission.

– Comme notre protocole est multi chemins, il permet à chaque nœud d'avoir plusieurs chemins (vers la même destination) chaque fois qu'il veut envoyer un paquet de données. Ce nœud peut donc envoyer ses paquets même après les ruptures de liens sans initialiser un nouveau processus de découverte de routes.

– Contrairement au protocole AODV, notre protocole n'utilise pas de mécanismes de diffusion lors de sa partie réactive. Rappelons que ce mécanisme augmente considérablement la charge du réseau et par conséquence influence négativement sur les délais de transmission.

5.2.4.2.3. La taille totale des messages de contrôle

Cette métrique mesure la consommation du protocole en matière de messages de contrôle générés. Par conséquence, les résultats obtenus nous donnent une idée sur sa consommation en termes de bande passante et d'énergie des nœuds. Notons que ce sont généralement les messages utilisés pour la découverte et la maintenance des routes qui consomment le plus de bande passante pour un protocole de routage.

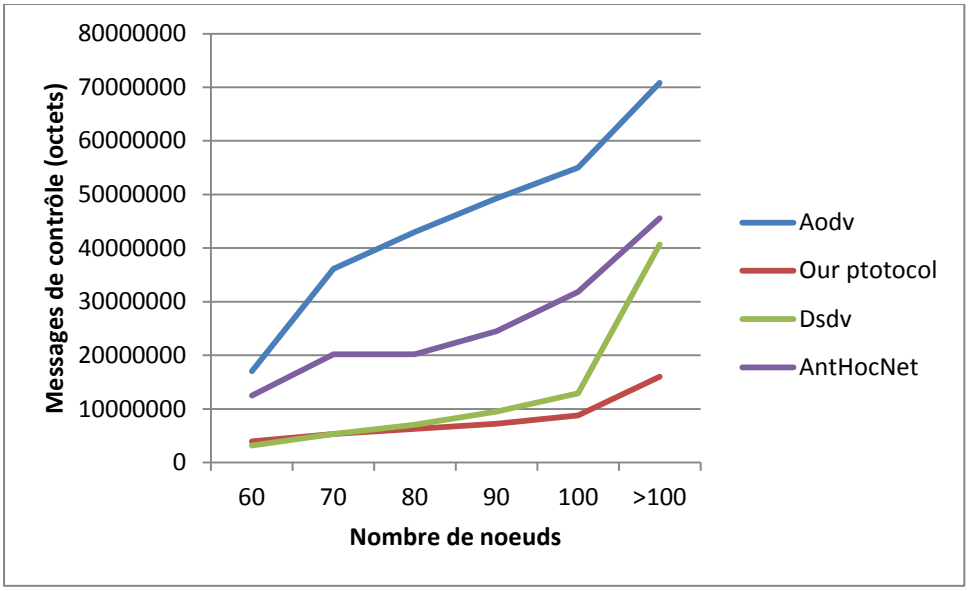


Figure 5-21- La taille totale des messages de contrôle (octets) en fonction du nombre de nœuds dans le réseau

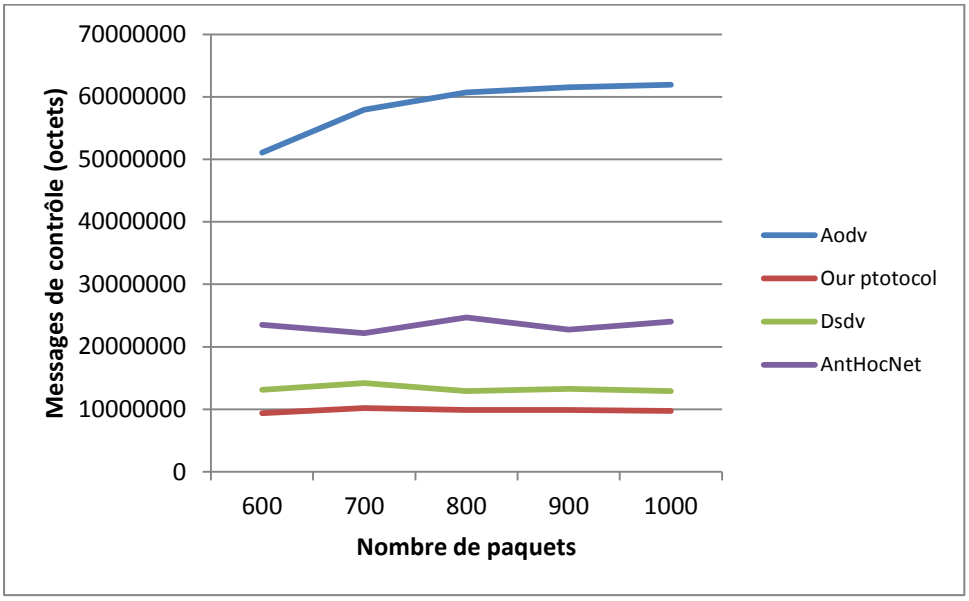


Figure 5-22- La taille totale des messages de contrôle (octets) en fonction du nombre total de paquets de données

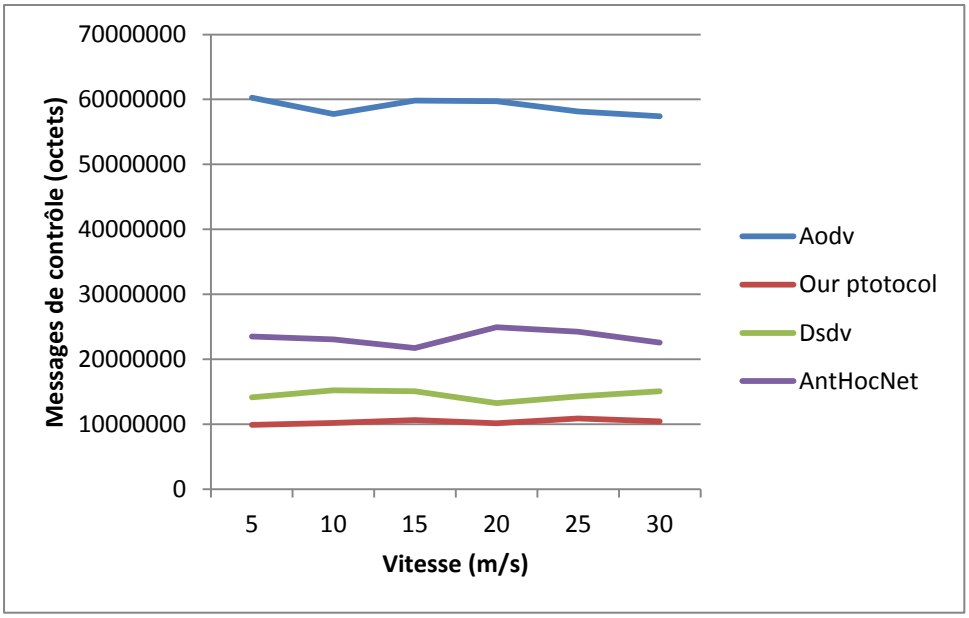


Figure 5-23-La taille totale des messages de contrôle (octets) en fonction de la vitesse (m/s) des nœuds

Les figures (5-21, 5-22 et 5-23) représentent respectivement la variation de la taille globale des messages de contrôle en fonction de la variation du nombre de nœuds dans le réseau, du nombre total de paquet de données et de la vitesse des nœuds. Nous remarquons clairement que notre protocole produit moins de messages de contrôles que les protocoles AODV, AntHocNet et même DSDV. Le nombre de paquets de contrôle augmente lentement et linéairement dans le cas de notre protocole même avec un grand nombre de nœuds et/ou de paquets de données, contrairement au protocole AODV où les messages de contrôle augmentent très rapidement. Cela est dû au fait que notre protocole gère et contrôle au niveau de chaque nœud le nombre d'agents dans le réseau qui est proportionnel à la taille du réseau et à la fréquence d'envoie des agents. Ainsi, notre protocole évite les techniques de diffusions qui génèrent trop de paquets de contrôle. Au lieu de cela, nous utilisons une méthode plus intelligente qui ne fait qu'utiliser les agents disponibles dans le réseau. Dans AODV par exemple, le nombre de paquets de contrôle dépend du nombre de demandes de routes qui influencent négativement sur la bande passante et les ressources énergétiques des nœuds du réseau puisque chaque nouvelle demande de route nécessite un certain nombre de diffusions. Cela augmente le nombre de collusions surtout dans un réseau dense. Ces collisions conduisent à l'augmentation du nombre de retransmissions et par conséquent induisent l'augmentation du nombre total de paquets dans le réseau. Ces figures montrent que notre protocole génère aussi moins de paquets de contrôle que le protocole DSDV, un protocole réputé efficace dans ce type de réseau.

Remarque : La version actuelle du protocole est obtenue suite à plusieurs tests en faisant varier à chaque fois certains paramètres pertinents du protocole à savoir: la fréquence d'envoi des agents Ant, la quantité initiale de

phéromone, la quantité de phéromone à incrémenter ou à décrémenter lors du phénomène d'évaporation, la valeur du TTL, etc.

