# Sommaire

# Tables des figures

# Introduction

Aujourd'hui, de plus en plus d'applications multimédias voient le jour dans les réseaux informatiques en général et dans les réseaux ad hoc en particulier. Cette émergence de ces applications nécessite des traitements différenciés entre les flux qu'elles génèrent et les types de trafics traditionnels qui sont plus élastiques. Pour apporter des solutions à cette problématique, certains travaux ont récemment été menés dans le but d'optimiser le processus de routage en instaurant une certaine Qualité de Service (QoS : Quality of Service) dans les réseaux ad hoc. Les protocoles proposés dans ce cas prennent principalement en compte les problèmes liés à la fiabilité de la transmission ou à la dynamicité de l'environnement, occultant très souvent une évaluation précise des ressources restantes à travers le réseau.

La plupart des protocoles de qualité de service pour les réseaux supposent que cette quantité de ressources disponibles est connue ou facile à estimer. Or, les contraintes dans les réseaux ad hoc complexifient cette estimation. Ce mémoire de DEA propose des mécanismes nécessaires basés sur les systèmes Multi Agents pour l'évaluation et la réservation des ressources disponibles dans un réseau ad hoc et faciliter ainsi le support de la qualité de service pour ces réseaux. Il s'agit plus précisément d'offrir aux applications des garanties sur la qualité des flux en tenant compte des contraintes liées à la nature des réseaux ad hoc (ressources limitées, medium partagé, forte mobilité, etc.). En effet, les agents d'un SMA, grâce à leurs attributs que sont l'autonomie, la communication, la coopération, la réactivité, l'apprentissage et l'adaptabilité, constituent un bon outil de modélisation de situations distribués et d'environnement dynamiques tels que les réseaux ad hoc et particulièrement le routage dans ces réseaux.

Aussi, afin d'assurer la disponibilité du service de Qualité de Service de notre proposition, il est nécessaire de ce prémunir des attaques qui ont vu le jour dans les réseaux ad hoc. En effet, le bon fonctionnement d'un protocole de routage avec qualité de service repose sur la disponibilité des ressources et aussi une bonne collaboration des nœuds que constitue l'environnement ad hoc.

Le contenu de ce mémoire est organisé en trois parties. La première partie présente un état de l'art de l'art sur les réseaux ad hoc. Dans la deuxième partie, nous posons la problématique de recherche liée au routage avec QoS dans les réseaux ad hoc. Et enfin la troisième partie relate nos deux mécanismes proposés à savoir le mécanisme de routage avec qualité de service basé sur les Systèmes Multi Agents et le mécanisme de sécurité pour assurer la disponibilité des ressources afin d'assurer une disponibilité de service de notre protocole de routage.

# PARTIE I : ETAT DE L'ART SUR LES RESEAUX MOBILES AD HOC

# *Chapitre 1: Le Routage dans les réseaux Ad hoc*

## 1.1 Présentation des réseaux mobiles Ad hoc

### 1.1.1 Définition

Un réseau mobile Ad hoc ou MANET (Mobile Ad hoc NETworks ,un groupe de travail issu de l'IETF [IET] qui travaille sur la normalisation des protocoles ad hoc sous IP) est une interconnexion d'unités mobiles ou nœuds capables d'échanger de l'information dont le seul moyen de communication est l'utilisation de leur interface sans fil, sans l'aide d'une infrastructure préexistante ou d'une administration centralisée [RFC 2501], [TAY]. De plus, le réseau doit pouvoir s'auto-configurer, sans aucune intervention humaine. Les nœuds du réseau doivent collaborer pour attribuer des adresses, pour s'entendre sur les fréquences radio à utiliser, etc.

La principale différence entre un réseau ad hoc et un réseau filaire se situe au niveau du rôle joué par les nœuds ou unités mobiles. Dans un réseau filaire, on a deux catégories d'entités, les terminaux et ceux assurant le routage. Dans un réseau ad hoc cette distinction n'existe pas, touts les nœuds collaborent et participent à l'acheminement des données.

Un réseau ad hoc peut être modélisé par un graphe:

$$G_t = (N_t, L_t) \quad (1)$$

Où

$N_t$ représente l'ensemble des nœuds (i.e. les unités mobiles) du réseau,

$L_t$ modélise l'ensemble des liens (ou connections) qui existent entre les nœuds.

Si $l = (u, v) \in L_t$ alors les nœuds u et v sont en mesure de communiquer directement à l'instant t.

La figure suivante représente un réseau ad hoc de 8 nœuds sous forme de graphe.

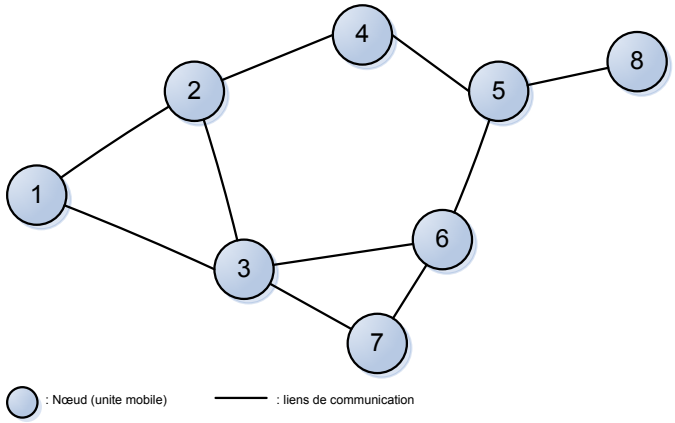



**Figure 1. modélisation d'un réseau ad hoc**

### 1.1.2 Caractéristiques des réseaux mobiles ad hoc

Un réseau mobile ad hoc est caractérisé par [RFC2501] :

- Une topologie dynamique causée d'une part par l'apparition et la disparition fréquente des unités mobiles et d'autre part par la mobilité des nœuds (voir figure 2).
- Un déploiement rapide ne nécessitant aucune infrastructure préexistence ou de câblages.
- Une contrainte d'énergie: les nœuds du réseau sont alimentés par des sources d'énergie autonomes. Cette énergie doit être optimisée afin de relayer les messages vers d'autres mobiles lors du routage.
- Une bande passante limitée: le médium radio est partagé pour la communication sans fil entre les différentes unités mobiles.
- Une sécurité limitée: le réseau ad hoc utilise des ondes hertziennes pour la transmission des messages entre deux nœuds. Ces messages peuvent être interceptés par un tiers nœud malveillant qui se connecte au réseau.
- L'absence d'infrastructure: les réseaux ad hoc se distingue des autres réseaux mobiles par la propriété d'absence d'infrastructure préexistante. Les unités mobiles sont responsables d'établir et de maintenir la connectivite du réseau d'une manière continue. Cette caractéristique constitue un frein dans la gestion et le control du réseau surtout en ce qui concerne l'apport en qualité de service.

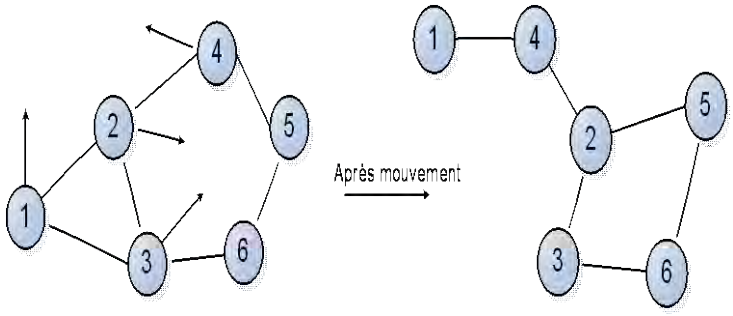



**Figure 2. Changement de topologie dans un réseau Ad hoc**

Ces avantages permettent aux réseaux ad hoc de connaitre un grand essor et, surtout, les rendent adaptés pour de nombreux situations ou le déploiement d'une infrastructure serait trop long, trop couteux ou tout simplement impossible.

L'utilisation des réseaux ad hoc est de deux natures:

- L'utilisation civile : déploiement lors des réunions, séminaires… mais aussi lors des catastrophes par les services de secours (séismes, raz de marée).
- Les applications militaires : déploiement des réseaux de capteurs, des troupes sur les champs de batailles.

Cependant, avec l'absence d'infrastructure d'une part, une bande passante faible et une très grande mobilité des nœuds d'autre part, certaines questions se posent sur la transmission des messages entre les différents nœuds du réseau.

"*Comment tracer et maintenir une route stable et optimale dans une topologie dynamique pour la transmission des messages entre deux nœuds?"*

## 1.2 Routage dans les réseaux mobiles Ad hoc

Le routage est une méthode d'acheminement des informations à la bonne destination dans un réseau d'interconnexion. Le problème classique du routage consiste à déterminer un acheminement optimal des paquets de messages à travers un réseau.

Pour les réseaux ad hoc mobiles, caractérisés par une absence d'infrastructure, chaque nœud est susceptible d'être mis à contribution pour participer au routage et pour retransmettre les paquets d'un nœud qui n'est pas en mesure d'atteindre sa destination; tout nœud joue ainsi le rôle de station et de routeur [TAY]. Le problème consiste à trouver l'investissement de moindre coût en capacités nominales et de réserves qui assure le routage du trafic nominal et garantit sa surveillance en cas de n'importe quelle panne de lien ou de nœud.

Afin de maintenir la mobilité des nœuds dans un réseau ad hoc, le Groupe MANET a fait une extension des protocoles de routage de l'IP fixe :

- Inondation
- Vecteur de distance
- Routage à la source
- Etat du lien

Ainsi, deux familles ont été formées à partir de la normalisation MANET : les protocoles réactifs et les protocoles proactifs. Cette classification est basée sur la manière de création et de maintenance des routes lors de l'acheminement des données.

### 1.2.1 Les protocoles proactifs

Dans ces types de protocoles, les routes sont établies à l'avance par l'échange périodique des tables de routage. Ces tables de routage dynamiques permettent de tracer la route optimale. Ils sont basés sur la même philosophie des protocoles de routage des réseaux fixes. Les deux principales méthodes utilisées sont:

- la méthode Etat de Lien ("Link State")
- la méthode du vecteur de Distance ("Distance Vector ")

Ces deux méthodes exigent une mise à jour périodique des données de routage qui doit être diffusée par les différents nœuds du réseau.

Nous allons décrire dans ce qui suit les protocoles les plus utilisés de cette classe.

### *A) Le protocole de routage DSDV (Destination Séquence Distance Vector)*

Le protocole DSDV s'inspire du protocole RIP [RFC1058] et est basé sur l'algorithme distribué de Bellman-ford en rajoutant quelques améliorations [PER].

Chaque station mobile maintient une table de routage qui contient:

- Toutes les destinations possibles
- Le nombre de nœuds (ou de saut) nécessaire pour atteindre la destination
- Le numéro de séquence (SN: Sequence Number) qui correspond à un nœud de destination.

Le numéro de séquence est utilisé pour différencier les anciennes routes des nouvelles, ce qui évite la formation de boucle de routage (routing loop).

Pour la mise à jour des tables de routage dans une topologie qui change fréquemment, chaque nœud transmet périodiquement sa table de routage à ces voisins directs. La mise à jour dépend donc de deux paramètres : Le temps, c'est à dire la période de transmission, et es événements (ou les déclencheurs). La mise à jour doit permettre à une unité mobile de pouvoir localiser, dans la plupart des cas, une autre unité du réseau.

- Un paquet de mise à jour contient :
  - Le nouveau numéro de séquence incrémenté, du nœud émetteur.
- Et pour chaque nouvelle route :
  - L'adresse de la destination.
  - Le nombre de nœuds (ou de sauts) séparant le nœud de la destination.
  - Le numéro de séquence (des données reçues de la destination) tel qu'il a été estampillé par la destination.

Le protocole DSDV a permis d'éliminer les problèmes de boucle de routage et de "counting to infinity" (inexistence de mécanisme pour déterminer quand est ce que le réseau doit arrêter l'incrémentation de la distance qui correspond à une destination donnée).

Le principal défaut de DSDV réside dans la convergence des tables de routage. Ce problème est causé par deux raisons, d'une part par la non synchronisation des échanges de vecteur de distance, et d'autre part par le fait qu'une route moins couteuse est remplacée dans la table de routage par une route plus couteuse, si et seulement si, le nœud ayant publié les deux routes est la même. En outre, le protocole DSDV utilise une mise à jour périodique et basée sur les événements, ce qui cause un contrôle excessif dans la communication.

### *B) Le protocole de routage OLSR (Optimized Link State Routing Protocol)*

Le protocole OLSR [RFC 3626] a été proposé afin de résoudre le problème de convergence des tables de routage. Il se base sur le protocole de routage OSPF (Open Shortest Path First) des réseaux filaires. Cette classe de protocole se distingue par le fait que les nœuds collectent des informations par

rapport à l'état des liens. Pour le protocole OLSR, cette collecte d'informations sur l'état des liens dans le voisinage est réalisée par chaque nœud grâce aux messages HELLO diffusés périodiquement par les 1-voisins (c'est-à-dire les voisins à un saut).

Ces messages contiennent :

- la liste des nœuds avec lesquels l'émetteur a des liens bidirectionnels,
- la liste des nœuds que l'émetteur peut entendre (ils entretiennent un lien unidirectionnel vers lui),
- la liste des nœuds que l'émetteur a choisis comme MPR.

Afin de disséminer ces informations, le protocole OLSR définit un sous-ensemble C de nœuds appelés relais-multi-points (MPR Multi Point Relay) qui est responsables de la diffusion des messages de contrôle. Ce sous-ensemble C est sélectionné par chaque nœud i parmi les 1-voisins de telle sorte que C couvre tous les nœuds pouvant être atteints en deux sauts par i. chaque nœud annonce périodiquement à ses voisins, les nœuds qui a été désignés comme MPRs, lesquels sont responsables de la diffusion dans le réseau de l'état des liens avec les nœuds voisins.

La diffusion de ces messages HELLO permet aux nœuds du réseau de stocker, dans leur table des voisins, une vision à deux sauts de leur voisinage et de calculer l'ensemble de leurs MPR. Cet ensemble est recalculé dès qu'un changement est détecté dans le voisinage à deux sauts.
La diffusion sur la totalité du réseau (via les MPR) de messages de contrôle de topologie (*TC messages*[1]) donne l'information topologique nécessaire au routage. Ces messages contiennent, pour chaque MPR, la liste des noeuds qui l'ont choisi. Grâce à ces messages, les nœuds peuvent maintenir une table de topologie, indiquant le dernier saut pour chaque destination.

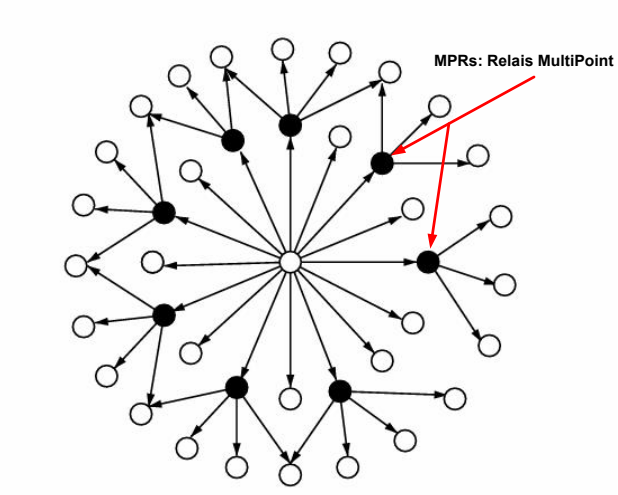



**Figure 3. Inondation par les MPRs**

[1] *Topology Control messages*

Le principal inconvénient des protocoles proactifs est leur coût en termes d'utilisation de la bande passante, qui est causé par l'échange excessif de messages de contrôle nécessaires à la maintenance des tables de routage. Cependant OLSR convient pour les grands réseaux grâce à son mécanisme de MPRs.

### 1.2.2 Les protocoles réactifs

Les protocoles de routage réactifs créent les tables de routage à la demande. Ainsi, ils génèrent à priori moins de message de contrôle que les protocoles proactifs. Lorsqu'un nœud désire transmettre, une procédure de découverte de route est lancée vers la destination. Le processus de recherche de route se décompose en deux étapes: la première correspond à la diffusion d'une requête de localisation de la destination, la deuxième consiste à collecter les réponses. Plusieurs protocoles ont été proposés selon différents contextes. Nous allons décrire dans ce qui suit quelques protocoles proactifs.

#### *A) Le protocole de routage DSR (Dynamic Source Routing)*

Le protocole DSR est basé sur la technique du routage à la source. La question principale est comment obtenir un itinéraire de la source pour une certaine destination. La procédure de recherche est déclenchée lorsque la source désire transmettre vers une destination pour laquelle elle ne détient pas de route valide [RFC 4728], [DSR].