5.3. Conclusion

Les algorithmes de routage classiques (utilisés dans les réseaux filaires) ne sont pas adaptés aux réseaux ad hoc et cela est dû au fait que ces algorithmes ne prennent pas en considération les spécificités de ce type de réseaux à savoir la puissance de calcul et les ressources énergétiques limitées des nœuds ainsi que la mobilité et l'équivalence des nœuds. Nous avons vu dans ce chapitre que les protocoles de routage dédiés aux réseaux ad hoc, proposés dans la littérature, se divisent en deux catégories : Les protocoles proactifs et les protocoles réactifs. L'inconvénient commun des deux catégories est essentiellement le nombre important de messages de contrôle qui influence négativement sur la bande passante, les ressources énergétiques des nœuds et les performances du réseau d'une manière générale. Cela produit aussi le problème de la congestion du réseau et par conséquent une perte de paquets et des délais de transmission élevés dans le réseau. Une troisième famille de protocole de routage, appelée hybride (réactive et proactive), a été proposée dans le but de bénéficier des avantages des deux approches. Parmi les protocoles utilisant ce principe, nous pouvons citer les protocoles inspirés du fonctionnement des colonies de fourmis qui s'adaptent bien aux changements de topologie et au caractère autonome des réseaux ad hoc. Ces protocoles souffrent malgré tout du problème de congestion du réseau et de perte de paquets à cause d'une diffusion de type *broadcast* de leur agent fourmi.

Pour répondre en partie à ces problématiques, nous avons proposé un nouveau protocole de routage hybride inspiré du fonctionnement des colonies de fourmis mais dont le principe diffère de celui des protocoles de cette famille et présentés dans ce chapitre. Le protocole proposé évite le principe de diffusion lors des demandes de route : la demande se fait localement au niveau de chaque nœud demandeur. Les messages de contrôle du protocole sont des agents et non pas des tables de routage contrairement aux protocoles proactifs existants. Ces agents circulent sans diffusion contrairement aux protocoles existants (réactifs, proactifs et hybride). Les résultats des simulations de notre protocole et sa comparaison avec d'autres protocoles (AODV, DSDV et AntHotNet) sont très encourageants. Notre protocole réduit considérablement le nombre de messages de contrôle, les délais de communication ainsi que le nombre de paquets perdus.

Nous envisageons à l'avenir de tester notre protocole dans d'autres environnements et d'autres topologies. Nous comptons évaluer notre protocole en faisant varier d'autres métriques tel que la densité du réseau et plus généralement porter des améliorations à notre protocole sous l'angle de l'optimisation de sa qualité de service (Qos).

CHAPITRE 6

6. Etude de la réputation des nœuds dans les réseaux ad hoc

Le mauvais comportement des nœuds est un problème majeur dont souffrent les protocoles de routage dans les réseaux ad hoc. Les protocoles de routage proposés dans la littérature (AODV, DSDV, DSR, OLSR, AntHocNet,…etc) font tous l'hypothèse que tous les nœuds du réseau coopèrent pour router les messages des autres, ce qui n'est pas toujours le cas dans un réseau dynamique ou les ressources énergétiques et les puissances de calcul des nœuds sont limitées. Un nœud peut refuser de router un message de donnée ou de contrôle afin d'économiser son énergie par exemple. Un modèle de renforcement de la coopération entre les nœuds est donc nécessaire afin de réduire la dégradation des performances et des fonctionnalités de ces réseaux dans le routage des messages. C'est sur cet aspect que porte notre contribution dans ce chapitre.

Dans la section suivante, nous allons présenter quelques modèles de renforcement de la coopération existants. Nous présenterons par la suite notre contribution en proposant un modèle de coopération entre les nœuds du réseau. Ce modèle est intégré à notre protocole de routage détaillé dans le chapitre précédent.

6.1. Solutions pour renforcer la coopération des nœuds

Dans les réseaux ad hoc, les nœuds mobiles sont limités par des faibles capacités de calcul et d'énergie. Par conséquent, il peut y avoir des nœuds ayant tendance naturellement à avoir un comportement « égoïste » en refusant de relayer les messages (de contrôle ou de données) des autres nœuds du réseau. Une solution à ce problème consiste à employer des systèmes de renforcement de la coopération des entités mobiles. Ces systèmes visent principalement à encourager les nœuds à collaborer entre eux et effectuer leurs tâches correctement. Il existe grosso modo deux sortes de solutions de renforcement de la coopération: les solutions basées sur un système de réputation et les solutions basées sur un modèle de micro-payement (appelé aussi modèle basé sur le crédit).

Les modèles basés sur la réputation des nœuds peuvent être présentés comme une couche de sécurité additionnelle permettant d'observer le comportement des nœuds du réseau. Chaque système proposé présente un outil extrêmement important qui permet aux entités d'évaluer le niveau de réputation entre elles. Autrement dit, ces systèmes offrent la possibilité à chaque nœud de mesurer la fiabilité d'une autre entité avant de décider d'interagir ou d'entrer en communication avec elle. La

réputation des nœuds est calculée en fonction de l'acheminement des paquets : à chaque fois qu'un nœud fait router correctement les paquets diffusés par ses voisins, sa valeur de réputation augmente. Les nœuds non coopératifs ayant une faible valeur de réputation sont mis en quarantaine grâce à des techniques d'isolation implémentées dans le modèle proposé. Les systèmes basés sur la réputation sont divisés en deux sous classes : les modèles où les nœuds s'appuient sur leurs propres observations pour évaluer la réputation des autres nœuds voisins et les modèles où les nœuds prennent en considération les observations des autres nœuds afin d'allouer les valeurs de réputation. Des systèmes opérationnels qui implémentent ce type de modèles existent déjà, nous citons par exemple OCEAN proposés dans [BB03], et CORE [MM02]. Dans les paragraphes suivants, nous présentons deux autres exemples dénommés CONFIDANT [BLB02], et SORI [HWK04]. Ces mécanismes sont présentés comme étant une extension aux protocoles de routage qui existent auparavant.

Il existe plusieurs version du protocole CONFIDANT (COoperation of Nodes, Fairness In Dynamic Ad-hoc NeTworks) [BLB02] [BB02, BBo02, BB04, BBo04] ici nous résumons le principe de base et le fonctionnement global introduits dans les travaux de Bouchegger et Le Boudec. Le mécanisme proposé est appliqué au protocole DSR afin d'avoir un système de renforcement de la coopération distribué et collaboratif et d'exclure les entités égoïstes du réseau. Les fonctionnalités du mécanisme sont établies au niveau d'exécution des processus de routage des données et de découverte de voisinage. Chaque nœud du réseau utilise quatre modules spécifiques qui interagissent entre eux.

1. Un moniteur ;
2. Un système de réputation ;
3. Un gestionnaire de confiance ;
4. Un gestionnaire de chemins.

Au cours des opérations de routage, le moniteur se charge de la fonction d'observation et d'évaluation des comportements des nœuds. Les mauvais comportements évalués par le moniteur sont envoyés au système de réputation qui maintient à jours les valeurs de réputation de chaque nœud observé. CONFIDANT utilise des techniques avancées pour renforcer la précision des mécanismes de détection. Ces techniques sont implémentées dans le module gestionnaire de confiance qui s'appuie principalement sur l'échange de recommandations entre les nœuds du réseau (CONFIDANT prend en considération seulement les recommandations négatives.). Le rôle principal du gestionnaire de confiance est d'agréger ensemble les recommandations reçues, et de prendre des décisions quant au partage des valeurs de réputation. Le dernier composant utilisé par CONFIDANT est le gestionnaire de chemin ; ce module intervient lors de la sélection des routes optimales, il favorise les chemins caractérisés par la fiabilité des nœuds qui les construisent, et peut éviter l'acheminement des paquets des nœuds ayant une faible note de réputation (qui est inférieure au seuil toléré par CONFIDANT).

Un autre protocole semblable à CONFIDANT a été proposé dans [HWK04]. Le protocole s'appelle SORI (Secure and Objective Reputation-based Incentive), et lutte principalement contre les nœuds qui ne retransmettent pas les paquets envoyés par leurs voisins. Dans ce mécanisme, la réputation des nœuds est observée dans une zone de voisinage locale limitée à un saut. Pour se faire, les entités employant ce protocole procède comme suit :

Chaque nœud N maintient localement une liste des voisins à un saut noté (NNLN), et pour chaque voisin X il sauvegarde les deux paramètres suivants :

$RF_N(X)$ *(Request For Forwarding)* qui représente le nombre total de paquets envoyés par le nœud N et à transmettre par le voisin X.

$HF_N(X)$ *(has-forwarded)* qui correspond au nombre total des paquets qui ont été transmis par le voisin X et observés par le nœud N.

Une fois ces deux paramètres calculés, le nœud N calcule la valeur de réputation (notée $G_N(X)$) de chacun de ses voisin, et pour approuver la confiance du nœud N par rapport à ses jugements sur la réputation de ses voisins, une valeur de niveau de confiance (noté $C_N(X)$) est calculée. La réputation de chaque voisin X, ainsi que le niveau de confiance sont obtenus selon les deux formules suivantes :

$$G_N(X) = \frac{RF_N(X)}{HF_N(X)} \Rightarrow G_N(X) \in [0,1] \quad et \quad C_N(X) = RF_N(X)$$

Apres avoir calculé ces deux valeurs, N crée un enregistrement local pour chaque voisin X dénommé $(LER_N(X))$ *(Local Evaluation Record)*, ces enregistrements sont mis à jour périodiquement en se basant sur les valeurs courantes de $RF_N(x)$ et $HF_N(x)$. L'enregistrement est diffusé au voisinage de N si la valeur de réputation d'un voisin a subi un changement significatif. N utilise l'enregistrement local $(LER_N(X))$ qui décrit ses observations sur le comportement d'un voisin X et l'enregistrement $(LERi(X))$ (tel que $i \in NNL_N \,\&\, i \neq X$) qui décrit les observations de son voisinage sur le comportement du même voisin X pour faire une évaluation globale du comportement de ce nœud. Cette évaluation est noté $OER_N(X)$ *(Overall Evaluation Record)* et elle est calculée comme suit :

$$OER_N(X) = \frac{\sum_{i \in NNL_N \cup \{N\},\ i \neq X} \lambda_N(i).C_i(X).G_i(X)}{\sum_{k \in NNL_N \cup \{N\},\ k \neq X} \lambda_N(k).C_k(X)}$$

Le paramètre λ présente la crédibilité du nœud i obtenue selon le point de vue du nœud N, dans ce schéma les auteurs ont affecté pour λ les valeurs suivantes : $\lambda_N(i) = G_N(i)$, spécialement $\lambda_N(N) = 1$, et $\lambda_N(i) = 0$ si $RF_N(i) = 0$.

En utilisant le paramètre $OER_N(X)$, N peut juger le comportement de chacun de ses voisins et prendre des décisions. Si la valeur d'évaluation globale d'un voisin X est inférieure au seuil fixé par le protocole, N punit le nœud X par une suppression probabiliste des paquets provenant de ce nœud. La formule de probabilité utilisée dans ce cas est présentée comme suit :

$$p = \begin{cases} q - \delta \; si \; q > \delta \; tel \; que: \; q = 1 - OER_N(X) \; et \; 0 < \delta < 1 \\ 0 \quad sinon \end{cases}$$

δ est un paramètre d'un système utilisé pour introduire une marge d'erreur qui représente les paquets abandonnés pour des raisons autres que l'égoïsme comme le phénomène de la collision.

SORI propose des mécanismes de sécurité supplémentaires. Ces mécanismes fournissent l'authentification originale des messages de réputation. Cela est obtenu en utilisant le système d'authentification TESLA, comme utilisé dans le protocole ARIADNE [HPJ05].

Le deuxième modèle de renforcement de la coopération des nœuds est le modèle basé sur le crédit. Dans ce type de modèle, la fonction de routage est considérée comme un service pouvant être évalué et rémunéré. Ces systèmes incorporent une sorte de monnaie virtuelle permettant de réguler le trafic entre les nœuds lors du routage des paquets. Tous les nœuds qui profitent du réseau que ce soit un émetteur et/ou un récepteur payent les nœuds intermédiaires fournisseurs de service. Des Système opérationnels répondant à ce type de modèles existent déjà, nous citons par exemple : TOKEN BASED [YML02], Sprit [ZCY03] et ad hoc-VCG [AE03].

Dans le modèle TOKEN BASED [YML02], toute entité mobile souhaitant accéder au réseau ad hoc doit présenter un jeton spécifique valable dans une période limitée. Pendant cette période, les voisins du nœud nouvellement inséré dans le réseau surveillent son comportement pour détecter une action malveillante, ou un mauvais comportement. Une fois que la période de validité du jeton est expirée, le nœud doit présenter une demande de renouvellement de sa participation auprès de ses voisins. La période de validité des jetons dépend extrêmement de la durée de participation active des nœuds au réseau.

6.2. Notre proposition : un modèle de coopération dans les réseaux ad hoc

Nous proposons dans cette section, un nouveau protocole de renforcement de la coopération entre les nœuds du réseau afin de répondre en partie au comportement « égoïste » de certains nœuds du réseau. Cette fonctionnalité est intégrée dans le protocole de routage qui a été présenté dans le chapitre précédent. Les simulations réalisées sur le fonctionnement de notre protocole ont montré que le nombre de messages de contrôle n'est pas très élevé comparé aux autres protocoles. Nous avons

profité de cette marge pour doter le protocole de routage d'une nouvelle fonctionnalité lui permettant de « punir » les nœuds « égoïstes ». Le mécanisme de renforcement de la coopération ajouté au protocole de routage est basé sur le calcul de la réputation des nœuds. Ce processus permet de construire au cours du temps, une relation de confiance entre les nœuds du réseau en prenant en considération leurs propres connaissances ainsi que celles des autres nœuds pour permettre aux nœuds de mieux conforter leurs décisions. L'objectif principal est de concevoir et mettre en œuvre un protocole de routage pour réseaux ad hoc performant et avec un calcul des routes « sûres ».

Principe de fonctionnement du protocole : Le mécanisme de renforcement de la coopération se compose des modules suivants (cf. figure 6-1) :

– Un module de collecte d'informations sur le comportement de chaque nœud du réseau en matière de routage de messages.
– Un module pour le calcul de la réputation de chaque nœud.
– Un module pour la punition des nœuds (isolation) qui se comportent mal (nœuds égoïstes).

Le protocole va doter chaque nœud N du réseau d'un agent local appelé « *RéputationAgent* » qui a comme rôle la surveillance des nœuds voisins en ce qui concerne l'envoi et la réception des paquets (l'analyse du trafic). Cette surveillance est effectuée en utilisant le mode promiscous et cela afin de calculer à chaque période de temps la réputation du nœud voisin surveillé Ns. Cette étape va servir pour définir un degré de pénalisation du nœud Ns par le nœud voisin qui le surveille définie par une valeur p comprise entre 0 et 1. La décision relative à la pénalisation ou pas du nœud surveillé est une fonction probabiliste (stochastique) dont la formule sera détaillée par la suite. Dans le cas où la décision est oui, le nœud surveillé sera pénalisé pendant une durée proportionnelle à p et aussi en fonction de son historique. La pénalisation d'un nœud N vis à vis d'un nœud Ns consistera alors à empêcher les agents mobiles *Ant* qui passent par le nœud N de se diriger vers le nœud sanctionné Ns. Cette décision aura pour conséquence que la mise à jour de la table de routage du nœud pénalisé sera moins fréquente voir rare si les autres voisins appliquent la même stratégie de pénalisation envers lui. Le nœud pénalisé sera donc isolé du réseau puisque il ne pourra pas envoyer ses paquets aux autres nœuds du réseau avec lesquels il veut communiquer (sa table de routage n'étant pas mise à jour et par conséquent peu de routes seront construites). Notons qu'à chaque fois qu'un nœud se comporte mal, sa note de réputation au niveau des autres nœuds régresse.

La solution proposée est décentralisée et s'adapte totalement au protocole de routage que nous avons proposé puisque les messages de contrôle (Agents mobiles) sont envoyés en unicast et non en broadcast

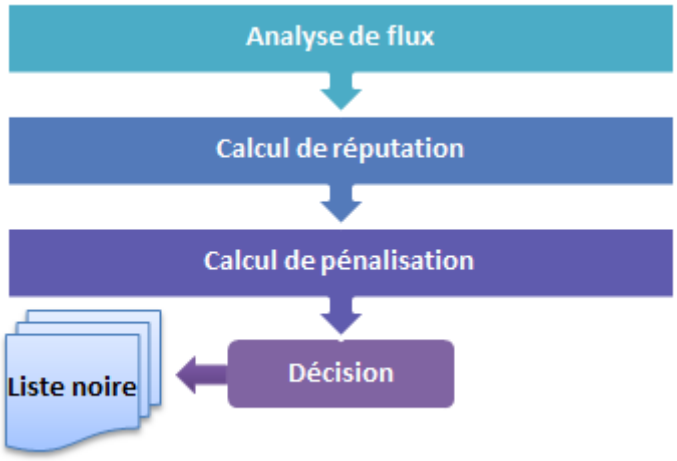


Figure 6-1- Les modules de notre système de réputation

Dans ce qui suit, nous détaillerons les fonctionnalités de chaque module décrit précédemment : la collecte d'informations, le calcul de la réputation et la décision de pénalisation d'un nœud.

6.2.1. Le module collecte d'information

A chaque intervalle de temps, chaque nœud du réseau surveille les activités de ses voisins, dans le but d'observer s'ils ont relayé les messages reçus ou pas. Pour cela, chaque nœud voisin conserve les informations suivantes:

- $Input_N$: représente le nombre de paquets envoyés par le nœud N au nœud voisin surveillé pour être retransmis.
- $Forward_N$: représente le nombre de paquets du nœud N qui ont été réellement retransmis par le nœud voisin surveillé.
- $Input_{others}$: représente le nombre de paquets envoyés par les autres nœuds voisins (autres que le nœud N) au nœud voisin surveillé pour être retransmis.
- $Forward_{others}$: le nombre de paquets des autres nœuds voisins (autres que le nœud N) qui ont été réellement retransmis par le nœud voisin surveillé.

La figure 6-2 résume le fonctionnement de ce module. Notons que ce processus utilise le mode *promiscous* (c'est à dire accepter tous les messages qu'un nœud reçoit, même si ceux-ci ne lui sont pas destinés).