L'opération de découverte de routes, permet à n'importe quel nœud du réseau ad hoc de découvrir dynamiquement un chemin vers un nœud quelconque du réseau. Un hôte initiateur de l'opération de découverte, diffuse un paquet requête de route (Route Request) qui identifie l'hôte cible (la destination). Si l'opération de découverte est réussie, l'hôte initiateur reçoit un paquet réponse de route (Route Replay) qui liste la séquence de nœuds à travers lesquels la destination peut être atteinte. En plus de l'adresse de l'initiateur, le paquet requête de route contient un champ enregistrement de route, dans lequel est accumulée la séquence des nœuds visités durant la propagation de la requête de route dans le réseau (voir la figure 4 (1)). Le paquet requête de route, contient aussi un identificateur unique de la requête. Dans le but de détecter les duplications de réceptions de la requête de route, chaque nœud maintient une liste de couples **[adresse source, identificateur de requête]**, des requêtes récemment reçues.

Lors de la réception d'un paquet de requête de route par un nœud **p** du réseau, le traitement suivant est effectué :

1. Dans le cas où le couple **[adresse source, identificateur de requête]** du paquet reçu, existe déjà dans la liste des requêtes récemment reçues le paquet est ignoré.
2. Dans le cas contraire, si l'adresse de **p** existe dans le champ enregistrement de route du paquet de la requête, le paquet est ignoré.
3. Sinon, si l'adresse de **p** est la même que l'adresse de la destination, alors l'enregistrement de route (contenu dans le paquet de la requête) contient le chemin à travers lequel le paquet de la requête est passé

avant d'atteindre le nœud **p**. Une copie de ce chemin est envoyée dans un paquet réponse de route à l'initiateur (voir la figure 4 (2)).

4. Sinon, l'adresse de **p** est rajoutée dans l'enregistrement de route du paquet reçu, et le paquet est rediffusé (voir la figure 4 (1)).

De cette manière, la requête de route est propagée dans le réseau, jusqu'à ce qu'elle atteigne l'hôte destination qui va répondre à la source. Le fait d'ignorer la requête, dans le cas où l'adresse du récepteur existe dans l'enregistrement de route, garantie que la propagation d'une unique copie de la requête ne peut pas se produire à travers des boucles de nœuds.

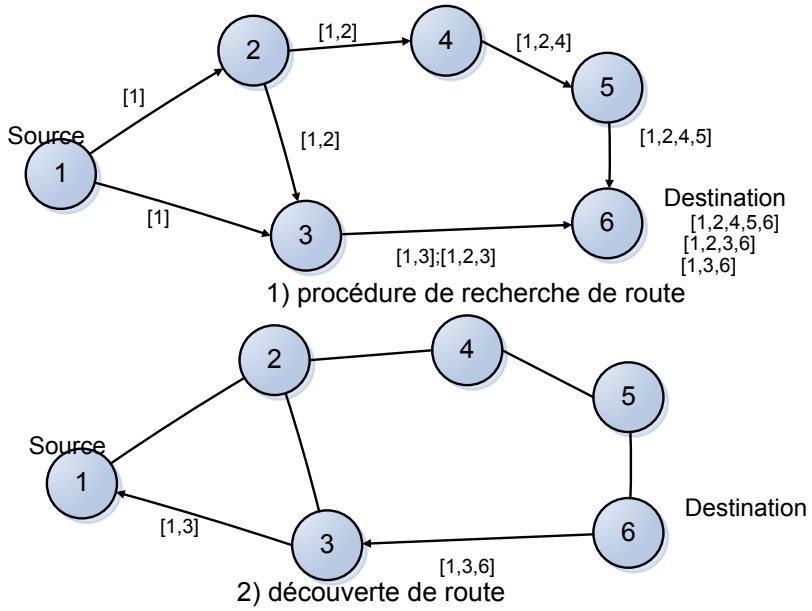



**Figure 4. Découverte de route dans DSR**

Afin de réduire le coût et la fréquence de la découverte de routes, chaque nœud garde les chemins appris à l'aide des paquets de réponses. Ces chemins sont utilisés jusqu'à ce qu'ils soient invalides.

Afin d'assurer la validité des chemins utilisés, le protocole DSR exécute une procédure de maintenance des routes. Quand un nœud détecte un problème fatal de transmission, à l'aide de sa couche de liaison, un message erreur de route (route error) est envoyé à l'émetteur (la source) original du paquet. Le message d'erreur contient l'adresse du nœud qui a détecté l'erreur et celle du nœud qui le suit dans le chemin. Lors de la réception du paquet erreur de route par l'hôte source, le nœud concerné par l'erreur est supprimé du chemin sauvegardé, et tous les chemins qui contiennent ce nœud sont tronqués à ce point là. Par la suite, une nouvelle opération de découverte de routes vers la destination, est initiée par l'émetteur.

Parmi les avantages de ce protocole, on peut noter le fait que les nœuds intermédiaires n'ont pas besoin de maintenir les informations de mise à jour pour envoyer les paquets de données puisque ces derniers contiennent les décisions de routage. En outre, on note aussi l'absence total de boucle de routage, car le chemin source – destination fait partie des paquets de données envoyés.

### *B) Le protocole de routage AODV (Ad-hoc On demand DistanceVector)*

Le protocole AODV (Ad hoc On demand Distance Vector) est une amélioration du protocole DSDV en y ajoutant le routage à la demande. [AODV], [RFC3561]. Ce protocole est basé sur l'utilisation des deux méthodes : découverte de routes et maintenance des routes.
De la même manière que le protocole DSR, AODV utilise les paquets Route Request, Route Replay et Route Error dans le but de créer un chemin de la source vers la destination. Cependant, il stocke une table de routage distribuée au niveau de chaque nœud. Une entrée de la table de routage contient essentiellement :

- L'adresse de la destination.
- Le nœud suivant.
- La distance en nombre de nœud (i.e. le nombre de nœud nécessaire pour atteindre la destination).
- Le numéro de séquence destination.
- Le temps d'expiration de l'entrée de la table.

Un envoi périodique de message HELLO est effectué afin d'assurer la maintenance des routes.

Dans ce protocole, les messages de control sont réduits, la présence de numéro de séquence permet de réduire d'éliminer les boucles de routage. Ce pendant comme le protocole DSR, il est lent du au processus de recherche de route avant la transmission.

## 1.3 Le routage multi agent dans les réseaux ad hoc : le protocole ARA

Pour comprendre le fonctionnement du protocole ARA, nous allons faire une description des systèmes multi agents.

Un système Multi- Agents est composé d'un ensemble d'agents en interaction, situés dans un environnement. Dans une modélisation par système multi agents, le système n'est plus représenté comme une entité monobloc composé de sous-systèmes bien identifiés et figés mais comme un ensemble d'entités autonomes en interaction appelés "**agent**" [FER].

Un agent est une "entité computationnelle", un programme informatique ou un robot, qui peut percevoir son environnement et agir sur lui de façon autonome. Il possède des ressources propres, des compétences, offre des services. Il est caractérisé par :

- son autonomie,
- sa réactivité et pro-activité,
- son interaction avec son environnement et les autres agents s'y trouvant,
- sa capacité de collaborer, de coordonner et de s'organiser avec les autres agents,
- sa capacité d'apprendre et de s'adapter à son environnement.

Plusieurs projets de recherche travaillent sur l'apport des SMA dans le contrôle des réseaux tels que les télécommunications, les réseaux de capteurs, les réseaux ad hoc, etc. Le protocole ARA fait partie de ces projets.

Le protocole ARA est basé sur l'algorithme "Ant Colony Optimization Meta-heuristic" qui est une classe particulière d'algorithme modélisée à partir du comportement des fourmis. Ces algorithmes sont des systèmes multi agents, qui se composent d'agents représentants le comportement des différentes fourmis, [SPA], [GUN].

L'idée principale est prise sur le comportement de vraies fourmis à la recherche de nourriture. Quand les fourmis sont sur un chemin à la recherche de nourriture, elles commencent à partir de leur nid et marchent vers la nourriture. Quand une fourmi atteint une intersection, elle doit choisir quelle branche prendre après. Tout en marchant, les fourmis déposent de la **phéromone** qui marque l'itinéraire pris.
Cette concentration de phéromone sur un certain chemin représente la fréquence de son utilisation. Cependant, avec le temps la concentration de phéromone diminue, due à l'effet de diffusion. Cette propriété est importante parce qu'elle permet l'interprétation du dynamisme dans le processus de recherche de chemin.

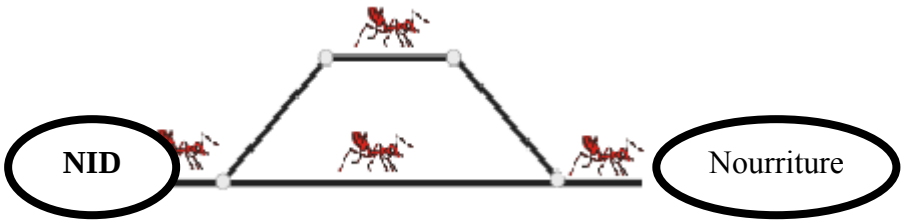



**Figure 5. Comportement des fournis à la recherche de nourriture**

La figure 5 montre un scénario de deux chemins quittant le nid vers l'emplacement de la nourriture, à l'intersection les premières fourmis choisissent aléatoirement la prochaine branche. Puisque le chemin en dessous est plus court que le supérieur, les fourmis qui prennent ce chemin atteindront en premier l'emplacement de la nourriture. Sur le chemin de retour vers le nid, les fourmis doivent encore choisir un chemin. Peu de temps après, la concentration de phéromone sur le chemin le plus court est plus élevé que sur le chemin le plus long. Parce que, les fourmis à l'aide du plus court chemin augmentent la concentration de phéromone plus rapidement. Le chemin le plus court sera ainsi identifié et par la suite toutes les fourmis emploieront celui-ci. Ce comportement peut être appliqué pour trouver le chemin le plus court dans les réseaux. En particulier, le comportement dynamique de cette méthode permet une adaptation élevée aux changements de la topologie du réseau mobile ad hoc, puisque dans ces réseaux l'existence des liens ne sont pas garanties dus aux changements fréquents de la topologie.

### 1.3.1 Algorithme d'optimisation méta-heuristique des colonies de fourmis

Soit G = (V, E) un graphe connecté avec n=|V|, l'algorithme d'optimisation méta-heuristic peut être utilisé pour trouver le plus court chemin entre la source Vs et la destination Vd du graphe G. la longueur du chemin est donnée par le nombre de nœuds (sauts) du chemin. Chaque lien e (i, j) ∈ E du graphe connectant les nœuds $V_i$ et $V_j$ à une valeur (coût) $\varphi_{i,j}$, représentant la quantité de phéromone artificielle, qui est modifiée par les fourmis quand ils visitent le nœud. La concentration de phéromone $\varphi_{i,j}$ est une indication sur l'utilisation du lien.

Une fourmi placée sur le nœud $V_i$ utilise une quantité de phéromone $\varphi_{i,j}$ du nœud $V_j \in N_i$ afin de calculer la probabilité pour que le nœud $V_j$ soit le prochain saut.
Ni= l'ensemble des voisins a un saut du nœud $V_i$.

$$p_{i,j} = \begin{cases} \frac{\varphi_{i,j}}{\sum_{j \in N_i} \varphi_{i,j}} & \text{if } j \in N_i \\ 0 & \text{if } j \notin N_i \end{cases}, \qquad \sum_{j \in N_i} p_{i,j} = 1, \qquad i \in [1, N]. \tag{2}$$

Durant le processus de recherche de route, les fourmis déposent la phéromone sur les liens. Dans "SACOMA", les fourmis déposent une quantité constante $\Delta\varphi$ de phéromone. Les fourmis changent la quantité de phéromone sur les liens e(Vi, Vj) quand il se déplace du nœud Vi vers le nœud Vj.

$$\varphi_{i,j} = \varphi_{i,j} + \Delta\varphi \tag{3}$$

Comme la vraie phéromone, la concentration artificielle diminue avec le temps empêchant ainsi une convergence rapide de la phéromone sur les liens. Dans SACOMA, ceci se produit exponentiellement par:

$$\varphi_{i,j}(t+\tau) = (1-q).\varphi_{i,j}(t), \qquad q \in (0,1] \tag{4}$$

### 1.3.2 Le protocole ARA pour les réseaux Ad hoc

L'algorithme de routage est très similaire aux autres approches de routage. Elle se déroule en trois phases.

#### *A) Phase de découverte de route*

Pendant cette phase, de nouvelles routes sont crées. Cette création de routes requiert l'utilisation de 2 agents *"Forward ANT"* et *"Backward ANT".* FANT est un agent qui établit le chemin de phéromone à partir de la source. En revanche BANT établit la route de phéromone à partir de la destination.
FANT est un paquet avec un numéro de séquence unique. Ainsi les nœuds peuvent distinguer les doublons sur la base du numéro de séquence et de

l'adresse source de FANT. Il est diffusé par l'émetteur et est relayé par ses voisins (voir figure 6).

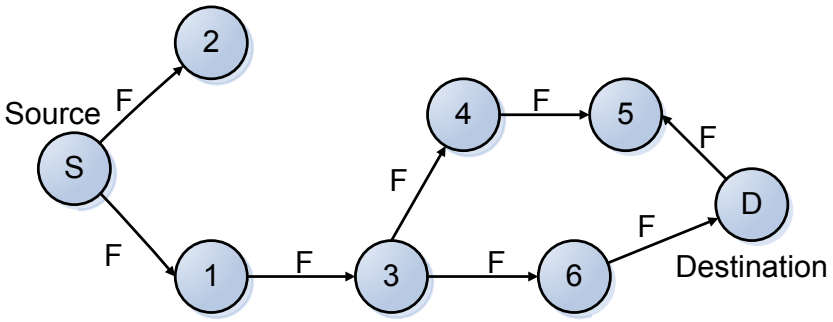



**Figure 6. Envoi requête de route par la source dans ARA**

Un nœud recevant un FANT en premier crée un enregistrement dans sa table de routage. Cette entrée est un triplet ***[@destination, next hop, φ]***. Le nœud interprète l'adresse source du FANT comme @ destination, l'adresse du nœud précédent comme next hop et calcule φ (valeur de phéromone) selon le nombre de saut utiliser par FANT pour atteindre le nœud. Ensuite le nœud relaie FANT à ses voisins, les doublons de FANT sont identifiés grâce au numéro de séquence et supprimés par les nœuds. Quand FANT atteint la destination, le nœud extrait l'information du FANT et le détruit. Ensuite, il crée un BANT et l'envoie au nœud source.

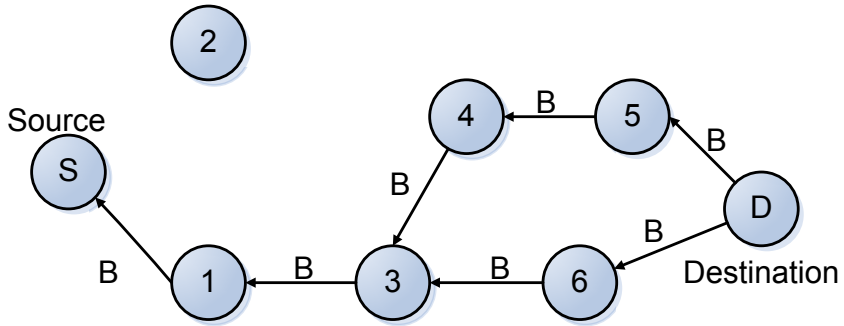



**Figure 7. Envoi réponse de route dans ARA par la destination**

BANT fait la même chose que le FANT, c.-à-d. établit la route pour le nœud de destination. Quant l'émetteur reçoit BANT du nœud de destination, la route est établie et la transmission des données peut commencer.

Les figures 6 et 7 illustrent schématiquement la phase de découverte de route. Dans cette exemple, le nœud 3 à deux choix de route, un passant par le nœud 4et un autre a travers le nœud 6. Dans ce cas ci, le FANT crée une seule voie de phéromone à partir de la source. Mais le BANT en crée deux à partir de la destination. Ce qui montre que la transmission, par chemin multiple, est supportée par ARA.

### *B) La maintenance de route*

La seconde phase de l'algorithme est appelée maintenance de route. Elle est responsable de l'amélioration des routes pendant la communication. Le protocole ARA n'a pas besoin d'aucun paquet spécial pour cette tache. Une fois que les FANT et BANT établissent les chemins de phéromone entre la source et les destinations, les paquets de données "transmis" sont utilisés pour assurer la maintenance. Par analogie à la nature, les chemins établis ne gardent pas leurs valeurs de phéromone de façons permanentes. Quand un nœud Vi relaie un paquet de données vers la destination à un nœud voisin Vj, ARA incrémente la valeur de phéromone de l'entrée (Vd, Vj, $\varphi$) par $\Delta\varphi$ c'est-à-dire le chemin vers la destination est renforcé par les données de paquets.

### *C) La phase de détection des erreurs de route*

La troisième et dernière phase du protocole ARA est la détection des erreurs de route qui sont principalement causés par la mobilité des nœuds et sont donc très fréquents dans les réseaux ad hoc. L'implémentation d'ARA intègre la fonction IEEE 802.11 [BRE] de la couche MAC. Ceci permet à ARA de détecter les erreurs de routes grâce à la non réception d'accusés (acknowledgement) sur la couche MAC. Si un nœud reçoit un message de ROUTE_ERROR pour un certain lien, il enregistre d'abord ce lien en mettant la valeur de phéromone à 0. Plus tard, le nœud recherche un lien alternatif dans sa table de routage. S'il y a un autre itinéraire à la destination elle enverra le paquet par l'intermédiaire de ce chemin. Autrement, le nœud informe ses voisins, espérant qu'ils pourront expédier le paquet vers la destination. Si le paquet n'atteint pas la destination, le nœud de source doit lancer un nouveau procédé de découverte d'itinéraire.

## 1.4 L'apport des Systèmes Multi Agent au routage dans les réseaux ad hoc

Ces différentes illustrations montrent l'apport considérable des SMA au routage dans les réseaux ad hoc. Nous en discutons en les reliant aux propriétés importantes des réseaux ad hoc mobiles.

- La topologie dynamique: cette propriété est la cause des mauvaises performances d'acheminement au niveau des algorithmes de routage dans les Manets. ARA repose sur la capacité des Systèmes multi agent à imiter le comportement des fourmis. Ceci permet une meilleure adaptation sur le changement de la topologie du réseau.

- L'approche locale: contrairement à d'autre approche de routage, le fonctionnement de ARA repose que sur les informations locales (tables de routage), c'est-à-dire pas d'échange de table de routage entre les voisins ou d'autres nœuds du réseau. Et ceci permet une meilleure réactivité et une meilleure gestion de l'état des liens avec le voisinage.

- Etat des liens: il est possible d'intégrer la qualité du lien dans le calcul de la concentration de phéromone particulièrement dans le processus

d'évaporation. Il est important de noter dans ce cas, que l'approche doit être modifiée de sorte que les nœuds puissent manipuler la concentration de phéromone indépendamment des fourmis, c'est-à-dire les paquets de données, car les nœuds doivent contrôler la quantité de phéromone sur les liens.

- Support de chemin multiple:c'est l'un des apports majeurs des SMA et plus particulièrement de A RA au routage dans les réseaux Ad hoc. Chaque nœud a un e table de routage contenant l'entrée de tous les voisins plus la concentration de phéromone. ***La règle de décision, pour choisir le prochain saut, est basée sur la concentration de phéromone sur le nœud courant, qui est donnée par chaque lien possible***. Ainsi l'approche soutient un routage multi-routes.

## Conclusion

Dans ce chapitre nous avons fait d'abord une pr ésentation du réseau mobile ad-hoc et ces caractéristiques, puis nous avons présenté quelques protocoles de routage proposés et nous avons terminé avec les protocoles de routage multi agent. Dans le chapitre qui suit, nous allons étudier la qualité de service dans les réseaux ad-hoc.

# *Chapitre 2: La qualité de service dans les réseaux Ad hoc*

Le terme qualité de service ou QoS a largement été utilisé pour définir de nombreux concepts sans toutefois converger vers un consensus. La QoS se définit comme un ensemble de garanties à assurer, par le réseau, pour le transport d'un trafic d'une source vers une destination [RFC2386]. Plus formellement, nous définissons la qualité de service comme l'ensemble des mécanismes mis en œuvre dans un réseau afin de satisfaire les besoins explicites des applications lors de l'acheminement des flux de données.

Les mécanismes classiques de qualité de service dans les réseaux filaires sont totalement inadaptés dans un environnement ad hoc. En effet, la plus part de ces méthodes reposent sur la connaissance d'informations précises quant à l'état du réseau (la bande passante utilisée, le délai, la gigue de phase, etc.). Dans un contexte sans fil, ces informations sont difficiles à évaluer notamment à cause des phénomènes propres aux transmissions sans fil (versatile du lien radio, interférences, atténuation du signal, etc.) et peuvent être amenées à varier très rapidement, en fonction de la mobilité.

## 2.1 La QoS dans les Ad hoc

Un état de l'art sur la QoS dans les réseaux ad hoc classifie les solutions de QoS en quatre grandes catégories [WU];

- les modèles de QoS définissent des architectures globales dans lesquelles des garanties peuvent être fournies.
- Les protocoles d'accès au medium cherchent à ajouter des fonctionnalités aux couches basses du modèle OSI afin de pouvoir offrir des garanties.
- Les protocoles de routage avec qualité de service recherchent les routes ayant suffisamment de ressources disponibles pour satisfaire une requête.
- Les protocoles de signalisation cherchent à offrir des mécanismes de réservation de ressources indépendants du protocole de routage sous-jacent.

Cette classification très souvent utilisée, permet d'identifier les différentes briques à mettre en œuvre pour assurer une certaine qualité de service. On distingue deux types d'approche [SAR]: les approches de qualité de service statique et les approches avec garanties quantitatives. Dans les approches de qualité de service statique, l'idée est d'offrir plus de ressources aux trafics prioritaires comparés aux trafics moins prioritaires, sans néanmoins assurer de garanties quantitatives. Dans un contexte ad hoc, l'ordre des priorités reste néanmoins difficile à respecter de part la difficulté de reporter des politiques locales de voisinage en voisinage. Pour des applications strictes comme la diffusion de vidéo, les approches avec garanties quantitatives nous semblent plus appropriées.

Parmi les approches avec garanties quantitatives, on distingue les approches à posteriori des approches à priori. Les approches à posteriori peuvent être basées sur n'importe quel protocole de routage et ne cherchent

qu'à réguler l'environnement afin d'offrir des garanties aux applications le nécessitant. A l'opposé, les approches à priori vont se baser sur un routage avec qualité de service. Le principe du routage avec qualité de service est de rechercher un chemin entre deux nœuds satisfaisant certaines contraintes. Plusieurs métriques peuvent être utilisées telles que le délai, la bande passante, la gigue ou encore le taux de perte.

Cette approche permet d'offrir un contrôle plus fin que celui offert par les approches a posteriori. Nous proposons de l'utiliser dans la suite de nos travaux.

Un très grand nombre de protocoles de routage de QoS ont été proposés [SAR], [WU]. On retrouve la dichotomie classique entre protocoles réactifs et proactifs. Ainsi, le routage avec qualité de service ajoute en général à ces protocoles de routages usuels un contrôle d'admission afin de sélectionner parmi les routes disponibles celles qui satisfont les contraintes spécifiées par l'application. Le contrôle d'admission est une étape très importante dans le routage QoS; il s'agit plus précisément de déterminer si les conditions du réseau permettent de transmettre les flux avec les garanties requises. L'admissibilité des flux est faite en fonction de deux paramètres:

- la quantité de ressources : elle est exigée par l'application. les applications doivent être en mesure de quantifier leur besoin en termes de ressources, comme par exemple la quantité de bande passante dont elles ont besoin. Ceci fait apparaître une spécificité du routage avec QoS: l'éligibilité d'une route doit être déterminée flux par flux et non plus destination par destination. Il est tout à fait envisageable d'utiliser des routes différentes pour des flux ayant des contraintes différentes. Le routage par flux permet par ailleurs d'assurer un contrôle plus fin des ressources du réseau.
- L'estimation des ressources résiduelles ou disponibles : elle doit être connue à priori avant l'envoi des flux QoS. C'est un point critique et beaucoup de protocoles de routage QoS se concentrent davantage sur la partie routage que sur la partie estimation des ressources résiduelles. Cette estimation des ressources est rendue plus complexe dans un contexte ad hoc où contrairement aux réseaux filaires, les liens entre mobiles sont versatiles, peu fiables, et partages inéquitablement.

Pour résumer, le routage QoS est un processus d'établissement et de maintenance des routes satisfaisant un ensemble de critères quant à la qualité de la transmission des données. Nous pensons que le point critique de ce processus reste l'estimation des ressources disponibles à travers le réseau car la précision du contrôle d'admission au niveau des nœuds dépend fortement de cette estimation.

## 2.2 Les techniques d'évaluation de la bande passante résiduelle

La bande passante dans les réseaux sans fil est très limitée et son partage est très compliqué, plus particulièrement dans les réseaux ad hoc. De plus, ce paramètre a un impact sur les autres paramètres du réseau comme par exemple le délai ou le taux de perte. Maîtriser les débits dans le réseau permet de limiter la congestion et par conséquent d'améliorer les délais de transmission

et les taux de pertes, par conséquent, les techniques d'évaluation de la bande passante résiduelle sont fondamentales.

La bande passante résiduelle ou disponible entre deux mobiles peut se définir comme le débit maximal qui peut être émis entre deux nœuds sans dégrader aucun des flux déjà présents dans le réseau. Cette notion est différente de la capacité qui représente juste le débit maximal d'émission entre deux mobiles.

Lors de la mise en place d'un protocole de réservation de bande passante, l'estimation de cette bande passante disponible doit être la plus précise possible quels que soient les trafics existants et de la topologie, afin de choisir les routes susceptibles d'offrir aux applications QoS les garanties désirées. Néanmoins, la bande passante résiduelle est une métrique difficilement quantifiable dans un environnement ad hoc multi saut pour les raisons suivantes:

- Lorsque n émetteurs à portée de communication sont en compétition pour l'accès au médium, la bande passante utilisable au niveau application par chaque émetteur n'est pas égale à la bande passante qu'obtiendrait un seul émetteur divisé par n. En effet, avant d'émettre, chaque nœud doit s'assurer que le canal radio a été libre pendant un certain temps aléatoire. Lorsque plusieurs émetteurs sont en concurrence, ces attentes ont lieu en parallèle, ce qui permet de réduire globalement le surcoût du protocole d'accès au médium. Toutefois, on ne peut augmenter indéfiniment le nombre d'émetteurs sans risquer de collisions. dans ce cas, les délais sont à nouveau allongés car le protocole 802.11 retransmet les paquets perdus.

- le médium étant partagé et la topologie multi saut, la perception de la bande passante utilisée et disponible est différente d'un mobile à l'autre. par conséquent, en plus de la bande passante qu'il consomme, un mobile doit avoir une estimation de la bande passante consommée par les mobiles voisins avec lesquels il partage le médium. Cela suppose une connaissance exacte du voisinage, ce qui n'est pas toujours le cas.

- le phénomène des stations cachées tend à diminuer le débit des communications. Cette situation provoque des collisions au niveau du récepteur qui voit son débit chuter.

Par conséquent, la bande passante résiduelle est une métrique complexe à évaluer car elle doit prendre en compte aussi bien les caractéristiques des transmissions voisines, que les phénomènes physiques dus au médium radio sous-jacents et à la technologie d'accès au médium (802.11 dans notre cas).

Cependant, les protocoles du groupe MANET [IET] ne prennent pas en compte tous les aspects d'un réseau Ad hoc à savoir l'évolutivité, la dynamicité etc. ces protocoles n'offrant pas de la QoS, beaucoup de travaux ont été menés aboutissant à l'apport QoS pour le routage dans les réseaux Ad hoc.

# *Chapitre 3: La sécurité dans les réseaux ad hoc*

Les réseaux mobiles ad hoc , de par leurs caractéristiques très particulières, tel la forte mobilité, le routage multi saut et l'absence d'infrastructure fixe, offrent davantage de v ulnérabilité que les réseaux classiques.

La sécurité des réseaux ad hoc traite les menaces qui s'étendent des attaques actives aux attaques passives. Les exigences fondamentales de sécurité indiquent le besoin de sécuriser l'exécution des protocoles de routage ad hoc de même que l a nécessité de mécanismes qui encouragent la coopération des nœuds pour le bon fonctionnement du r éseau. Les protocoles de routage sécurisés se fondent sur la disponibilité d'association de sécurité entre les nœuds. L'authentification des entités impliquées dans le routage et la vérification d'intégrité des messages d'acheminement requiert un s chéma de gestion de clé pour fournir aux parties concernées, dans l'exécution du protocole, un mécanisme cryptographique approprié. Les mécanismes de coopération exigent également l'authentification des nœuds afin d'empêcher les attaques de type "Spoofing". En outre, les informations sur le comportement des nœuds qui participent au fonctionnement du réseau peut être employés par des protocoles de routage sécurisés pour modifier le critère de choix des routes afin d'éviter des nœuds potentiellement malveillants, une technique qui va sous le nom de "watchdog - path rater".

## 3.1 Types d'attaques sur les réseaux ad hoc

Comme nous l'avons présenté précédemment, le bon fonctionnement d'un réseau ad-hoc repose sur la collaboration des nœuds et de leur volonté à effectuer leur tâche surtout de routage. Dès lors qu'un nœud ne respecte pas ces conditions, cela peut engendrer une défaillance du réseau.

Le terme "attaque" désigne une action visant à c ompromettre la confidentialité ou l'intégrité des informations transitant sur le réseau, ou, d'une manière générale, à altérer son bon fonctionnement [CHAO].Les attaques sur un réseau ad hoc peuvent être regroupées en deu x catégories: Les attaques passives qui sont une conséquence directe de l a technologie sans fil, où l'attaquant écoute le trafic sans influer sur le processus de r outage et les attaques actives (usurpation, rejeu, modification de données) lorsque les nœuds influent directement sur le processus de routage par injection de paquets dans le réseau [LEB], [ANJ].

### 3.1.1 Attaques passives

En raison de la nature même du medium d'accès, il est très facile pour un nœud quelconque, d'écouter des communications à l 'insu des participants. En effet, dans les réseaux ad hoc, les nœuds communiquent par l'interface radio avec un a ccès partagé du support. Une attaque triviale consiste alors pour un attaquant à s e mettre en mode écoute (*promiscuous listening*) pour capter tout ce qui se passe par l'interface radio et ainsi, analyser le trafic. Partant de ce

constat, suivant la signalisation utilisée par le protocole, un nœud peut extraire toutes sortes d'informations stratégiques (comme le contenu des données), mais aussi la connectivité du réseau, la localisation de certains nœuds, leurs adresses IP (Internet Protocol), MAC (Medium Access Control), etc.

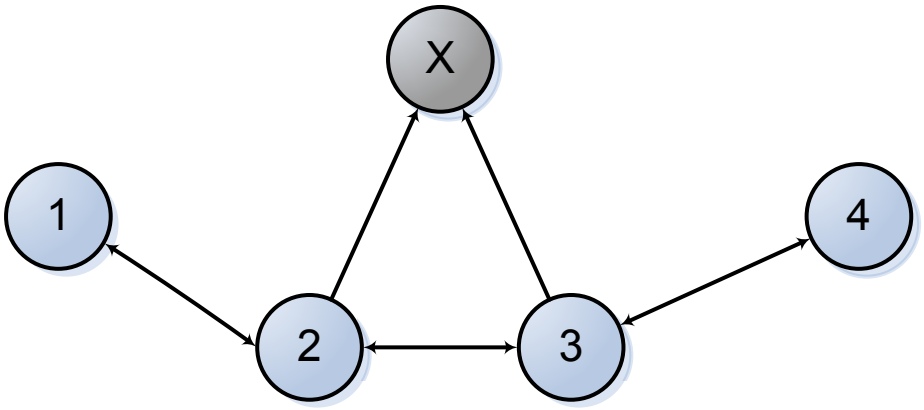



**Figure 8. illustration d'une attaque passive. Le nœud X capte discrètement les messages des autres nœuds sans émettre de signalisation**

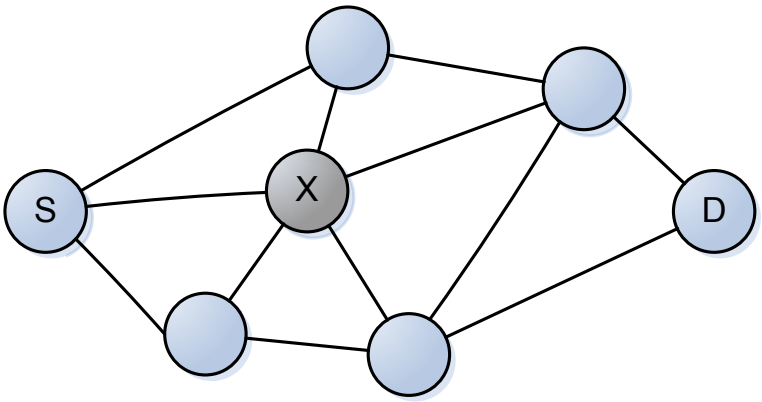
### 3.1.2 Attaques actives

Ces attaques peuvent être menées aux différents niveaux du modèle OSI. Ainsi, à l'instar des autres types de réseau sans fil, les réseaux ad hoc sont tout d'abord vulnérables à des attaques au niveau de la couche physique (par exemple brouillage) ou de la couche liaison (par exemple problème de la station cachée). Mais en raison de la particularité du routage dans ces réseaux, c'est la couche réseau qui apparaît la plus vulnérable. Nous allons dans ce qui suit présenter les attaques actives les plus en vue.