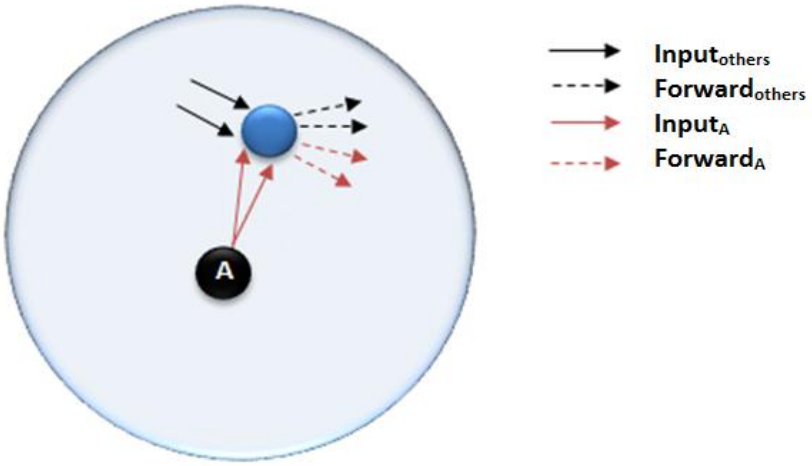


Figure 6-2- Le module collecte d'informations

6.2.2. Le calcul de la réputation

Pour bâtir des relations de confiance entre les éléments du réseau, chaque nœud calcule pour chacun des nœuds voisins qui figurent dans sa table de routage une valeur représentant sa note de réputation NR. Cette valeur permet à un nœud de connaitre les nœuds auxquels il pourra faire confiance. Un nœud est dit « de confiance » si sa note de réputation dépasse un seuil prédéterminé. Nous avons expliqué précédemment que chaque nœud N calcule sa confiance envers un autre en se basant sur ses propres observations ainsi que celles (les recommandations) des autres nœuds. Pour cela, la note de réputation est calculée à partir de deux autres valeurs qui sont:

La note de réputation directe : le nœud N se base sur sa propre surveillance du comportement de son voisin.

La note de réputation externe : le nœud N prend en considération les recommandations des autres nœuds du réseau au sujet du nœud pour lequel il veut évaluer la confiance.

Dans ce qui suit, nous détaillerons le calcul de chaque valeur.

6.2.2.1. La note de réputation directe

Pour le calcul de cette note, le nœud surveillant se base sur ses propres observations de son voisin en utilisant le mode promiscous. Ci-après la formule qui permet à un nœud A de calculer la note de réputation directe au sujet de son voisin B:

$$\boldsymbol{DRN(A,B) = \alpha \frac{Forward_A}{Input_A} + (1-\alpha)\frac{Forward_{others}}{Input_{others}}} \qquad (1)$$

$$0.5 < \alpha < 1$$

$\boldsymbol{DRN(A,B)}$ Est la note de réputation directe calculée par le nœud A concernant son voisin B.

$Input_A$, $Forward_A$, $Input_{others}$ et $Forward_{others}$ correspondent aux valeurs récupérées à partir du module précédent (voir le module collecte d'information 6.2.1).

Le contenu de la valeur α sert à régler le poids à donner au comportement du nœud B concernant la retransmission des paquets de A, par rapport à son comportement concernant la retransmission des paquets des autres nœuds. Nous favorisons les informations relatives à la retransmission des paquets de A ; ce choix est justifié par le fait que B peut refuser de relayer les paquets d'un autre nœud C s'il s'est mal comporté avec lui ; par conséquent le nœud B sera mal jugé. Avec une petite valeur de (1- α), nous allons donc minimiser l'influence de ces mauvais jugements sur la note de réputation DRN.

Un nœud peut être isolé par ses voisins s'il a un mauvais comportement. Cependant, il peut toujours bénéficier des services du réseau s'il se déplace vers d'autres emplacements. Pour éviter cela, nous prévoyons un mécanisme qui permet de diffuser ce type d'information à travers le réseau et que nous appelons note de *réputation externe.*

6.2.2.2. La note de réputation externe

Comme expliqué précédemment, notre système ne se limite pas à la note DRN pour l'établissement des relations de confiance, mais prend en considération aussi les recommandations des autres nœuds. Pour cela, nous proposons un autre moyen d'évaluation de la confiance dit « note de réputation externe ». Cette dernière est calculée à partir des notes de réputations reçues des autres nœuds du réseau. Notons que ce calcul ne tient compte qu'uniquement des recommandations des nœuds de confiance et celles reçues durant la période de collecte d'informations. Cette note est calculée de la façon suivante :

$$\boldsymbol{Next(i,j)} = \frac{\mathbf{1}}{|\boldsymbol{N(j)}|} \sum_{\boldsymbol{k \epsilon N(j)}} \boldsymbol{DRN(k,j)} \times \boldsymbol{RN(i,k)} \qquad (2)$$

N(j) représente l'ensemble des nœuds faisant partie de la table de confiance du nœud i qui ont une note de confiance concernant le nœud j. Les nœuds appartenant à N(j) doivent avoir une note de réputation RN ≥seuil au niveau de la table de confiance du nœud i.

RN (i, k) représente la note de réputation du nœud k au niveau du nœud i ; cette note doit être supérieure ou égal à un seuil dans le but de tenir compte uniquement des recommandations des nœuds de confiance.

6.2.2.3. La note de réputation finale

La note de réputation directe et la note de réputation externes nous permettront de calculer la note de réputation finale qui sera le moyen d'évaluation de la confiance que porte un nœud i envers un nœud j. La note de réputation finale est alors donnée par la formule suivante:

$$\boldsymbol{RN_t(i,j) = \frac{\alpha RN_{t-1} + [\beta DRN(i,j) + (1-\beta) Next(i,j)]}{\alpha + 1}} \qquad (3)$$

Tel que : $\boldsymbol{0 < \alpha < 0.5}$ *et* $\boldsymbol{0.5 < \beta < 1}$

$\boldsymbol{RN_t(i,j)}$ est la note de réputation finale à l'instant t.

Nous favorisons la note de réputation directe par rapport à la note externe pour des mesures de précaution. Un nœud peut avoir un bon comportement puis basculer en ayant un mauvais comportement suite à un évènement spécifique comme par exemple : l'instabilité du support sans fil, la congestion, le débordement de la file d'attente de transmission, le manque de ressource énergétique, etc. Pour éviter que ce cas de figure baisse trop la note de réputation, nous intégrons dans la formule de la note de réputation finale, la note de réputation précédente. La valeur de α est entre 0 et 0.5 pour obliger le système à « oublier » régulièrement les anciennes surveillances.

La note de réputation directe d'un nœud inactif régresse avec le temps même s'il avait un bon comportement avant sa déconnection du réseau, cela est réalisé dans le but d'éviter à ce nœud de profiter longtemps de ses anciens bons comportements, sa note de réputation diminue jusqu'à avoir la valeur 0.5 qui représente un comportement neutre. Notons ainsi que lorsqu'un nœud vient de s'insérer pour la première fois dans le réseau, sa note de réputation directe DRN aura la valeur 1 et sera considéré donc comme un nœud de confiance.

Remarquons que toutes les notes de confiance (NC) sont évaluées dans l'intervalle [0,1] : ∀ x, y deux nœuds, 0 <= NC(x, y) <= 1.

6.2.3. La punition

La décision de punition (ou pénalisation) d'un nœud vis à vis du nœud qu'il surveille sera effectuée de façon aléatoire avec une probabilité de P=1-RN (par tirage aléatoire).

Si le nœud surveillé s'avère être puni, alors le temps de punition sera d'un temps t proportionnel à la valeur de P.

Pendant ce temps t, le nœud en question sera mis dans une liste noire (Black List). Punir un nœud, revient à « empêcher » les agents *Ant* de notre protocole de routage de « transiter » par le nœud puni durant toute la période de pénalisation t ; cela aura comme conséquence que la table de routage de ce nœud ne sera pas mise à jour et il va y avoir de moins en moins de routes établie vers ces nœuds si ce dernier à la même réputation vis à vis de ses autres voisins. Ainsi, pour éviter aux nœuds punis d'établir des routes avec leurs propres agents, chaque agent provenant d'un nœud égoïste sera supprimé par les nœuds récepteurs. En cours du temps, ce nœud sera donc isolé du réseau. Le système de punition que nous proposons oblige donc les nœuds à collaborer s'ils souhaitent toujours bénéficier des services du réseau. Apres l'expiration de la durée de punition, le nœud puni deviendra opérationnel (en le supprimant de la liste noire) et pourra donc bénéficier des opérations du réseau, histoire de donner une « autre chance » à ce nœud.

Le système de réputation proposé, nous oblige à effectuer quelques modifications sur le fonctionnement de notre protocole de routage. Par exemple: Lorsqu'un agent doit décider du prochain nœud à visiter, il le fait de façon aléatoire à partir de sa la table de voisins et sa table de phéromones (fonctionnement initial du protocole), mais aussi de la table de réputation de ses nœuds voisins.

Lorsque un nœud décide de punir son voisin, il génère un agent de type *RectifierAnt* comme si que le lien de voisinage avec ce nœud est réellement rompu, cet agent informera les autres du mauvais comportement de ce voisin. Chaque nœud recevant un agent *RectifierAnt* procèdera de la même façon que s'il avait reçu un agent *RectifierAnt* ordinaire.

Pour intégrer notre mécanisme de renforcement de la coopération, nous avons ajouté d'autres structures de données et modifié certaines qui existaient déjà. Dans ce qui suit, nous expliquons le contenu d'une nouvelle table qu'on a intégré dans notre protocole de routage. Les entrées de cette table sont définies par :

<Id_noeud><VoisinsId_noeud><Input><Forward><RNRec><DRN><Next><NR>
<Pénalisé><DuréeP>

où *<Id_noeud>* est l'adresse du nœud pour lequel le nœud courant a calculé une note de réputation RN, *<VoisinsId_noeud>* contient les voisins du nœud *<Id_noeud>* pour lesquels il recommande une note DRN, *<Input>* et *<Forward>*,contiennent les variables expliquées dans le module collecte d'informations, *<RNRec>* contient les notes de réputation directes qu'attribut le nœud *<Id_noeud>* à

ses voisins, *<DRN>* est la note de réputation directe attribuée par le nœud courant au nœud *<id-nœud>*, *<Next>* est la note de réputation externe associée au nœud *< Id_noeud >*, *<NR>* est la note de réputation finale attribuée par le nœud courant au nœud < Id_noeud >, <Pénalisé> est un booléen qui définit si le nœud < Id_noeud > est pénalisé ou pas et <DuréeP> est le temps de pénalisation de < Id_noeud > s'il est puni.