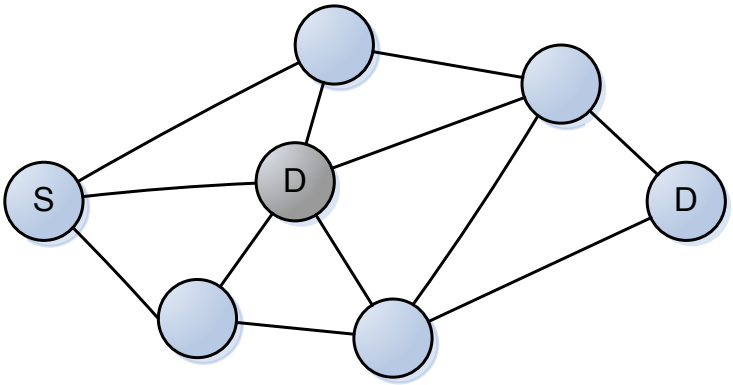
#### *A) Usurpation d'identité*

Cette attaque permet à un nœud de se faire passer pour un autre. Ainsi, il pourra masquer sa véritable identité et recevoir tout le trafic à destination de sa victime. En fonction du control d'accès mis en œuvre, celui-ci peut en ensuite favoriser la venue d'autres nœuds malicieux (par exemple sur des réseaux ad hoc fonctionnant par cooptation).

L'usurpation d'identité peut se limiter à l'envoi de message semblant venir d'un autre nœud (Spoofing) ou au rejeu de messages valides que l'attaquant aura enregistrés.




a) topologie du réseau initiale




b) topologie vue par le nœud S sous l'influence du nœud X qui usurpe l'identité du nœud D

**Figure 9. usurpation d'identité**

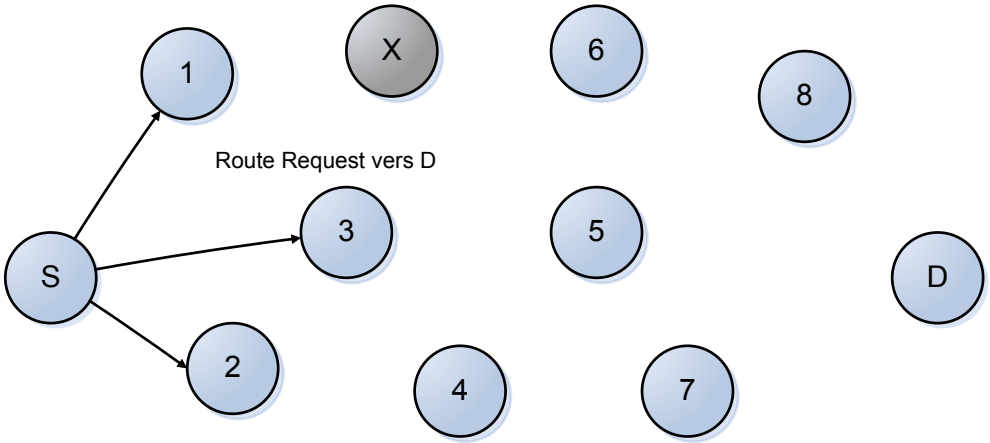
Dans cette figure, le nœud X utilise les éléments d'identification du nœud D (adresse IP, adresse MAC) pour communiquer avec les autres nœuds du réseau. Ainsi, le nœud S va voir le nœud D comme étant son voisin.

Cette attaque pouvant se dérouler à tous les niveaux dans les couches réseaux, elle peut être difficile à détecter. Cependant l'utilisation d'algorithmes d'authentification permet de réduire les risques d'une usurpation d'identité.

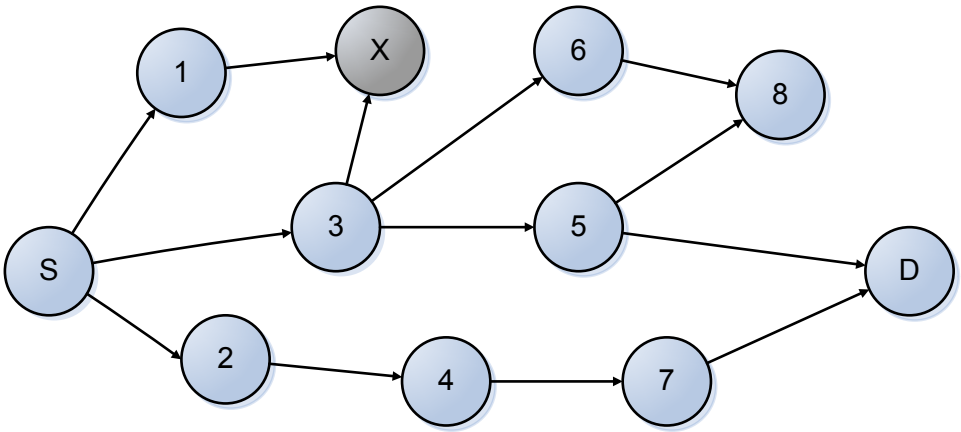
Une variante de cette attaque est "l'attaque de Sybil" où l'attaquant pratique une usurpation d'identités multiples. Il ne se contente pas d'usurper une seule identité mais se fait plutôt passer pour plusieurs nœuds à la fois. Dès lors, l'attaquant a la possibilité d'intercepter l'ensemble du trafic même s'il est reparti sur plusieurs routes.

### *B) Attaque black hole (trou noir)*

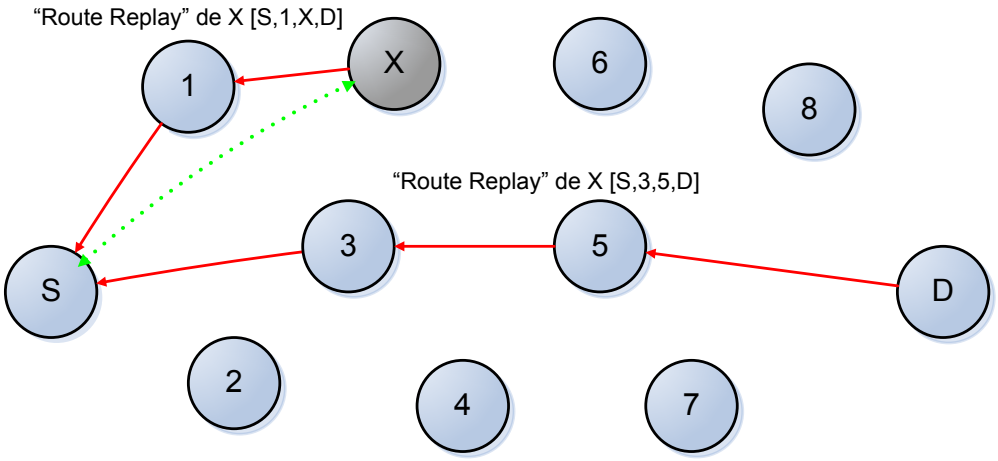
Si un nœud malicieux à la capacité de falsifier les informations telles que vecteurs de distance, liens, routes, etc., il peut lors du mécanisme de découverte de route répondre au nœud initiateur avec un message de type "route replay " en annonçant un chemin, avec un coût minimal, vers le nœud demandé. Le nœud émetteur mettra alors sa table de routage à jour avec cette fausse route. Les paquets de données du nœud émetteur vers le nœud destinataire transiteront par le nœud malicieux qui pourra tout simplement les ignorer. Les paquets sont captés et ignorés par le nœud malicieux. D'où son appellation de "black hole" (trou noir) [BLA]. Ainsi, les connexions sont complètement interrompues, ce qui, en plus de paralyser les communications sur de grands rayons, peut occasionner de véritables partitions du réseau.




a) : Diffusion par le nœud S d'un paquet "route request" pour trouver une route vers le nœud D




b) Chaque nœud intermédiaire rediffuse le paquet en rajoutant son adresse




c) Le nœud malicieux X renvoie un “route replay” à S plus optimisé en nœuds intermédiaires que le “route replay” du nœud D Cette route est choisie par S

**Figure 10. Attaque blackhole**

Dans cette figure, le nœud malicieux X, met dans sa table de route de fausses informations concernant le nœud D (chemin avec nombre de saut plus optimale). Ainsi, lors du mécanisme de recherche de route, il va répondre au nœud S avec un RREP plus optimisé en nombre de sauts que le nœud D. Tous les paquets envoyés par le nœud S vers le nœud D passeront par le nœud X.

Une variante de ce type d’attaque est appelée "grey hole" où seuls certains types de paquets sont ignorés par le nœud malicieux. Par exemple, les paquets de données ne sont pas retransmis alors que les paquets de routage le sont. Un attaquant peut aussi créer des boucles infinies dans le réseau ou imposer aux paquets de faire des détours consommant la ressource radio inutilement. Un nœud malicieux ayant usurpé l'identité d'un nœud valide peut aussi générer des messages d'erreurs de type erreur de route (route error), de manière aléatoire, pour perturber le fonctionnement du mécanisme de maintenance des routes

### *C) Attaque du Wormhole (trou de ver)*

Le terme "Wormhole" [WOR] fait référence aux trous de ver en astronomie, qui sont des raccourcis entre deux points éloignés de l’espace. Le même principe est utilisé ici : l’attaquant utilise un chemin hors du réseau pour acheminer les messages. L’attaque "Wormhole" exige la présence d’au moins deux nœuds malveillants s’entendant dans le réseau. Le trou de ver entraine des problèmes pour les protocoles de routage dont le fonctionnement repose sur la connaissance de la distance entre deux nœuds.

Un nœud malveillant X capture les paquets transmis par un nœud A et les déroutent vers un autre nœud malveillant Y, qui est supposé être localisé à quelque distance. Le deuxième nœud malveillant est alors chargé de retransmettre les paquets détournés. ( voir figure 11)

Il existe deux méthodes pour établir le lien entre les différents nœuds malveillant afin de détourner le trafic.

a) Première méthode

Le nœud malveillant noté X, encapsule les paquets reçus de son voisin A puis les transmet à destination du nœud malveillant Y. Le nœud Y retransmet ces paquets dans son voisinage incluant ainsi le nœud B. Exemple : si le paquet original transmis par le nœud A et détourné par le nœud X était un paquet HELLO, alors le nœud B en recevant ce paquet verra le nœud A comme son voisin, ce qui est faux. Un autre exemple, si le nœud A transmet un paquet **Route Request** (RREQ) pour le nœud B, alors le nœud X peut détourner le trafic vers le nœud Y en l'encapsulant Le résultat est que le RREQ arrivera au nœud de destination B avec un nombre de saut plus petit que les autres. Ce qui entrainera le routage des paquets à travers les nœuds X et Y. notés que les nœuds qui relayent les paquets entre les nœuds X et Y ne peuvent pas les interpréter car ils sont encapsulés donc ils ne pourront pas incrémenter le champ Hop count (nombre de saut).

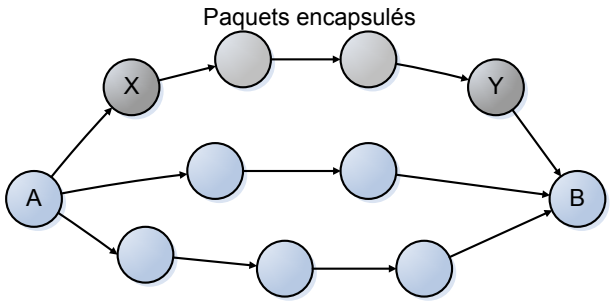



**Figure 11. attaque Wormhole avec paquets encapsulés**

b) Deuxième méthode

Pour établir le lien, les nœuds malveillants sont supposés accéder à une grande bande passante hors canal. Ce lien peut être établi par exemple en utilisant une connexion filaire entre ces deux nœuds ou en ayant un lien sans fil de grande portée utilisant des fréquences différentes. Ainsi, cette méthode nécessite des équipements spécifiques et est plus difficile à mettre en œuvre que la méthode précédente.

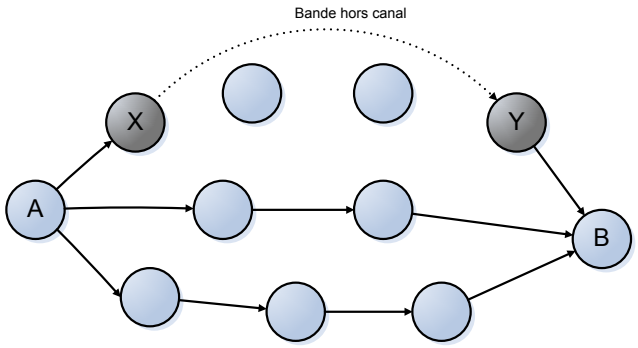



**Figure 12. attaque Wormhole (out-of-band channel)**

### *D) Consommation d'énergie (Sleep deprivation torture)*

Les réseaux ad hoc étant par définition des réseaux dynamiques reposant sur un support sans fil et utilisant des sources d'énergie autonome. Cette source d'énergie a une durée de vie limitée et donc peut être la cible d'une attaque.

Le principe de cette attaque est de forcer le nœud cible à consommer toute son énergie en l'inondant de paquets (déni de service classique). Le nœud doit recevoir ces paquets et les traiter. En maintenant un rythme soutenu, un attaquant peut dés lors épuiser complètement un nœud. D'où l'appellation anglaise qui vient du fait que l'on va constamment demander au nœud de faire quelque chose, l'empêchant ainsi de pouvoir se reposer [STA].

L'attaque pourra être rendue plus efficace en exploitant le protocole de routage. Par exemple, si deux nœuds malveillants utilisent un nœud cible comme relai et échangent des informations de manière soutenue, le nœud relai va recevoir des paquets des deux cotés et va les renvoyer aux destinataires. Si l'on compare la consommation de chaque nœud, il en ressort que le nœud cible consommera autant d'énergie que les deux nœuds malveillants réunis, ce qui l'amènera plus tôt à manquer d'énergie.

Une variante de cette attaque est le comportement égoïste d'un nœud. Le principe de cette attaque est qu'un nœud peut refuser de relais multipoint ou de nœuds intermédiaires, simplement pour économiser son énergie et donc pouvoir profiter plus longtemps du réseau.

### *E) Attaque de Rushing*

Cette attaque affecte surtout les protocoles réactifs. Le but est de faire passer le maximum de messages par l'attaquant. Pour cela, l'attaquant exploite une des propriétés du protocole réactif, c'est-à-dire qu'un nœud ne transmet que le premier message de type **Route Request** reçu et ignore tous les autres (pour une même demande bien sure). En effet s'il reçoit deux même **Route Request**, la deuxième est forcement la moins bonne car plus longue (sinon il l'aurait reçue en premier) [RUS].

L'attaquant va donc retransmettre très rapidement tous les messages **Route Request** qu'il reçoit pour avoir le plus de chance d'être le premier donc plus de chance que la route passe par lui. Pour cela il élimine les temporisations avant l'envoi des messages que vont supporter ces voisins (et concurrents). Ces temporisations ont deux origines:

- Le MAC (Medium Access Control) impose souvent un délai entre le moment où le paquet est délivré à l'interface pour l'envoi et le moment où il est effectivement envoyé.
- Si ce n'est pas le MAC, la majorité des protocoles réactifs le font pour éviter des collisions entre les paquets "Route Request". En effet, puisque les paquets RREQ sont transmis en broadcast, la détection de collision est difficile à gérer. Les protocoles de routage imposent un délai aléatoire avant de retransmettre les paquets

Il est évident que le détournement de la majeure partie du trafic présente un avantage non négligeable pour réaliser ensuite les attaques citées précédemment.

D'une manière générale, la plupart des protocoles routages dans les réseaux Ad hoc sont vulnérables à ces attaques brutales de types déni de service. L'attaque précédente comme on l'a vu, utilise le mécanisme de découverte de routes des protocoles réactifs. Il est aussi possible d'exploiter le mécanisme de maintenance des routes en diffusant à travers le réseau des messages de type "RRER" pour annoncer des ruptures de routes. Les nœuds recevant ce message mettent à jour leurs tables de routage et suppriment des routes existantes, occasionnant éventuellement des partitions du réseau. Les protocoles proactifs ne sont pas plus à l'abri. En effet, un nœud peut très bien diffuser régulièrement des messages "HELLO" dans lesquels, il revendiquera tantôt des liens bidirectionnels, tantôt des ruptures de liens. En réalité, on peut considérer que toutes les attaques décrites précédemment constituent des attaques de type déni de service, puisqu'elles privent les nœuds de ressources, que ce soient la bande passante, l'accès au réseau, l'espace d'adressage, ou leur énergie. Chaque mécanisme d'un protocole peut constituer une faille potentielle et les mécanismes de sécurité doivent non seulement empêcher les attaques existantes mais également veiller à ne pas offrir de vulnérabilités supplémentaires.

## 3.2 Les mécanismes de sécurité dans les réseaux ad hoc

Les mécanismes de sécurité permettent d'offrir un niveau de sécurité supérieur aux protocoles classiques en permettant de renforcer le processus d'acheminement des paquets. Cependant, afin de satisfaire les exigences de sécurités de bases que sont la confidentialité et l'authentification, les protocoles les plus efficaces utilisent des mécanismes plus conventionnels hérités des réseaux filaires tels que le contrôle d'accès, la cryptographie à clés publiques, les fonctions de hachage, les signatures numériques. Ces mécanismes sont dits préventifs car ils visent à empêcher à l'avance, les attaques de nœuds compromis sur le réseau. Parallèlement, certaines approches visent à détecter en temps réel les attaques au sein du réseau, à favoriser la coopération entre les nœuds afin de restreindre l'impact des nœuds malicieux. Ces approches sont dites réactives et peuvent être utilisées en supplément des approches préventives [CHAO].

Dans la suite, nous allons présenter les différentes solutions de sécurité.

### 3.2.1 Solutions pour l'authentification

L'absence d'infrastructure centralisée dans les réseaux ad hoc compromet l'utilisation directe des systèmes d'authentification basés sur la cryptographie à clé publique. En effet, ces systèmes d'authentification supposent l'utilisation de certificats établis par une autorité centrale. Le certificat, signé par l'autorité centrale, permet de garantir qu'une clé publique appartient bien à son propriétaire et non à un usurpateur. L'opération de vérification de certificat ne se limite pas à contrôler la signature de l'autorité centrale. Il est aussi nécessaire de

s'assurer que le certificat est toujours en cours de validité et qu'il n'a pas été révoqué. Une révocation de certificat est indispensable si la clé privée du propriétaire a été volée ou divulguée. Il existe trois grands courants dans le domaine de l'authentification pour les réseaux sans fil ad hoc . Deux de ces orientations se basent sur l'établissement d'une clé secrète permettant par la suite l'authentification des participants. Toute la complexité réside en la manière d'établir et de gérer cette clé. Les deux modèles basés sur une clé secrète sont:

- La clé secrète commune: les participants s'entendent sur une clé secrète [ANJ].
- The Duckling Security Policy Model: le modèle d'authentification présenté dans [STA]

Le troisième axe de recherche pour l'authentification au sein de réseaux ad hoc, se base sur la cryptographie à clé publique et cherche à s'affranchir du besoin d'une entité centrale de certification. Un exemple est l'architecture de certification distribuée présenté dans [ZHO].

### *A) Clé secrète commune ("Key Agreement")*

La clé secrète commune dans les réseaux ad hoc est un mécanisme permettant de partager une clé commune entre plusieurs participants qui ne se connaissent pas à priori [ANJ]. Les participants établissent entre eux une clé secrète leur permettant de s 'authentifier afin de communiquer de manière sécurisée. La mise en place de cette clé peut se faire de manière ***distribuée***. Dans ce cas, la clé secrète est fournie aux participants du réseau ad hoc via un canal supposé sûr. C'est le cas lorsque des collègues, souhaitant établir une communication sûre entre eux à l'occasion d'une réunion dans une salle de conférence close, distribuent un mot de passe inscrit sur un morceau de papier. Une autre manière d'établir une clé secrète commune est de faire en sorte que chaque participant apporte sa ***contribution*** à la clé finale. Lorsqu'il n'y a que deux nœuds, le protocole de Diffie-Hellman [HEL] peut être utilisé.

Il faut noter que le protocole de Diffie-Hellman ne résiste pas aux attaques du type "man in the middle". Il faut donc rajouter des éléments dans les échanges permettant une authentification mutuelle des participants.
Lorsqu'il y'a plus de deux nœuds, le protocole de **Diffie-Hellman doit être généralisé à de multiples participants**. Chaque nœud i possède son exposant secret ei, la clé résultante sera $\alpha^{e1.e2....en}$ pour n participants. La mise en pratique du protocole de Diffie-Hellman à de multiples participants n'est pas chose simple et fait l'objet de nombreuses recherches. En effet, il ne suffit pas que chacun envoie la valeur $\alpha^{ei}$ aux autres pour que tous les nœuds aient la possibilité de déterminer la clé. Le protocole de Diffi-Hellman s'appuie sur le fait que les participants calculent la clé à l'aide de la valeur reçue des autres participants et leur exposant secret. Il faudrait donc que le nœud i reçoive la valeur $\alpha^{e1.e2..ei-1.ei+1....en}$ pour être capable de calculer la clé. La mise en pratique peut se faire comme suit.

1 le premier nœud envoie $\alpha^{e1}$ au second nœud

2 le deuxième nœud envoie $\alpha^{e1.e2}$ au troisième nœud

3 le (n-2) ième nœud envoie $\alpha^{e1.e2...en-2}$ au (n-1) ième

4 le (n-1) ième nœud envoie la valeur $\eta = \alpha^{e1.e2...en-2}$ à tous les autres nœuds

5 le dernier nœud n peut alors déterminer la clé secrète partagée $k = \alpha^{e1.e2...en-2.en} = \eta^{en}$

6 chaque nœud i envoie la valeur $\eta^{Ei/ei}$ au nœud n (Ei est un facteur de brouillage choisi par i)

7 le nœud n renvoie à chacun des nœuds la valeur $(\eta^{Ei/ei})^{en} = \eta^{en.Ei/ei}$

8 les nœuds i enlèvent le facteur de brouillage Ei/ei pour retrouver la clé $k = \eta^{en=>} (\eta^{en.Ei/ei})^{ei/Ei} = \eta^{en}$

Cette application de Diffie-Hellman, à multiples participants a pour inconvénient la nécessité d'établir au préalable une relation d'ordre entre les nœuds du réseau. De plus, il faut que la pénultième et l'antépénultième éléments aient la possibilité de communiquer avec tous les autres nœuds, ce qui est une contrainte beaucoup trop forte pour les réseaux ad hoc. D'autres possibilités ont été étudiées [CHAO] [ANJ]. Elles se basent sur le principe d'établir deux clés secrètes indépendamment entre deux groupes parallèles de deux entités à l'aide du protocole de Diffie-Hellman. Les deux clés secrètes sont ensuite utilisées comme exposant secret pour établir une troisième clé secrète entre les deux groupes de deux entités.

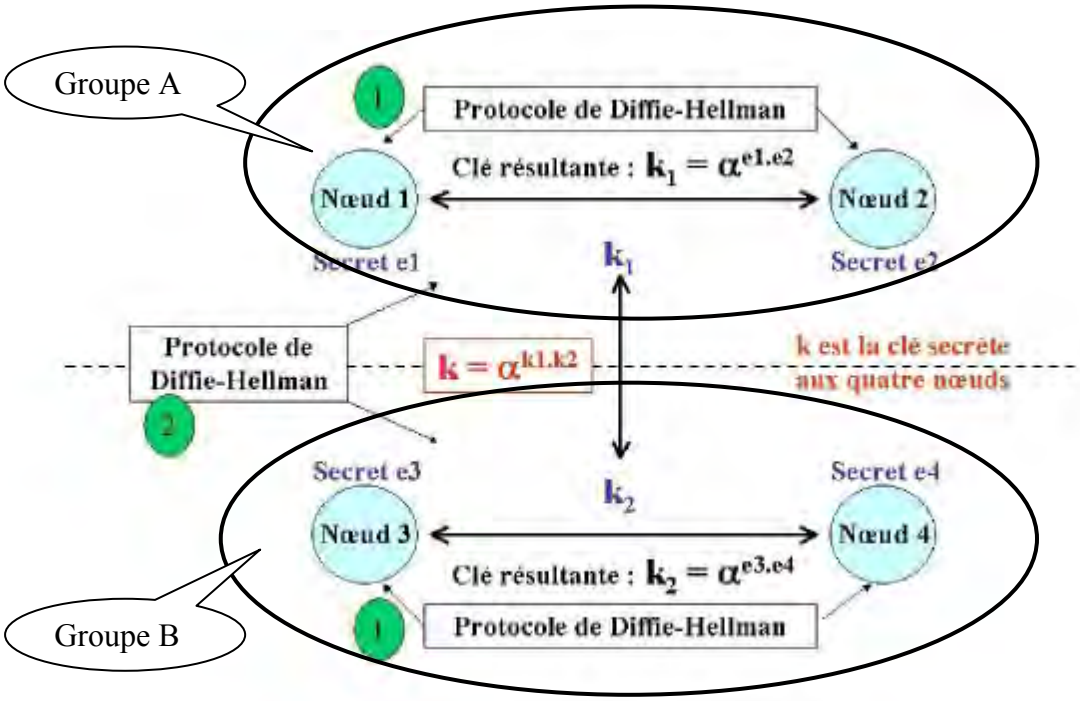



**Figure 13. Diffie-Hellman à quatre participants**

### *B) Le "resurrecting duckling"*

Dans une volonté de permettre une distribution facilitée des clés dans un réseau ad hoc, Frank Stajano et Ross Anderson ont proposé un mécanisme pour échanger une clé secrète entre deux nœuds appelé le modèle ***Resurrecting Duckling***. Ce modèle repose sur le principe d'une relation Mère-fils (Maitre-esclave) entre deux nœuds du réseau qui se feront dés lors mutuellement confiance. On dit que le maitre appose son empreinte à l'esclave. Ainsi, au première démarrage ("naissance"), le nœud considère le premier équipement qu'il voit comme sa "mère'. Il lui obéira jusqu'à la mort. Lors de cette opération une clé secrète est échangée entre les deux entités via un canal supposé sûr. Il peut s'agir d'un contact physique entre le maitre et l'esclave. Lorsque le nœud quitte le réseau, il reviendra à l'état "d'œuf" et peut renaitre. Par la suite, cette clé peut être utilisée pour chiffrer et authentifier des informations, comme une liste d'autres clés partagées par exemple.

Cette approche apporte énormément en termes de sécurité même si sa mise en place n'est pas aisée. Il reste pourtant un problème: le cas des nœuds corrompus. En effet, un nœud correctement authentifié qui décide d'agir de manière délictueuse peut le faire en toute impunité [CHAO].

### *C) L'architecture de certification distribuée*

Afin de remédier aux contraintes induites par l'absence d'infrastructure centralisée, il a été proposé une nouvelle approche de gestion des certificats profitant des caractéristiques intrinsèques des réseaux ad hoc [ZHO]. Dans ce système de certification de clés, l'autorité est non plus confiée à une seule entité fixe mais il est au contraire distribué entre plusieurs nœuds du réseau. De ce fait, le service de certification obtenu revient à définir une autorité de certification distribuée disposant d'un paire de clés publique/privée. La clé publique **K** est connue de chaque nœud du réseau, ce qui leur permet de vérifier en confiance tout certificat signé avec cette clé privée. La clé privée **k** n'est connue d'aucun nœud particulier, mais se trouve en fait partiellement distribuée sur des nœuds serveurs appelés ***contributeurs***.

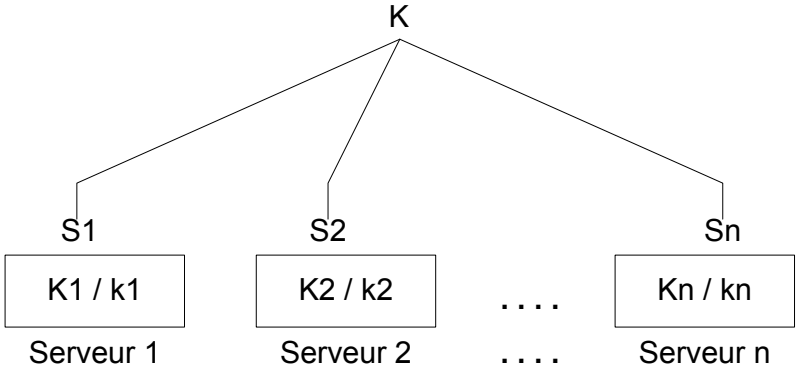



**Figure 14. Configuration du service de gestion des clés**

Ainsi, un nœud client qui souhaite obtenir les clés publiques des autres clients ou lancer des mises à jour pour changer sa propre clé publique, émet une

requête vers le service de certification. Pour garantir un niveau suffisant de sécurité même dans un contexte distribué, le service de certification repose sur la ***cryptographie à seuil***. Un schéma de cryptographie à seuil ***(n, t+1)*** est conçu de telle sorte que parmi les ***n*** nœuds qui partagent la gestion des clés, ***t+1*** auront la possibilité de procéder aux opérations de chiffrement, tandis que ***t*** nœuds seulement en seront incapables, même en coalition. Ainsi, lorsque le service doit signer un certificat, chaque nœud serveur génère une signature partielle en utilisant sa clé privée, et transmet le résultat à un autre serveur appelé ***assembleur*** qui sera chargé d'assembler les portions de signature des **t** nœuds. Lorsque le nœud serveur a reçu ***t+1*** signatures partielles correctes, il est capable de calculer la signature finale du certificat.

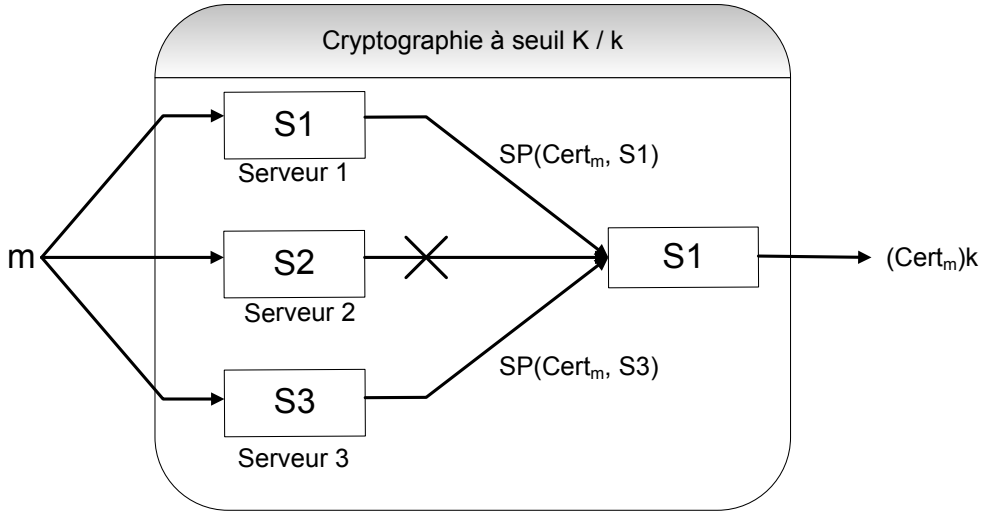



**Figure 15. la technique de cryptographie à seuil avec le paramétrage (3,2)**

On notera que ce procède d'assembleur peut être rempli par n'importe lequel des ***n*** nœuds contributeurs. Pour renforcer la robustesse du dispositif et déjouer la compromission éventuelle de ce serveur, les autres préconisent d'affecter éventuellement ce rôle à ***t+1*** nœuds simultanément. Ainsi la phase de vérification des signatures en est considérablement alourdie. L'avantage de ce modèle réside dans le fait que t nœuds malveillants complices ne peuvent créer de certificat valide puisque t+1 signature partielles valides sont nécessaires. Cependant, nous ne sommes pas à l'abri d'un attaquant qui génère systématiquement de fausses signatures, en vue de conduire à la création d'un certificat invalide d'une signature en utilisant la clé publique du service. Dans le cas où la vérification échoue, l'assembleur se doit de désigner un autre ensemble de t+1 signatures partielles. Cette procédure continue jusqu'à ce qu'il parvienne à générer une signature correcte.