6.2.4. Simulation

Rappelons que notre système de renforcement de la coopération entre les nœuds est intégré dans le protocole de routage que nous avons proposé dans le chapitre précédent. A chaque intervalle de temps t, un nœud déclenche un agent local qui a comme rôle le calcul des notes de réputation selon les informations récoltées ; l'agent passera par la suite aux calculs de pénalisation. Toutes ces informations sont stockées dans la table de réputation du nœud. La figure 6-3 donne une idée sur le fonctionnement de notre système.

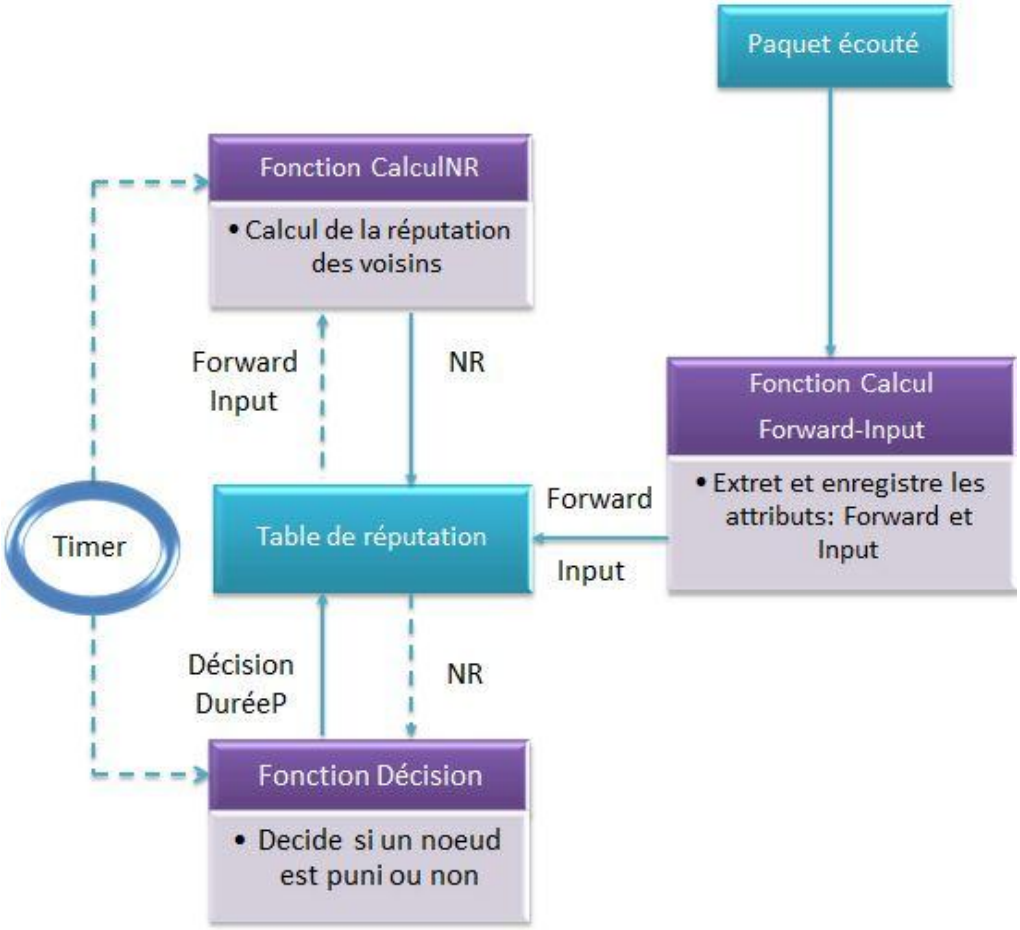


Figure 6-3- Fonctionnement global de notre système de réputation

La note de réputation finale que nous avons implémentée est celle qui prend en considération seulement les observations des nœuds voisins d'un nœud donné, à savoir :

$RN(i,j) = DRN(i,j)$.

Nous prévoyons comme perspective, l'implémentation de la note de réputation externe ($Next(i,j)$) et d'en tenir compte lors du calcul de la note de réputation finale.

La durée de pénalisation est calculée selon une fonction proportionnelle à p tel que $penalisation(p) = kp$ (où k est un réel positif).

Nos tests de simulation se basent sur l'étude du comportement d'un nœud N (choisi au hasard dans le réseau) ainsi que celui de ses voisins dans les deux cas suivants :

- Lorsque le nœud N est un nœud non égoïste.
- Lorsque le nœud N est égoïste.

Nous avons analysé la performance de notre système de renforcement de la coopération suivant les paramètres suivants :

- Le temps de simulation: la durée de simulation varie de 30 s à 200 s, le nombre de nœuds est stabilisé à 80 et le nombre de paquets envoyés est fixé à 500.

Dans toutes les simulations : $\alpha = 0.5$ et $k = 100$

Les métriques étudiées sont les suivantes :

Le nombre de nœuds que le nœud surveillé atteint : c'est le nombre de nœuds qui ont le nœud surveillé comme voisin et à qui ils ont confiance. Cette métrique permet de montrer si les nœuds du réseau prennent en considération la dimension confiance lorsqu'ils choisissent des routes.

Le nombre de voisins **dans les routes d'un nœud surveillé** : c'est le nombre de voisins qui existent dans la table de routage du nœud surveillé. Cette métrique nous informera sur l'état de la coopération du nœud surveillé avec les autres nœuds.

6.2.4.1. Graphes 1 : Le nombre ***de nœuds atteints par le nœud surveillé***

Les figures 6-4 et 6-5 représentent la variation du nombre de nœuds que le nœud surveillé atteint en fonction du temps. La durée de simulation dans la première figure est de 100s alors que dans la deuxième, elle est de 200s.

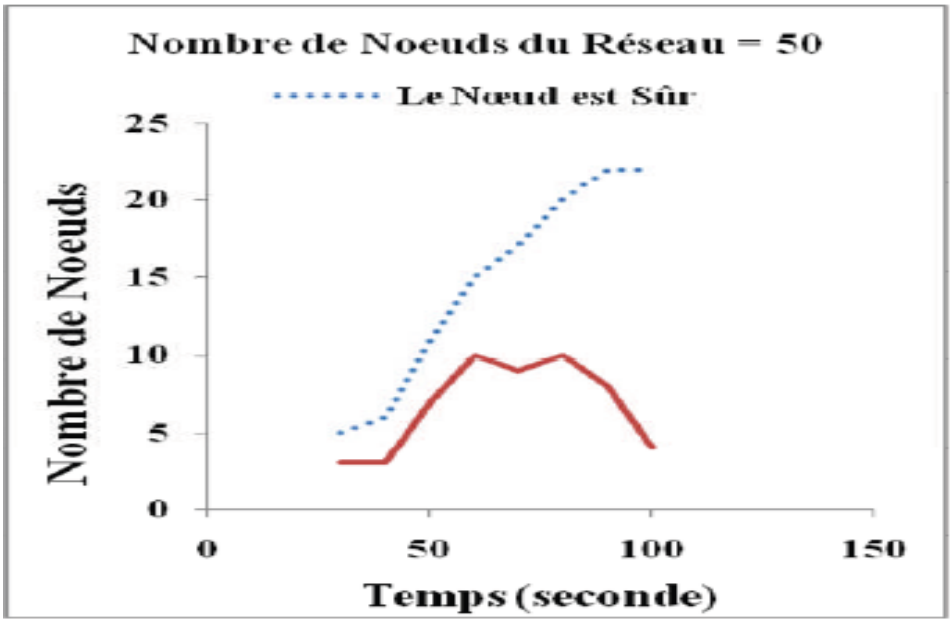


Figure 6-4- Le nombre de nœuds que le nœud surveillé atteint en fonction du temps (durée de simulation 100 secondes)

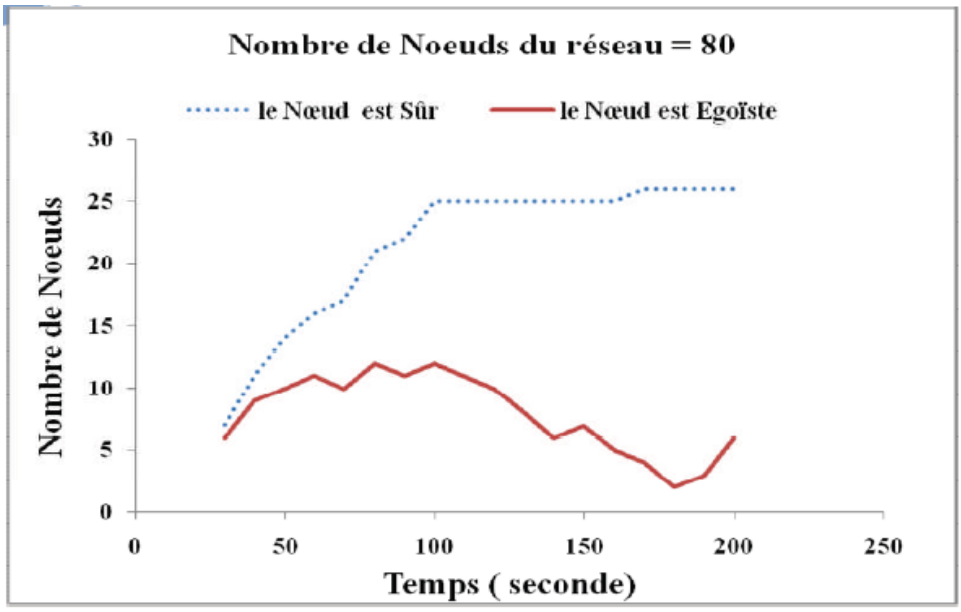


Figure 6-5- Le nombre de nœuds que le nœud surveillé atteint en fonction du temps (durée de simulation 200 secondes)

Nous remarquons que lorsque le nœud surveillé N est égoïste, le nombre de nœuds que N peut atteindre dans le réseau diminue à un certain temps, contrairement au cas où N est un nœud non égoïste. Cela veut dire que certains voisins du nœud surveillé ont déjà puni ce nœud.