### 3.2.2 Le protocole TESLA

Le protocole TESLA (Time Effcient Stream Loss-tolerant Authentication) a été proposé dans [PERR]. C'est un mécanisme de sécurité qui permet d'assurer l'authentification de la source des messages avec un Code d'Authentification de Message (MAC) dépendant d'une clé secrète qui n'est divulguée par l'émetteur du message qu'après un délai d'attente **ζ**. La valeur **ζ** est calculée de manière à ce qu'on soit sur que le destinataire a reçu le message avant la divulgation de la

clé, cette condition garantie l'intégrité du message. Le temps **ζ** ne doit pas être trop important pour limiter les latences dans le réseau, en effet un destinataire doit attendre la divulgation de la clé secrète avant de pouvoir effectivement traiter un message. La figure ci-dessous décrit le fonctionnement de TESLA.

Le mécanisme commence avec l'initialisation des clés MAC. Pour cela, l'émetteur génère une série de clés K1, K2 …, Kt à l'aide d'une fonction de hachage à sens unique. Il détermine ainsi une premier clé de manière aléatoire et calcule les clés suivantes en appliquant successivement la fonction de hachage : Ki = h (Ki+1). En suite, il génère les clés MAC à l'aide d'une autre fonction de hachage à sens unique : K' = h (Ki+1). L'utilisation de deux fonctions de hachages distinctes est une précaution prise par les auteurs pour renforcer encore d'avantage la sécurité.

Par la suite, l'émetteur associe à chaque paquet un MAC calculé à partir de son contenu et généré grâce à la fonction de hachage à sens unique. Ensuite le temps est divisé en plusieurs intervalles **ζ** de durée fixe. Une fois un paquet envoyé et à l 'expiration d'un délai prédéfini, il peut divulguer la clé correspondante qui servira à authentifier le paquet.

De ce fait, lorsque le récepteur reçoit un paquet comportant un indice d'intervalle **ζ**, il doit estimer l'intervalle dans lequel se trouve l'émetteur à l'aide de son horloge locale en prenant en compte le temps de transmission d'un paquet et une estimation de l'horloge de l 'émetteur. Cette estimation permettra de vérifier que l'émetteur n'a pas encore divulgué la clé Ki. Si cette condition n'est pas respectée, l'intégrité n'est plus formellement garantie et le paquet est rejeté. Dans le cas contraire, le récepteur ne peut pas encore vérifier l'authenticité du message pendant l'intervalle **ζ** sans la clé correspondante Ki. Une fois celle-ci reçu, le récepteur s'assure de sa légitimité en l a hachant successivement plusieurs fois et en comparant l'empreinte obtenue à la valeur d'une clé antérieure.

Un des principaux avantages de TESLA est lié aux propriétés des chaines de hachage. A partir d'une chaine révélée, on peut calculer toutes les clés précédentes, de sorte que même si plusieurs paquets d'un même intervalle sont perdus, un nœud peut toujours les vérifier à partir d'une clé obtenue ultérieurement. Ceci caractérise la capacité de tolérance aux pertes de TESLA. On peut citer aussi le caractère unidirectionnel des flux, c'est-à-dire de la source vers une ou plusieurs destinations. C'est cette capacité de passer à l'échelle qui permet à TESLA d'être utilisé dans le cadre des flux ***multicast***.

Cependant, certains aspects de TESLA requièrent une attention toute particulière. Le choix du délai **ζ** est crucial. Il doit être choisi à la fois suffisamment grand pour que le destinataire ait reçu le message avant la clé et en même temps suffisamment court afin d'assurer une bonne réactivité au réseau. A ce sujet, plusieurs conditions doivent être satisfaites pour faire fonctionner ce protocole. Le premier est la capacité du réseau à se synchroniser entre eux, le deuxième est le besoin pour TESLA d'être démarré par un système permettant de mettre en place les clés.

### 3.2.3 Le protocole NLP (neighbour lookup protocol)

Ce mécanisme permet de détecter les nœuds qui ont un comportement suspect et à les exclure du réseau. Même si elle a été élaborée pour un protocole réactif, le principe peut cependant être adapté pour un protocole de routage proactif [LEB], [MAR].

Ce système se décompose en deux parties

- Le ***watchdog*** qui se charge de surveiller le comportement des voisins.
- Le ***pathrater*** qui va prendre en compte le comportement des voisins pour le choix des routes à emprunter.

Les différentes catégories de mauvais comportement d'un nœud sont :

- Nœud surchargé, le nœud voudrait bien se comporter normalement mais n'a pas de ressources pour le CPU, mémoire, bande passante, etc.
- Nœud égoïste, le nœud ne va pas router le trafic qui ne le concerne pas.
- Nœud endommagé, un problème matériel ou logiciel empêche le nœud de faire correctement son travail.
- Nœud malicieux, le nœud va jeter les paquets pour réaliser une attaque de type déni de service par exemple.

Chaque *watchog* écoute le trafic qui transite et relève les incohérences dans celui-ci. Chaque incohérence constatée va augmenter le score de "mauvais comportement " du nœud considère. Si ce score dépasse un certain seuil, le *pathrater* considère le nœud comme pas sur et va donc utiliser un chemin qui l'évite.

### 3.2.4 Mécanismes basés sur la réputation

Les mécanismes détaillés précédemment se révèlent efficaces pour assurer les fonctionnalités de sécurités classiques que sont la confidentialité, l'intégrité et surtout l'authentification. Ils permettent ainsi d'empêcher de nombreuses attaques qui perturbent considérablement le processus de routage. En revanche, ils ne se révèlent pas du tout adapter pour résoudre le problème de non-participation des nœuds. En effet les mécanismes cryptographiques, aussi efficaces soient-ils ne permettent pas d'assurer qu'un nœud participe au processus de routage en relayant tous les paquets. Or, dans le contexte des réseaux ad hoc, c'est là une fonctionnalité primordiale dans la mesure où ce type de réseaux est basé sur la coopération entre les nœuds. C'est pourquoi en sus des mécanismes de sécurité, certains protocoles visent plus spécifiquement l'incitation à la coopération [CHAO].

### *A) Mécanismes de micro paiement*

Le concept consiste à monnayer les services auxquels les nœuds souhaitent accéder en échange de crédits virtuels. Pour obtenir ces crédits, chaque nœud doit fournir des services aux autres nœuds. Les crédits sont ultérieurement dépensés pour pouvoir acheter des services. Si un nœud n'a plus assez de crédits pour acheter le moindre service, cela signifie alors qu'il n'a pas suffisamment participé au bon fonctionnement du processus de routage [CHAO].

Le protocole NUGLETS [BUT], s'inscrit dans cette optique. Son objectif est à la fois d'inciter les nœuds à participer et de limiter les inondations du réseau, dès lors qu'elles deviennent payantes. Deux modèles sont spécifies par le protocole. Dans le premier, un nœud désirant envoyer un paquet doit au préalable y incorporer suffisamment de crédit. Par la suite, chaque nœud intermédiaire sur la route prélève une quantité de crédits. Si le nombre de crédits est insuffisant, le paquet est rejeté. L'intérêt de cette approche est qu'elle limite les attaques de types déni de service dans la mesure où aucun nœud ne peut se permettre de financer une inondation. En revanche, elle implique que chaque nœud connaisse par avance le nombre de nœuds sur la route. Dans le second modèle, le routage fait l'objet de transaction puisque ce sont ici les nœuds destinataires qui doivent payer pour recevoir pour recevoir les paquets qui leur sont destinés. En effet, chaque nœud achète les paquets reçus de son voisin amont et le destinataire d'un paquet l'achète donc au dernier nœud intermédiaire. Cette approche présente un inconvénient encore plus conséquent que la précédente puisqu'elle ne permet pas d'empêcher un attaquant d'inonder le réseau. Au contraire, un nœud peut être tente de relayer beaucoup de paquets vers de nombreux nœuds afin de maximiser ses profits lors des transactions.

D'une manière générale, ces types de protocoles collent pas suffisamment au modèle ad hoc. Tout d'abord, ils ne prennent pas assez en compte la mobilité des nœuds. En effet, si un nœud intermédiaire quitte la route, le paquet est perdu ainsi que l'investissement en termes de crédits. Enfin, cette approche pose de gros problèmes concernant le fonctionnement même du protocole de routage. Ainsi dans le cas d'un protocole réactif, les nœuds peuvent être tentés de ne pas envoyer de messages RRER lors de la détection de la rupture d'un lien puisqu'ils auraient alors à payer cela. Dans le cas d'un protocole proactif, cela concernerait les messages de contrôle qui deviendraient alors trop couteux.

### *B) Mécanismes basés sur la confiance*

Ces mécanismes ont pour but de fournir des classements des nœuds afin de différencier les "bons" nœuds, qui ont une bonne réputation car ils coopèrent régulièrement, des "mauvais" qui adoptent un comportement égoïste.

### a) Le protocole CONFIDANT (Cooperation Of Nodes – Fairness In Dynamic Ad hoc Network)

Le protocole COFIDANT [BUC] utilise une infrastructure à clés publiques auto-organisée inspirée du protocole PGP (Prety Goog Privacy). L'objectif de CONFIDANT est de traiter à la fois les nœuds malicieux et égoïstes à travers la supervision et l'analyse de deux processus du routage à savoir le transfert de données et la découverte de voisins. CONFIDANT se compose de quatre éléments complémentaires : le moniteur, le moniteur de confiance, le système de réputation et le mécanisme de gestion des chemins. Le rôle du moniteur consiste à s'assurer que les voisins du nœud auquel il est rattaché relaient correctement le paquet. Lorsque le moniteur détecte une anomalie ou une incohérence, il avertit le système de réputation, qui de son coté maintient à jour des listes de notes pour chaque nœud observé. Les listes peuvent être éventuellement échangées entre les nœuds. Ainsi, si une liste est reçue d'un nœud de grande confiance, le récepteur peut directement enregistrer les informations à l'intérieur de sa propre liste. Dans le cas contraire, si la liste est envoyée par un nœud suspect, le récepteur peut l'ignorer totalement ou bien encore l'accepter tout en lui donnant moins d'importance qu'une liste reçue d'un nœud sûr. Finalement le mécanisme de gestion de chemins détermine les routes les plus sûres à partir des listes de nœuds exclus et des nœuds de confiance. En outre, il peut décider de refuser de relayer les requêtes en provenance de nœuds mal notés.

Concernant la gestion de la confiance, l'approche s'inspire de celle utilisée dans PGP. Ainsi, les nœuds disposent de quatre niveaux de confiance : ami, marginal, inconnu ou ennemi. Chaque nœud enregistre ses amis dans une liste dédiée. Par la suite si un nœud A parvient à détecter un comportement malicieux de la part d'un nœud B, le nœud A va avertir tous les amis contenus dans la liste à l'aide d'un message d'alarme signée.

Cependant une des principales motivations qui peuvent poussé un nœud à ne pas participer au routage est l'économie d'énergie entrainant ce dernier à adopter un comportement égoïste. Pour remédier à ce problème, le protocole CORE a été proposé par Michardi et Molva.

### b) Le protocole CORE (a Collaborative reputation mechanism to enforce node cooperation in mobile ad hoc network)

Ce protocole CORE [MIC] permet non pas d'exclure les nœuds mais au contraire de les obliger à participer en rejetant leurs paquets jusqu'à ce qu'ils coopèrent au processus de routage. CORE se base sur les hypothèses suivantes :

- Les identités des nœuds sont uniques et non modifiables
- Un mécanisme de routage adapté est à même de sécuriser la phase de découverte de route
- Le trafic à l'intérieur du réseau est suffisamment dense

CORE souffre malheureusement de défauts importants. Premièrement, il ne résout pas réellement le problème de non-participation. Certes, les paquets des nœuds égoïstes sont systématiquement rejetés mais en contrepartie de

grandes quantités de données sont perdues, diminuant considérablement le rendement du réseau. Enfin, le protocole repose sur des hypothèses telles que routage sécurisé, adresses uniques et non usurpables... Et ceux-ci constituent un inconvénient commun à tous les protocoles basés sur la réputation. En effet, reposant sur l'information observée sur les nœuds, ils doivent requérir un mécanisme d'authentification forte afin d'affecter une note aux nœuds légitimes. D'autre part, il y'a le problème de "dénonciation calomnieuse" dans laquelle un nœud malicieux génère de fausses alertes pour mettre sur liste noire des nœuds honnêtes. Ce type de mécanismes est d'autant plus vulnérables face à des nœuds complices qui s'attribuent entre eux de bonnes notes et affectent en contrepartie de mauvaises notes aux nœuds honnêtes. Enfin la confiance ne s'établissement que progressivement entre les nœuds, une attaque peut plus facilement être menée au début de la mise en place du réseau.

Afin de rendre plus sécuriser le routage dans les réseaux ad hoc, certains auteurs ont ajouté ces mécanismes de sécurité aux protocoles décrits dans le chapitre 2.

## 3.3 Sécurisation du routage pour réseaux ad hoc

### 3.3.1 Protocoles basés sur la cryptographie symétrique

#### *A) Le protocole SRP (Secure routing Protocol)*

SRP [PAP] est un protocole spécialement adapté aux caractéristiques du protocole DSR. Les auteurs ont conçus SRP avec une extension de l'entête des paquets Route_Request er Route_Reply. SRP utilise des numéros de séquences à l'intérieur des requêtes afin de garantir leur fraicheur. Cependant, ce numéro de séquence ne peut être vérifié qu'au niveau de la destination. Il établit en outre des associations de sécurité, entre les nœuds communiquant uniquement. Cette association est utilisée pour authentifier les paquets Route_Request et Route_Reply par le biais de MAC. Au niveau de la destination, SRP permet de détecter des modifications de paquets de type Route_request tandis qu'au niveau de la source, c'est l'intégrité des Route_Reply qui est analysée.

Le fait que SRP utilise des associations de sécurité entre les entités communiquant, rend ce protocole relativement léger. En revanche, certains de ces défauts sont assez pénalisants. En plus de ne pas sécurisé le mécanisme de maintenance des routes, SRP ne permet de détecter les modifications sur les modifiables des paquets. Ainsi, un nœud peut aisément corrompre voir supprimer le contenu de la liste de nœuds dans le paquet Route_Request. Enfin, l'intégrité des paquets étant vérifié qu'au niveau des nœuds sources et destination, un attaquant peut corrompre des paquets de manière à gaspiller les ressources du réseau avec des rejeu.

### *B) Le protocole SAR (Security-award Ad hoc Routing protocol)*

Le protocole SAR [YI] se base lui sur la cryptographie symétrique. Il a été élaboré pour prévenir des attaques "trou noir" qui consiste à supprimer l'intégralité des paquets au niveau d'un nœud malicieux. SAR est conçu pour sécuriser les protocoles AODV et DSR et utilise la notion de niveau de confiance pour établir la sécurité d'un chemin. Ainsi, lorsqu'un nœud désire établir une route avec un certain niveau de confiance, il génère un nouveau paquet RREQ indiquant le niveau requis et seuls les nœuds satisfaisant le niveau de sécurité requis peuvent rediffuser la requête à ses voisins. Si le paquet atteint la destination, celle-ci génère un paquet RREP avec le même niveau de sécurité. Si aucune route ne garantit le niveau de sécurité requis, celui-ci peut être ajusté par le nœud source.

Cette approche nécessite de lier l'identité d'un nœud à un certain niveau de sécurité. Pour ce faire, une clé secrète est créée pour chaque niveau de sécurité et celle-ci doit être distribuée à tous les nœuds du réseau satisfaisant ce niveau de sécurité. Le contenu ainsi que l'entête des paquets sont ensuite chiffrés par la clé de sorte que les nœuds de niveau inferieur ne peuvent pas les lire.

Le fait de partitionner le réseau en différents niveaux de sécurité fait de ce protocole toute son originalité. Cependant quelques faiblesses sont à noter. Le principal défaut réside sur la distribution des clés. Celle-ci doit être effectuée à la mise en place du réseau, par le biais d'un canal sécurisé. D'autre part, le fait de chiffrer et déchiffrer les paquets (y compris les entêtes) risque d'avoir un impact important sur les ressources du réseau et ouvre la porte aux attaques de types déni de service.

### *C) Le protocole ARIADNE*

Ce protocole s'inspire de DSR et s'appuie sur des mécanismes de chiffrement symétrique [HU]. Le but était de proposer un protocole adapté aussi bien sur les portables puissants que sur les assistants personnels, c'est pourquoi trois méthodes d'authentification sont utilisées afin de s'adapter aux capacités de calculs des nœuds:

- une clé partagée pour chaque paire de nœuds ;
- une clé partagée entre chaque paire de nœuds communicants combinée à une authentification par diffusion ;
- les signatures numériques.

ARIADNE peut se diviser en deux parties. La première permet à un nœud destinataire de vérifier l'authenticité de l'émetteur d'une requête de route (RREQ). La seconde consiste à recourir à des techniques de hachage afin de s'assurer de l'intégrité de la liste des nœuds embarquée dans la requête.