Nous remarquons aussi qu'après un certain temps, le nombre de nœuds atteint par le nœud égoïste commence à augmenter (cf. figure 6-5), cela est dû au mécanisme de seconde chance implémenté dans notre système de réputation où après l'expiration de la durée de pénalisation, le nœud N redeviendra fonctionnel pour les voisins qui l'avaient puni.

D'après ces deux figures, nous constatons l'effet de notre système de renforcement de la coopération sur les nœuds du réseau puisque ces derniers ne coopèrent qu'avec les nœuds non égoïstes. Les autres nœuds seront pénalisés et ne bénéficient pas des services du réseau (moins de routes construites).

*6.2.4.2. Graphes 2 : **Le nombre de voisin dans les routes d'un nœud surveillé***

Les deux prochaines figures ont comme but de donner une idée sur les activités du nœud surveillé au sein du réseau. On mesure dans cette partie, le nombre de voisins appartenant à la table de routage du nœud surveillé.

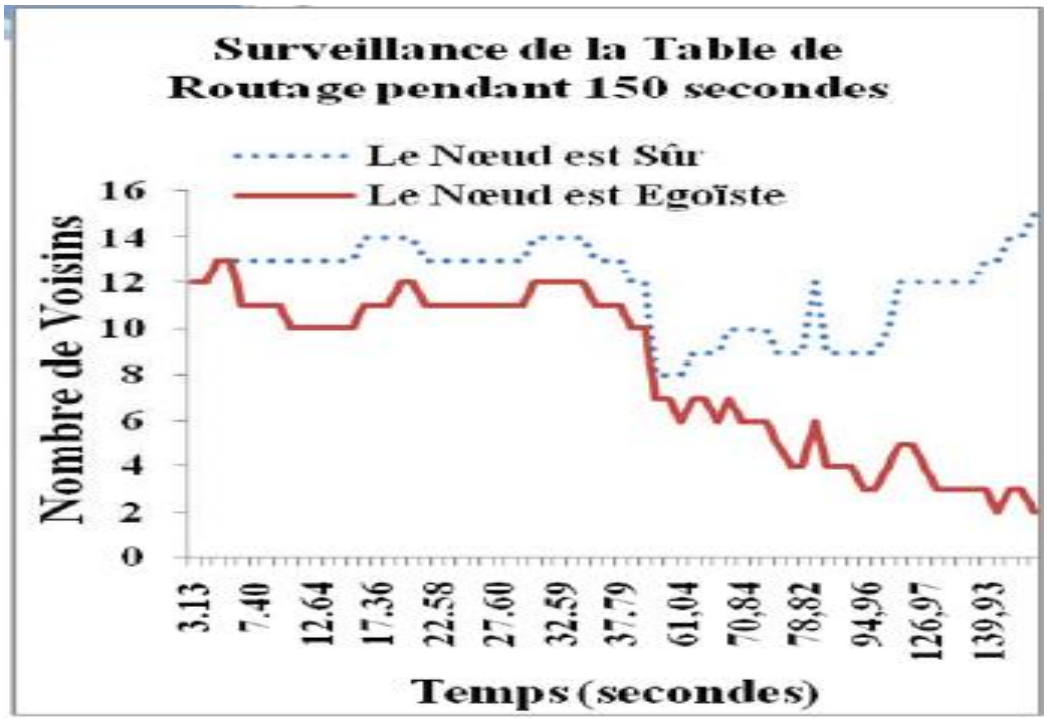


Figure 6-6- La variation du nombre de voisins dans la table de routage du nœud surveillé en fonction du temps (durée de simulation 150 secondes)

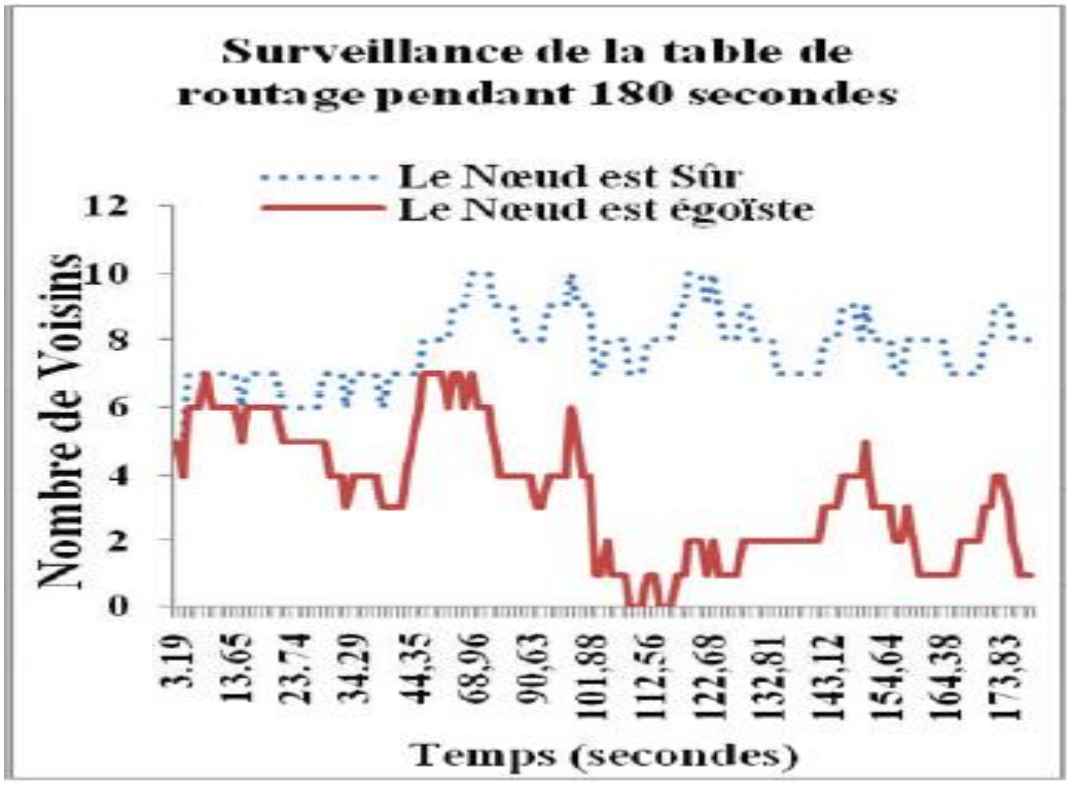


Figure 6-7- La variation du nombre de voisins dans la table de routage du nœud surveillé en fonction du temps (durée de simulation 180 secondes)

Les figures 6-6 et 6-7 représentent la variation du nombre de voisins dans la table de routage du nœud surveillé en fonction du temps. Remarquons qu'au début de la simulation, le nombre de voisins dans cette table contient la même valeur lorsque N est égoïste et lorsqu'il est confiant. Au cours du temps, ce nombre commence à diminuer dans le cas ou N est égoïste jusqu'à atteindre la valeur 0 (dans la figure 6-7). Cette valeur montre que tous les voisins du nœud l'ont puni à cause de son mauvais comportement. Le nœud N est donc isolé du réseau et ne peut même pas envoyer ses propres paquets de données ; par conséquence il ne pourra bénéficier des services du réseau. Apres un certain moment, le nombre de voisins dans la table de routage du nœud N commence à augmenter, ce qui montre que ce nœud devient de nouveau fonctionnel. Cela s'explique par le fait que certaines durées de pénalisations de quelques voisins de N ont expirés et que les nœuds concernés décident donc de router de nouveau les paquets de N.

Nous constatons à partir de ces simulations, que notre système de renforcement de la coopération entre les nœuds du réseau permet d'écarter les nœuds égoïstes qui ne souhaitent pas router les paquets des autres nœuds. Afin d'éviter de punir définitivement un nœud, nous avons implémenté un mécanisme qui permet de donner une second chance à un nœud qui s'est mal comporté au cas où il changerait de comportement.

6.3. Conclusion

Nous avons proposé dans ce chapitre, un protocole de renforcement de la coopération entre les nœuds du réseau, ce système est implémenté dans le protocole de routage que nous avons proposé et présenté dans le chapitre précédent. Chaque nœud surveille le trafic de ses voisins et leur attribut une note de réputation chacun suite aux informations récoltées à travers cette surveillance. Cette note sera diffusée dans le réseau pour éviter à un nœud qui s'est mal comporté de bénéficier toujours du réseau en se déplaçant vers d'autres régions. Pour calculer une note de réputation finale, un nœud X prend en considération en plus des résultats de ses observations, les recommandations des autres nœuds « à condition que ces nœuds soient confiants pour X ». Pour chaque nœud voisin figurant dans sa table de routage, X calcule une note de réputation globale qui lui permet de choisir les routes les plus sure lorsqu'il souhaite envoyer un paquet de données. En fonction de cette note de réputation globale, sera prise une décision probabiliste afin de punir les nœuds qui ne coopèrent pas (qui ne routent pas les paquets des autres). Punir un nœud, revient à ne pas mettre à jours sa table de routage en refusant aux agents au passage de passer par lui, à ne pas prendre en considération les agents qui proviennent de ce nœud et à ne pas établir des routes vers lui. Le nœud puni se retrouvera isolé du réseau vu qu'il ne possède aucune route valide, cela oblige le nœud à collaborer dans le futur s'il veut bénéficier des services du réseau. Si ce nœud souhaite devenir confiant, notre système lui accorde une deuxième chance après l'expiration de la durée de pénalisation.

Le mécanisme que nous avons proposé dans cette partie a augmenté considérablement les performances de notre protocole de routage. Les résultats de simulation montrent réellement que les nœuds qui se comportent mal avec les autres seront effectivement punis voir isolés carrément du réseau.

Comme perspectives, nous comptons améliorer ce protocole en implémentant la note de réputation globale (directe et indirecte que nous avons donné plus haut) qui permet de tenir compte de la mobilité des nœuds égoïstes.

Chapitre 7

7. Conclusion et perspectives.

L'évolution rapide des technologies d'internet a permis l'émergence de nouveaux types de réseaux très dynamiques avec des architectures fortement décentralisées et dont les services sont organisés de manière autonome. Ces spécificités ont un réel avantage : le déploiement et la mise en place rapide et peu couteux de ce type de réseaux. Mais en contrepartie, ces réseaux peuvent engendrer des difficultés lors de la mise en œuvre de certains services tels que le routage et la sécurité en général. Dans le cadre de cette thèse, nous avons pris comme cas d'étude le maintien de l'anonymat dans les réseaux P2P et le routage dans les réseaux ad hoc.

La première partie de la thèse a porté sur les aspects de maintien de l'anonymat des communicants ainsi que la confidentialité des ressources échangées dans les réseaux P2P. Ces propriétés sont de plus en plus revendiquées par les utilisateurs d'internet en général et ceux de réseaux P2P en particulier. Nous avons réalisé une étude et une synthèse des travaux existants dans le domaine ce qui nous a permis de mettre en avant un certain nombre de problèmes : Certains systèmes assurent un haut degré de confidentialité des échanges et de maintien de l'anonymat des communicants mais souffrent de problèmes de performances, de robustesse et de difficulté d'utilisation. D'autres systèmes sont plus performants mais présentent certains risques pour le maintien de l'anonymat des demandeurs et/ou des fournisseurs de ressources dans certaines situations. D'autres vulnérabilités sont liées à la confidentialité des échanges dus essentiellement à la difficulté d'échanger des clés de chiffrement au sein d'un environnement distribué. Pour répondre en partie à ces problématiques, nous avons proposé un nouveau protocole de partage de ressources de type P2P non structuré. Notre protocole s'inspire du concept des systèmes multi agents mobiles et tente de minimiser l'impact de l'anonymat et de la confidentialité des échanges sur les performances et la robustesse. Les résultats obtenus avec une première version de notre système, confirment nos prévisions.

Nous avons mis en œuvre notre protocole P2P à l'aide du simulateur PeerSeem, réalisé une analyse de sécurité et comparé ses performances avec celles du protocole Gnutella : Les résultats obtenus sont très prometteurs. Nous envisageons par la suite de déployer notre protocole dans un réseau réel et à accès publique tel que internet.

Dans les réseaux ad hoc, on s'est intéressé à la problématique du routage. Apres avoir fait une étude et une synthèse des différents protocoles de routage existants ainsi que les problèmes rencontrés par ces protocoles, nous avons proposé un nouveau protocole de routage hybride. Le protocole

fonctionne à la manière d'un système Multi Agents mobiles à l'aide d'un algorithme distribué original pour le calcul des routes. Les résultats de simulation de notre protocole sont très encourageants comparés aux protocoles de routage cités dans les autres travaux (AODV, DSDV et AnthotNet). Notre protocole ne nécessite aucun mécanisme d'inondation (broadcast) du réseau lors de la recherche de routes, ce qui a pour effet de réduire considérablement les délais de communication, les pertes de paquets dans le réseau et la surcharge du réseau. Notre protocole n'établit pas nécessairement le meilleur chemin entre deux nœuds donnés, en revanche il établit plusieurs chemins vers la même destination. Cette spécificité nous a permis d'avoir un protocole de routage robuste et d'envisager éventuellement d'autres critères de sélection d'une route en intégrant par exemple d'autres métriques pour la qualité de service (bande passante, surcharge réseau, nœuds malveillants, nœuds égoïstes, etc.).

Afin d'éviter des dégradations considérables des performances de notre protocole de routage dû au comportement égoïstes de certains nœuds du réseau, nous avons doté notre protocole d'un mécanisme de renforcement de la coopération entre les nœuds basé sur le calcul de la réputation. Le mécanisme se déroule en trois phases : La première consiste à chaque nœud de faire une collecte d'informations sur le comportement de ses voisins quant au routage des paquets de contrôle, chaque nœud du réseau surveille le comportement de ses voisins puis leur affecte une note de réputation. Cette note associée à un tirage aléatoire va servir de base pour permettre au nœud de décider lesquels de ses voisins seront « punis » et seront écartés du réseau. Le fonctionnement de l'algorithme de routage est enrichi afin de permettre à certains nœuds de supprimer de leur table de routage les nœuds « punis » et empêcher ainsi que des Agents mobiles passent par ces derniers. La conséquence de cela est que les nœuds « punis » auront des tables de routage moins fréquemment mises à jour et par conséquent un accès au réseau plus difficile. Les résultats de simulation montrent réellement que le nouveau mécanismes intégré au protocole de routage réagit positivement face aux nœuds égoïstes en les écartant du réseau provisoirement puis en les réinsérant dans le réseau une fois le temps de « punition » écoulé et tout cela avec un comportement complètement autonome des différents nœuds du réseau.

Bibliographie

[AE03] L.Anderegg, S.Eidenbenz. Ad hoc-VCG: a Truthful and Cost-Efficient Routing Protocol for Mobile Ad hoc Networks with Selfish Agents. In Proceedings of 9th Annual International Conference on Mobile Computing and Networking, pages 245-259, 2003.

[AH99] P. Anelli & E. Horlait : NS-2: Principes de conception et d'utilisation, Version 1.3, 1999.

[ARAT12] Ikram Allam, Mohamed Amine Riahla, Boussad Ait Salem and Karim Tamine. A Cooperation-Enforcement Protocol for a New Hybrid Routing Protocol for MANETS. In Proceedings of the Sixth International Conference on Sciences of Electronic, Technologies of Information and Telecommunications, 2012.

[ART12] Boussad Ait Salem, Mohamed Amine Riahla, Karim Tamine. A hybrid multiagent routing approach for wireless ad hoc networks. Wireless Networks 18(7): 837-845, 2012.

[BB02] S. Buchegger and J.-Y. Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing PDP, Las Palmas de Gran Canaria, Canary Island, Spain, January 9-11, 2002, pages 403-410. Euromicro, Jan 2002.

[BB03] S. Bansal and M. Baker. Observation-based Cooperation Enforcement in Ad Hoc Networks. In Proceedings of CoRR, 2003.

[BB04] S. Buchegger and J.-Y. Le Boudec. A robust reputation system for P2P and mobile ad hoc networks. In Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems, P2PEcon 2004, Cambridge, MA, US, June 4-5, 2004, pages 403–410. Harvard University Press, Jun 2004.

[BBo02] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the CONFIDANT protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks). In J. Hubaux, J. J. Garcia-LunaAceves, and D. Johnson, editors, Proceedings of the Third ACM International Symposium on Mobile ad hoc Networking and Computing, Lausanne, Switzerland, June 9 11, 2002, pages 226–236. ACM Press, June 2002.

[BBo04] S. Buchegger and J.-Y. Le Boudec. Self-policing mobile ad hoc networks. In Mohammad Ilyas and Imad Mahgoub, editors, Mobile Computing Handbook, pages 435–456. CRC Press, 2004.

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. From Natural to Artificial Swarm Intelligence. Oxford University Press, 1999.

[Ber03] Jem E. Berkes. Decentralized Peer-to-Peer Network Architecture: Gnutella and Freenet, University of Manitoba Winnipeg, Manitoba Canada, April 9, 2003.

[BLB02] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the confidant protocol. In MobiHoc ’02 : In Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, pages 226-236. ACM, 2002.

[Bou02] Imed Bouazizi. Ara - the ant-colony based routing algorithm for manets. In ICPPW ’02 : Proceedings of the 2002 International Conference on Parallel Processing Workshops, page 79. IEEE Computer Society, 2002.

[BS06] S. A. Baset, H. G. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In Proceedings of the 25th IEEE International Conference on Computer Communications, 2006.

[CC05] T. Chothia and K. Chatzikokolakis, A Survey of Anonymous Peer-to-Peer File-Sharing. In Proceedings of EUC Workshops, pages 744-755, 2005.

[CHMSW02] Ian Clarke, Theodore W. Hong, Scott G. Miller, Oskar Sandberg, Brandon Wiley. Protecting Free Expression Online with Freenet. IEEE Internet Computing 6(1): 40-49, 2002.

[Cho06] Tom Chothia. Analysing the mute anonymous file-sharing system using the pi-calculus. In FORTE2006, volume 4229 of LNCS, 2006.