Pour la phase de découverte de routes, supposons qu'un nœud S veut établir la route vers la destination D avec lequel il partage la clé secrète K. pour prouver à la destination que les champs du paquet RREQ sont corrects, le nœud S

ajoute un MAC calculé à partir de la clé K, ainsi qu'une estampille. Par la suite, D peut facilement vérifier l'authenticité et la fraicheur du message en utilisant la clé secrète. Lors de cette phase, le nœud de destination authentifie les nœuds intermédiaires traversés grâce au protocole TESLA.

Aussi, ARIADNE utilise également les fonctions de hachage à sens unique afin de garantir la sécurité globale du routage (exemple : la suppression d'adresse dans la liste des nœuds d'une requête). Ainsi si un nœud malicieux désire ajouter ou supprimer une adresse, il doit soit capturer un paquet sans l'adresse de ce nœud, soit être capable d'inverser la fonction de hachage (ce qui est supposé infaisable).

Il est à noter que l'utilisation de TESLA permet de s'affranchir de la couteuse et délicate distribution de clés privées. En revanche, ce gain est effectué au détriment de la réactivité du protocole puisque l'utilisation de TESLA occasionne une sensible augmentation du délai d'authentification.

Malgré ces protections, ARIADNE est vulnérable à une attaque de type tunnel (wormhole). Ainsi les auteurs proposent de choisir les routes suivant leur performance en termes d'acheminement de paquets. Aussi ce protocole ne permet pas de contrer les attaques par non-participation. Par exemple, il ne permet pas de gérer les nœuds malicieux qui refusent de transmettre les messages d'erreur de route (RERR). Pour parer ces attaques aussi complexes que la non-participation, des mécanismes comme les protocoles CORE ou CONFIDANT peuvent être utilisé en complément.

### 3.3.2 Protocoles basés sur la cryptographie asymétrique

#### *A) Le protocol ARAN (A secure Routing protocol for Ad hoc Network)*

ARAN [SAN] est un protocole à la demande qui fournit un service d'authentification de saut en saut par le biais d'une infrastructure à clés publiques. Il suppose donc l'existence d'un serveur d'authentification T, dont le rôle est de gérer les certificats et dont la clé publique est connue de tous les participants. Il utilise la cryptographie à clés publiques pour sécuriser les routes. Ainsi, avant d'entrer dans le réseau, chaque nœud doit s'authentifier auprès du serveur et solliciter un certificat qui lui servira à signer les messages qu'il transmettra. Ce certificat contient l'adresse IP du nœud, sa clé publique, une première estampille qui rend compte de la date de création du certificat, et une seconde qui indique sa date d'expiration. Le principe de ARAN est de sécuriser le mécanisme de découverte de route de nœud en nœud. Lorsque qu'un nœud désire émettre un message, il génère, signe puis diffuse un paquet de type RDP (Route Discover Packt). Par la suite, chaque nœud intermédiaire recevant ce paquet vérifie le certificat du noud précédent, appose son propre certificat et rediffuse ce paquet. Arrivée au nœud de destination, celui-ci vérifie à son tour le certificat et répond en unicast, par un message de type REP (Reply Packet) qui en son tour vérifier de nœud en nœud.

### *B) SAODV (Secure Ad hoc On-demand Distance Vector)*

Il a été proposé dans [ZAP]. C'est un protocole dédié à la sécurisation de AODV. L'idée principale de ce protocole consiste à utiliser des signatures numériques afin d'authentifier la plupart des champs non modifiables des paquets Route_Request et Route_Replay et d'utiliser des chaînes de hachage pour protéger l'intégrité du compteur de sauts.

Ainsi, SAODV arrive à contrer les attaques "Usurpation d'identité". SAODV nécessite la présence d'une autorité de certification afin de vérifier les paquets signés assurant ainsi leur authenticité. Dans SAODV, chaque paquet RREQ inclut une extension de signature simple. La source choisit un nombre de saut maximal et il génère ensuite une chaîne de hachage à sens unique d'une longueur égale au nombre de sauts plus 1. Avant de relayer une requête, un nœud commence par vérifier l'authenticité du message afin de s'assurer que chaque champ est valide. Il incrément le nombre de saut, le hache, ajoute l'empreinte et rediffuse le tout. Arrivée à la destination, celle-ci vérifie son authenticité. Si la requête est invalide, elle est simplement supprimée. Autrement, le processus est similaire à AODV.

Ce protocole assure une bonne authentification des messages ainsi qu'une bonne intégrité. Cependant, l'utilisation de chaîne de hachage ne permet pas d'empêcher à cent pour cent les attaques sur le nombre de sauts. Ainsi, bien que le hachage du nombre de sauts empêche un nœud malicieux d'annoncer des routes plus courtes, rien ne l'empêche par contre d'augmenter volontairement la longueur d'une route. En effet, un tel nœud peut appliquer plusieurs fois la fonction de hachage plusieurs fois avant de relayer le nœud. La route apparaît ensuite plus longue qu'elle n'est en réalité.

Nous terminons ce chapitre par ce tableau ci-dessous résumant les protocoles sécurisés, les attaques résolues ainsi que la sécurité apportée aux protocoles de routage réactifs.

| **Performance des Types paramètres** | **ARAN** | **ARIADNE** | **SAODV** | **SRP** |
|---|---|---|---|---|
| **Algorithme de cryptage** | Asymétrique | Symétrique | Asymétrique | Symétrique |
| **Protocole MANET** | AODV/DSR | DSR | AODV | DSR/ZRP |
| **Autorité centrale de confiance** | CA exigé | KDC exigé | CA exigé | CA exigé |
| **Authentification** | Oui | Oui | Oui | Oui |
| **Confidentialité** | Oui | Non | Non | Non |
| **Intégrité** | Oui | Oui | Oui | Oui |
| **Non répudiation** | Oui | Non | Oui | Non |
| **Antispoofing** | Oui | Oui | Oui | Oui |
| **Attaques DoS** | Non | Oui | Non | Oui |
| **Black hole** | Oui | Oui | Oui | Non |
| **Worm hole** | Non | Oui | Non | Non |

Tableau 1. Tableau récapitulatif des protocoles sécurisés

## Conclusion

Nous avons vu dans ce chapitre que tous les protocoles de routage classiques dans les réseaux Ad hoc (AODV, DSDV, OLSR…) sont particulièrement vulnérables à un grand nombre d'attaques qui peuvent aller de la capture d'informations à la paralysie totale du réseau. Cependant les procédés proposés pour sécuriser ces protocoles diffèrent d'un algorithme à l'autre et les caractéristiques des réseaux Ad hoc (la mobilité, l'absence d'infrastructure, la limitation de ressources) imposent de repenser complètement les dispositifs de protection classiques et obligent les concepteurs a faire un compromis entre la sécurité des protocoles et les contraintes de performances.

# PARTIE II : PROBLEMATIQUE DE RECHERCHE

# *Chapitre 4: Problématique de recherche*

L'apparition de nouvelles applications multimédias, telles que la vidéo, l'audio etc., a généré de nouveaux types de trafics dans les réseaux. Ces nouveaux trafics nécessitent des traitements spécifiques pour être en mesure de bien gérer leurs applications surtout dans les réseaux ad hoc mobiles. C'est cette différentiation de traitement entre ces nouveaux flux multimédias et les flux traditionnels (plus élastiques) que nous proposons d'apporter au routage ad hoc grâce aux Systèmes Multi Agents (SMA). Notre apport de routage avec QoS, grâce aux SMA, va se baser sur le protocole de routage Multi agent ARA (ant-colony based Routing algorithm for manet) que nous avons détaillé dans le chapitre 1.

## 4.1 Constat

Pour certaines applications qui ont vu le jour dans les réseaux ad hoc, comme par exemple la diffusion de vidéo ou la téléphonie sur IP, les protocoles de routage proposés par le groupe MANET (Mobile Ad hoc NETworks) ne sont pas toujours en mesure d'assurer les contraintes demandées par ces applications. La croissance exponentielle du trafic au niveau des réseaux ad hoc provoque un déséquilibre au niveau du routage. Ce déséquilibre est caractérisé par des contraintes en termes de délai, débit, gigue ou variation de délai, pertes de paquets etc., mais aussi un manque de collaboration au processus de routage.

En effet à cause des propriétés propres aux réseaux ad hoc, les protocoles de routage proposés par le groupe Manet deviennent inadaptés pour prendre en charge ces types de trafic. Ces protocoles ont des limites surtout dans le cadre du routage où la recherche de routes provoque des temps de latence importants et une forte consommation en bande pas sante. En effet, ces protocoles sont sans qualité de service, c'est-à-dire que leur objectif est de trouver une route vers un nœud (mobile) destinataire sans tenir compte de l'état des ressources à travers le réseau, il semble donc nécessaire de s'orienter vers d'autres approches, comme par exemple la mise en place de la qualité de service dans le processus de routage.

Parmi les propriétés propres aux réseaux ad hoc que sont :

- le médium ou canal radio, partagé par tous les nœuds, qui entraîne une faible ressource en bande passante.
- La mobilité des nœuds inhérente à ces réseaux qui se révèle un facteur contraignant car gérant des modifications de topologies.
- La versatilité du médium qui change rapidement entraîne une instabilité des transmissions radio.
- Autonomie en énergie limitée : les applications relatives aux réseaux sans fil tirent leur autonomie des batteries et les opérations (émettre, recevoir des données, écouter le support radio) consomment de l'énergie non négligeable.
- Une faible sécurité : le canal radio n'étant pas isolé, il est très facile à l'aide d'une antenne espionne d'écouter les informations

qui circule sur le canal d'autant plus qu'il est partagé. Les solutions se trouvent donc au niveau d'un cryptage par l'émetteur.

- Le manque d'administration centralisé,

A la suite de l'étude faite sur la sécurité des réseaux Ad hoc, on remarque que, d'une part l'authentification et l'intégrité des nœuds permettent de se protéger contre les attaques de type usurpation d'identité, déni de service et autres que peuvent lancer des attaquants (exemple envoi de faux paquets RREQ), et d'autre part une bonne coopération des nœuds est nécessaire voir même obligatoire pour la bonne marche du processus de routage dans ces réseaux. Chaque nœud du réseau joue un rôle important dans l'acheminement des paquets de la source vers la destination, d'où la nécessité d'assurer une totale collaboration de ces derniers afin d'assurer les services nécessaires à la bonne marche du réseau. Donc, les nœuds ne doivent pas être des nœuds égoïstes.

En effet, dans le routage avec QoS, la non coopération des nœuds égoïstes pour le routage des paquets peut causer un disfonctionnement du réseau, entraînant une inefficacité du protocole de routage. En plus, les attaques de types déni de service vu dans le chapitre 3 peuvent aussi, par exemple, paralyser le fonctionnement de ces protocoles. Un nœud malveillant peut inonder le réseau de faux paquets RREQ de réservation de bande passante, causant à la longue la non disponibilité des ressources au sein des nœuds pour les paquets valides. Pour un protocole avec qualité de service, chaque nœud intermédiaire disposant des ressources (bande passante, énergie,…) doit le mettre à la disposition des autres nœuds du réseau le sollicitant.

Une autre problématique est que les paquets des protocoles de routage, à savoir RREQ, RREP sont transmis en clair dans le réseau. Si aucun control n'est fait sur la provenance des messages de routage, un nœud pourra facilement causer des perturbations au fonctionnement du protocole. Par exemple, un nœud malveillant peut, en faisant du " *promiscuous listening* ", capturer et modifier les contenus des paquets. Cela est d'autant plus facile que les réseaux ad hoc n'ont pas de barrière physique pour se protéger et que tous nœuds à portée peuvent potentiellement participer au mécanisme de routage. À partir du paquet RREQ capturé, il peut connaître l'adresse de la source et de la destination et ainsi forger un faux paquet RREQ en se présentant comme la Source ("Usurpation d'identité"). Ainsi, il peut mettre œuvre d'autres attaques comme le "Black hole", le "Rushing attack". Des nœuds malicieux peuvent être amenés à envoyer de faux paquets RREQ de réservations de ressources, les nœuds intermédiaires qui reçoit ces paquets et qui satisfont à la demande réservent leur bande passante. Cette attaque aura comme conséquence de saturer le réseau car aucune bande passante ne sera disponible pour les vrais paquets RREQ QoS.

Même si les réseaux ad hoc constituent une solution tout à fait prometteuse aux problèmes actuels lies à la mobilité des utilisateurs et des réseaux eux-mêmes, leur développement est freiné aujourd'hui par l'absence de mécanismes de sécurité efficaces pour subvenir aux besoins actuels en protection des données et de ces ressources requises tels que les applications

multimédias. En effet les solutions décrites dans 3.4 permettent d'améliorer sensiblement la sécurité du processus de routage, mais ils offrent des vulnérabilités accrues aux attaques de types déni de service comme par exemple la consommation des ressources au niveau des nœuds pour un routage avec QoS.

## 4.2 Solutions envisageables

Partant de ce constat, il nous faudra offrir un environnement intelligent capable de s'adapter aux changements fréquents de la topologie afin de fournir une qualité de service optimale pour les flux multimédias. Ainsi, afin de résoudre ces différents problèmes et ainsi permettre une meilleure gestion des flux multimédias, nous proposons un mécanisme de routage ad hoc avec Qualité de Service basé sur les Systèmes Multi Agent et un mécanisme de sécurité pour parer aux attaques de type déni de service sensibles aux applications multimédias.

Pour atteindre cet objectif, nous allons nous appuyer sur les avantages des SMA que nous offrent le protocole ARA afin d'intégrer la qualité de service dans le routage des flux multimédia. Ces Systèmes Multi Agents nous permet de créer un environnement intelligent capable de prendre en compte les besoins en bande passante des applications multimédias (vidéos, audio, images…). En effet la technologie agent permet de résoudre des problèmes dont la complexité ne permet pas d'être résolus par les systèmes classique (voir le protocole ARA) car elle est fondée sur le principe de l'intelligence collective : bénéficier des compétences de chaque composant du système. Les agents par leur autonomie, leur adaptabilité, leur coopération, leur souplesse et leur réactivité sont incontournables dans la résolution des problèmes des systèmes distribués tels que les réseaux de télécommunications en général et les réseaux ad hoc en particulier.

L'algorithme de routage multi agent avec QoS que nous proposons est basé sur du routage réactif, plus exactement sur le protocole Multi agent ARA (Ant-colony based Routing Algorithm) mais selon une métrique autre que le traditionnel nombre de sauts. Il effectue une demande de route tout en réservant de la bande passante pour les flux multimédias prioritaires. L'intérêt de cet algorithme est qu'il essaye d'apporter de la QoS à tous les flux, en donnant à chaque mobile une connaissance du voisinage étendu dans le but d'une meilleure évaluation de la bande passante pour la réservation. Cela ne peut être effectif qu'avec l'apport des SMA.

Comme dans le protocole ARA, notre modélisation repose sur les deux types de paquets agent du mécanisme de découverte de route à savoir les paquets RREQ (Route REQuest) et RREP (Route REPlay) et permet de représenter le routage suivant la qualité requise et l'adaptation des services à l'état actuel du réseau. Ces paquets agents RREQ et RREP grâce à leurs propriétés Multi Agent seront à mesure de trouver des routes ayant suffisamment de ressources afin d'assurer un routage optimal des flux QoS et Best-Effort. Pour ce faire, ces paquets fournissent, au control d'admission des nœuds traversés, les paramètres tels que par exemple la bande passante sollicitée par les

applications multimédias afin de faire la réservation des ressources (bande passante) ou de la refuser.

La qualité de service ainsi proposée sera fournie suivant une différentiation de flux en deux classes de services : le service Best-Effort et le service avec QoS. La première classe est dédiée aux flux élastiques n'ayant aucune contrainte en termes de bande passante, délai, etc. La seconde quant à elle assure une réservation de bande passante à chaque flux les demandant et ayant présenté une requête admissible. Elle est basée sur l'architecture **IntServ** (Integrated Service) [ABE]. Pour assurer une qualité globale du réseau, les flux QoS seront privilégiés au détriment des flux Best-Effort.

*L'algorithme d'ordonnancement de cette architecture de différentiation de services est basé sur le principe de PQ (priority Queueing). En effet, la classe Service QoS est prioritaire à la classe Service Best-Effort et donc sera favorisée par rapport à cette dernière. Cependant, les files d'attente définies au sein de chaque classe, suivent la discipline FIFO.*

Pour la gestion de la securite du routage avec QoS dans les réseaux ad hoc, nous proposons un mécanisme de découverte de route pour cet algorithme qui se base sur les mécanismes de sécurité classique et solutions de sécurité adaptés à l'environnement ad hoc abordés dans la deuxième partie. Ainsi, on pourra assurer l'authentification et l'intégrité des paquets Route Request et Route Response lors de la phase de découverte de routes et de réservation de la bande passante.