[Cho07] Tom Chothia. Securing pseudo identities in an anonymous peer-to-peer file-sharing network. In Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm. Pages 279-282, 2007.

[CR06] C. De Cannière and C. Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In X. Lai and K. Chen, editors, Advances in Cryptology – ASIACRYPT 2006, volume 4284 of Lecture Notes in Computer Science, pages 1-20. Springer-Verlag, 2006.

[CSWH01] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: a distributed anonymous information storage and retrieval system. In Proceedings of Privacy Enhancing Technologies, 2001.

[CWC05] Nicolas Christin, Andreas S. Weigend, and John Chuang. Content availability, pollution and poisoning in _le sharing peer-to-peer networks. In EC '05 : Proceedings of the 6th ACM conference on Electronic commerce, pages 68-77, New York, NY, USA, 2005. ACM.

[DBDAJ01] Johnson David B, Maltz David A, and Broch Josh. Dsr : The dynamic source routing protocol for multi-hop wireless ad hoc networks. In In ad hoc Networking, edited by Charles E. Perkins, Chapter 5, pages 139–172. Addison- Wesley, 2001.

[DDC99] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. pages 11-32, 1999.

[DCDG04] G. Di Caro, F. Ducatelle, and L. M. Gambardella. Anthocnet : an ant-based hybrid routing algorithm for mobile ad hoc networks. In Proceedings of Parallel Problem Solving from Nature (PPSN VIII), 3242: 461-470, 2004.

[DD98] GianniDiCaro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 9:317-365, 1998.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, IT-22:644-654, 1976.

[DKB05] D. Djenouri, L. Khelladi, and A. N. Badache. A survey of security issues in mobile ad hoc and sensor networks. Communications Surveys & Tutorials, IEEE, 7(4):2-28, 2005.

[DMS04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second generation onion router. In Proceedings of USENIX Security Symposium, 2004.

[Dou02] John R. Douceur. The sybil attack. In IPTPS ’01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, pages 251-260. Springer-Verlag, 2002.

[DR01] J. Daemen and V. Rijmen. Algorithm alley: Rijndael : The Advanced Encryption Standard. Dr. Dobb’s Journal of Software Tools, 26(3):137-139, March 2001.

[Elg85] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans inf Theo, 31:469-472, 1985

[GDJ06] Saikat Guha, Neil Daswani , Ravi Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In IPTPS'06: The 5th International Workshop on Peer-to-Peer Systems, 2006.

[GFLM05] Di Caro Gianni, Ducatelle Frederick, and Gambardella LucaMaria. Anthocnet : An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transactions on Telecommunications, 16 :443-455, 2005.

[GNDGT03] Valerie Gayraud, Loutfi Nuaymi, Francis Dupont2, Sylvain Gombault, and Bruno Tharon. La sécurité dans les réseaux sans fil ad hoc. SSTIC03 12 Juin 2003.

[GSB02] Mesut Gunes, Udo Sorges, Imed Bouazizi. ARA-The Ant-Colony Based Routing Algorithm for MANET. International Workshop on ad hoc Networking (IWAHN 2002),Van-couver, British Columbia, Canada, August 18-21, 2002.

[Haa97] Zygmunt Haas. A new routing protocol for the reconfigurable wireless networks. In Proceedings of the 6th IEEE International Conference on Universal Personal Communications, IEEE ICUPC'97, October 12-16, 1997, San Diego, California, USA, volume 2, pages 562–566. IEEE, IEEE, 1997.

[Hei99] G. Heine. GSM networks: protocols, terminology, and implementation. Artech House, 1999.

[HJP02] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. Sead : Secure efficient distance vector routing for mobile wireless ad hoc networks. Mobile Computing Systems and Applications, IEEE Workshop on, 0:3, 2002.

[HPJ05] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. Wirel. Netw., 11(1-2) :21-38, 2005.

[HWK04] Q. He, D. Wu, and P. Khosla. SORI: A secure and objective reputation-based incentive scheme for ad-hoc networks. In Proceedings of the Third IEEE Wireless Communications and Networking Conference, WCNC 04, Atlanta, GA, US, March 21-25, 2004, volume 2, pages 825–830. IEEE Press, Mar 2004.

[IPKA07] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Leveraging BitTorrent for end host measurements. In Proceedings of PAM, 2007.

[IPKA10] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-preserving P2P data sharing with OneSwarm. In Proceedings of ACM SIGCOMM, September 2010.

[ISO89] ISO 7498-2:1989 Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture.

[KRD06] Sunil Kumar, Vineet S. Raghavan, and Jing Deng. Medium access control protocols for ad hoc wireless networks: A survey. ad hoc Netw., 4(3) :326-358, 2006.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In Proceedings of the 12th international conference on World Wide Web, WWW '03, pages 640-651, New York, NY, USA, 2003. ACM.

[LKR04] J. Liang, R. Kumar, and K. Ross. Understanding KaZaA. Technical report, Department of Computer and Information Science, Polytechnic University, Brooklyn, New York, USA, 2004.

[MM02] Pietro Michiardi and Refik Molva. Core : a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, pages 107–121. Kluwer, B.V., 2002.

[MM03] Sergio Marti and Hector G. Molina. Identity crisis: Anonymity vs. reputation in P2P systems. In 3rd International Conference on Peer-to-Peer Computing (P2P'03), pages 134-141, 2003.

[MM99] Pietro Michiardi, Refik Molva. “ad hoc networks security”. ST Journal of System Research, Volume 4, N°1, Mars 2003.

[Mut03] Vinod Muthusamy. An Introduction to Peer-to-Peer Networks. Presentation for MIE456 - Information Systems Infrastructure II, October 30, 2003.

[MVV96] A. Menezes, P. Van Oorschot, S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.

[NR06] Naoum Naoumov and Keith Ross. Exploiting p2p systems for ddos attacks. In InfoScale'06 : In Proceedings of the 1st international conference on Scalable information systems, page 47, New York, NY, USA, 2006. ACM.

[OLS03] Optimized link state routing protocol (olsr), 2003.

[OV03] Alberto Ornaghi and Marco Vallerie. Man In The Middle attacks demos. Blackhat conference, USA, 2003.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination sequenced distance-vector routing (dsdv) for mobile computers. In Proceedings of the conference on Communications architectures, protocols and applications, pages 234-244, 1994.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing, 2003.

[PCTS00] Adrian Perrig, Ran Canetti, J. D. Tygar, Dawn Song. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. IEEE Symposium on Security and Privacy, S&P, Oakland, California, USA. 2000, pages 56-73, 2000.

[PH02] P.Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation - CNDS, pages 193-204, 2002.

[PIAKV07] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent. In Proceedings of NSDI, 2007.

[PK00] Andreas Pfitzmann and Marit Köhntopp. Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology. In Hannes Federrath, editor, Designing Privacy Enhancing Technologies, volume 2009 of LNCS, pages 1-9, Berkeley, CA, USA, July 2000. Springer-Verlag, Berlin Germany.

[Pre05] Baptiste Prétre. Attacks on Peer-to-Peer Networks, Dept. of Computer Science Swiss Federal Institute of Technology (ETH) Zurich, Autumn 2005.

[Res11] E. Rescorla. Diffie-Hellman Key Agreement Method, RFC 2631, IETF Network Working Group, http://www.ietf.org/rfc/rfc2631.txt, 2011.

[RFHKS01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications (San Diego, California, United States). SIGCOMM '01. ACM, New York, NY, USA ©2001, Pages 161-172, 2001.

[Riv92] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), 1992. http://www.ietf.org/rfc/rfc1321.txt.

[RTG12] Mohamed Amine Riahla, Karim Tamine and Philippe Gaborit. A Protocol for File Sharing, Anonymous and Confidential, Adapted to P2P Networks. In Proceedings of the Sixth International Conference on Sciences of Electronic, Technologies of Information and Telecommunications, 2012.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2):120-126, 1978.

[SB88] M. E. Smid and D. K. Branstad. The Data Encryption Standard : Past and future. Proc. IEEE, 76(5):550–559, 1988.

[SCDR04] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. In EW 11: Proceedings of the 11th workshop on ACM SIGOPS European workshop, page 21, New York, NY, USA, 2004. ACM.

[Sch96] Bruce Schneier. Applied Cryptography, Second Edition, John Wiley & Sons 1996, ISBN: 2-84180-000-8.

[SDLSR02] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, E. Belding-Royer. "A secure routing protocol for ad hoc networks". In the 10th IEEE International Conference on Network Protocols (ICNP 02). Novembre 2002.

[SGG03] Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. Multimedia Systems, Vol. 9 Issue 2, Pages 170-184, August 2003.

[SGR97] P. Syverson, D. Goldschlag and M. Reed. Anonymous connections and onion routing. In Proceedings of the IEEE Symposium on Security and Privacy, 1997.

[SMKKB01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pages 149-160, New York, NY, USA, 2001. ACM.

[SNDW06] A. Singh, T. W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In Proceedings of the 25th IEEE International Conference on Computer Communications., pages 1-12, 2006.

[YLYLZ04] H. Yang, H. LUO, F. YE, S. LU, and L. ZHANG. Security in mobile ad hoc networks: Challenges and solutions. IEEE Wireless Communications, pages 38-47, 2004.

[YML02] H.Yang, X.Meng, and S.Lu. Self-Organized Network-Layer Security in Mobile ad hoc Networks. In Proceedings of the 1st ACM workshop on Wireless security, Pages 11-20, 2002.

[ZCY03] S.Zhong, J.Chen, and R .Yang. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In Proceedings of IEEE INFOCOM2003, April 2003.