Notre proposition se focalise surtout sur le mécanisme de Découverte de route en faisant une réservation de bande passante au niveau des nœuds source et destination en passant par les nœuds intermédiaires pour les flux QoS et en trouvant le plus court chemin entre la source et la destination pour les flux best-effort.

# PARTIE III : OPTIMISATION ET SECURISATION DU ROUTAGE MULTI AGENT DANS LES RESEAUX AD HOC

## *Chapitre 5: Mécanisme de routage avec QoS basé sur les SMA : ARAQ (ARA avec QoS)*

Le protocole de routage ARA ainsi que les autres protocoles du groupe Manet effectue un routage au mieux. C'est-à-dire la recherche de la meilleure route pour joindre une source à la destination se fait suivant une seule métrique à savoir le nombre de saut. Ce service fourni par ces protocoles est avantageux pour la transmission des flux best effort ne nécessitant aucune contrainte (bande passante, délai).

L'architecture proposée tente d'intégrer de la qualité de service aux routages des flux présentant des contraintes (délai, bande passante, gigue…). En effet, ces flux ne peuvent se contenter d'une route ayant un minimum de saut, ce qu'ils ont besoin ce sont des routes ayant suffisamment de ressource en bande passante. En apportant une solution à ce problème avec la réservation de la bande passante, nous résolvons du coup le problème de délai, de gigue, taux de perte etc. Car, ces métriques sont très étroitement liées. En se basant sur les Systèmes Multi Agents, le mécanisme effectue une différentiation de services. Il classifie les flux en deux classes : la classe des flux Best-effort et celle des flux demandant de la bande passante.

## 1.1 Apport des Systèmes Multi Agents au Routage QoS

### 1.1.1 Le marquage des routes par dépôt de phéromone

Afin respecter l'approche du SACOMA (Simple ant colony optimization meta-heuristic algorithm) décrit dans le protocole ARA, nous avons conservé le même principe dans ARA-Q afin de créer les routes phéromones et faire la maintenance de ces routes pendant la transmission des données. Et c'est là l'un des apports fondamentaux des Multi Agents, car en déposant cette phéromone sur les chemins empruntés, les paquets auront une idée des routes qui permettent de joindre un nœud.

Cette phéromone déposée sur les liens des nœuds joue le même rôle que le TTL (Time To Leave) des entrées des tables de routage dans le cas des protocoles de routage du groupe MANET. Tant que la quantité de phéromone est supérieure à zéro l'entrée est valide et le lien est susceptible d'être emprunté. Si la quantité de phéromone est égale à zéro, l'entrée est supprimée de la table de routage

Durant le processus de recherche de route, les fourmis déposent la phéromone sur les liens. Dans "SACOMA", les fourmis déposent une quantité constante $\Delta\varphi$ de phéromone. Les fourmis changent la quantité de phéromone sur les liens e(Vi, Vj) quand il se déplace du nœud Vi vers le nœud Vj.

$$\varphi_{i,j} = \varphi_{i,j} + \Delta\varphi \qquad (5)$$

Dépôt de phéromone sur les liens

Comme la vraie phéromone, la concentration artificielle diminue avec le temps empêchant ainsi une convergence rapide de phéromones sur les liens. Dans SACOMA, ceci se produit exponentiellement par:

$$\varphi_{i,j}(t+\tau) = (1-q).\varphi_{i,j}(t), \qquad q \in (0,1] \quad (6)$$

Diffusion de phéromone sur les liens

### 1.1.2 L'évaluation des métriques.

Dans le routage avec Qualité de Service, le control d'admission est l'étape la plus importante car permettant d'accepter ou de refuser une route QoS. Grâce aux propriétés SMA intégrés dans les paquets RREQ et RREP de ARA-Q, le nœud reçoit les métriques les plus récentes sur l'état des liens empruntées par les agents RREQ et RREP grâce à leur capacité d'interactivité. Ainsi, le control d'admission situé au niveau du nœud pourra faire le bon choix pour l'acceptation ou le refus de la demande de réservation et ainsi faciliter le bon fonctionnement du service QoS.

En effet, comme dit dans le chapitre trois, l'estimation des ressources est un point critique dans le routage QoS. Ainsi, grâce à l'approche locale apportée par les Multi agents, les nœuds seront à mesurer de faire une bonne estimation des ressources disponibles sur les liens avec leurs voisins afin de satisfaire les requêtes.

## 1.2 Description de l'algorithme ARA-QoS

Notre algorithme est basé sur le protocole de routage Multi Agent ARA qui s'inspire lui de AODV. De ce fait, le formatage des Paquets Requête de Route (RREQ) et Réponse de Route (RREP) a été respecté. Il est à souligner que l'intégration de la notion d'agent au protocole ARA nous a permis de mieux modéliser notre approche et d'ajouter de façon plus concise cette fonctionnalité de qualité de service.

Comme dans le protocole ARA, nous nous appuyons sur les deux agents RREQ (Route Request) et RREP (Route Replay) qui sont dotés de tous les propriétés des agents classiques tels que la réactivité, la flexibilité, l'interactivité et la capacité reproductive. En effet ces propriétés Multi Agent des paquets RREQ et RREP seront les clés de notre apport de Qualité de Service au routage dans le Protocole ARA. Ces paquets pourront interagir avec le control d'admission des nœuds et ainsi, lui transmettre à temps réel les paramètres nécessaires pour le choix d'une route QoS satisfaisant la demande en bande passante d'une requête.

Comme tout routage réactif, notre mécanisme de recherche de route comporte deux phases : la découverte de route par les paquets agents RREQ et RREP et leur maintenance par les paquets de données.

### 1.2.1 La découverte de route

Notre processus de découverte de route ne s'effectue pas de la même manière que le protocole ARA. Le fonctionnement des paquets RREQ et RREP et des nœuds a été modifié pour mieux traiter les requêtes de demande de route QoS. Il effectue une différentiation de service entre les flux élastiques n'ayant aucune contrainte et appelés **flux Best Effort** et le flux ayant un besoin réel en bande passante, appelés ici **flux QoS**. Alors que les flux Best Effort sont servis selon le service fourni par le protocole ARA, les flux QoS eux, font l'objet d'une réservation de bande passante selon le modèle **IntServ** [ABE]. C'est-à-dire que chaque flux QoS formule sa requête contenant la quantité de bande passante dont il a besoin. Selon la disponibilité des ressources du réseau, cette demande lui sera refusée ou accordée. Auquel cas il lui est fait une réservation de bande passante.

La différentiation de services est réalisée par le nœud source à l'initialisation de la phase de recherche de route. Cette classification repose sur le champ QoS *id_flux* qui correspond au champ ToS (Type of Service) des paquets IPv4.

- Si c'est une requête pour un flux Best Effort alors le champ ***id_flux*** du paquet RREQ est initialisé à 0 ainsi que le champ ***Bande_passant_demandée***

- Si c'est une requête pour un flux QoS alors le champ ***id_flux*** est égal à 1 et le champ ***Bande_passante_demandée*** est égal à la bande passante exigée par l'application

Le nœud source formate le paquet agent RREQ avec ces différentes valeurs et initialise la quantité de phéromone qu'il va déposer sur chaque lien emprunté. Le nœud émetteur diffuse ensuite le paquet agent RREQ pour établir la route de phéromone à partir de la source, c'est-à-dire la route le plus optimale pour atteindre la destination.

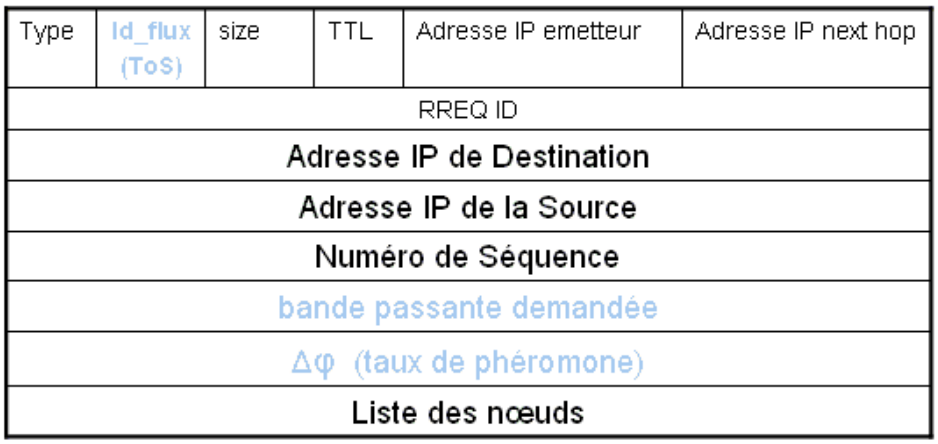



**Figure 16. Format paquet RREQ**

Si un nœud reçoit un paquet RREQ, il vérifie d'abord le type de paquet :

- si c'est une requête pour un flux **Best Effort**, il crée autant d'enregistrement dans sa table de routage que de voisins, cette entrée contient ***[@destination, next hop, bp-reservée, φ]***. Le nœud interprète l'adresse de destination du paquet RREQ comme "***@destination",*** l'adresse du nœud suivant c-à-d l'adresse du nœud vers qui le paquet RREQ sera émis comme "***next hop",*** le champ "***bp-reservée"*** sera égal à 0 car aucune réservation n'est faite pour ce type de requête, on ajoute la valeur de phéromone au champ **"φ".** Ensuite, le nœud retransmet en diffusion le paquet RREQ à tous ces voisins sauf le nœud précédent jusqu'à atteindre la destination.

- Si c'est une requête pour un **flux QoS**, le nœud effectue un *control d'admission au terme duquel il décide de réserver de la bande passante pour le flux concerné sur les liens avec ses voisins ou de refuser*. En effet, si la bande passante demandée par le flux est supérieure à celle restante sur le lien d'un nœud avec un voisin alors la réservation est rejetée et le paquet RREQ est supprimé. Avant de retransmettre le paquet RREQ à ses voisins, le nœud crée l'entrée dans sa table de routage comme dans le cas d'une requête Best Effort mais cette fois si seulement pour les liens admissibles qui ont été acceptés par le control d'admission. Cette fois ci, le champ "***bp_reservée"*** sera égal à la bande passante demandée par le flux et contenue dans le champ ***Bande_passante_demandée*** du paquet RREQ.

**Bp_restante <= b_passante résiduelle -∑ b_passante réservée**

Bande passante restante sur un lien

Comme dans le protocole ARA, le numéro de séquence contenu dans le paquet RREQ permet de déceler les doublons afin d'éviter les boucles de routage en les supprimant.

| @ destination | Next hop | B. Passante réservé | **phéromone** |
|---|---|---|---|
| | | | |

**Figure 17. Table de routage d'un nœud dans ARA-Q**

Seuls les paquets qui ont empruntés les routes les plus optimales, c'est-à-dire les chemins les plus rapides et ayant suffisamment de bande pas sante demandée, arrivent sur le nœud destination. Mais à cause du numéro de séquence des paquets RREQ seul le premier paquet arrivé est traité et les autres ignorés. Comme dans le protocole ARA, la destination envoie un paquet

RREP en diffusion vers la source tout en déposant une quantité de phéromone sur les liens traversés. Pour ce faire, le nœud de destination récupère les informations contenues dans le paquet RREQ et formate le paquet RREP. Le champ **adresse de destination** sera l'adresse source du paquet RREQ, le champ **adresse de source** sera l'adresse du nœud de destination, le champ **id_flux** aura la même valeur que l'id_flux du paquet RREQ reçu ainsi que les champs **taux de phéromone** et **bp_réservée** avant d'envoyer le paquet en diffusion vers le nœud source.

Le fonctionnement du nœud à la réception du paquet agent RREQ peut se représenter par cet algorithme. (Voir page suivante)

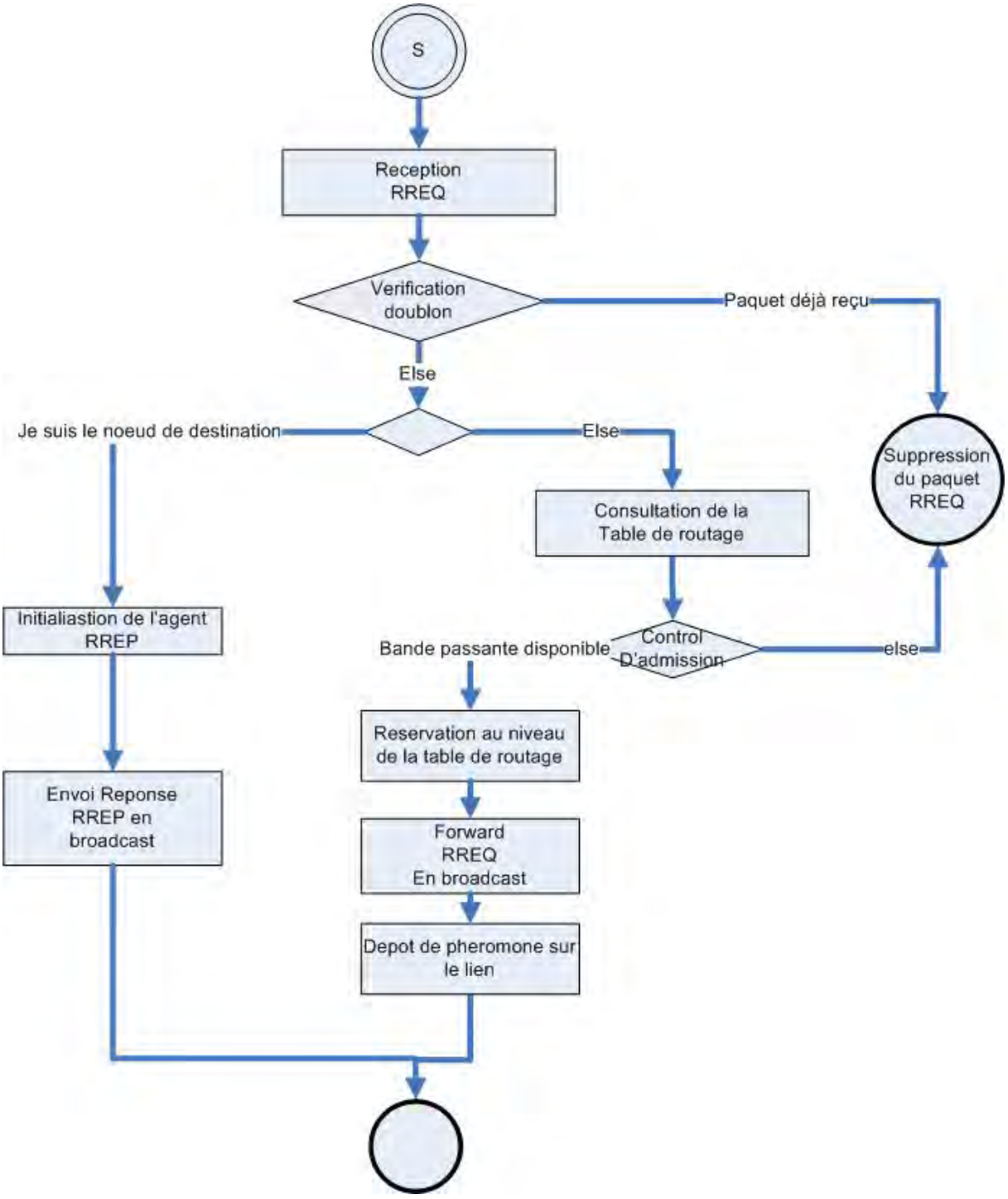



**Figure 18. Fonctionnement du nœud a la réception du paquet RREQ**

Si le paquet RREP arrive sur un nœud, il vérifie si l'entrée du lien empruntée existe dans sa table de r outage. C'est-à-dire l'entrée ayant comme "***@destination"*** l'adresse source du pa quet RREP, et comme "***next hop"*** l'adresse de l'émetteur du paquet RREP.

- Si l'entrée existe, il *incrémente la quantité de phéromone* se trouvant sur cette entrée de l a table de r outage avant de r éémettre le paquet en diffusion.

- Si l'entrée n'existe pas ce qui signifie que c'est un nouveau nœud qui a reçu ou émis le paquet RREP, *alors le nœud le traite comme si c'était un paquet RREQ,* en créant l'entrée dans sa table de routage et en sollicitant le control d'admission afin d'accepter ou refuser la demande de réservation de la bande passante dans le cas d'une requête pour un flux QoS.

L'algorithme ci-dessous résume le comportement du nœud à la réception du paquet agent RREP. (Voir page suivante)

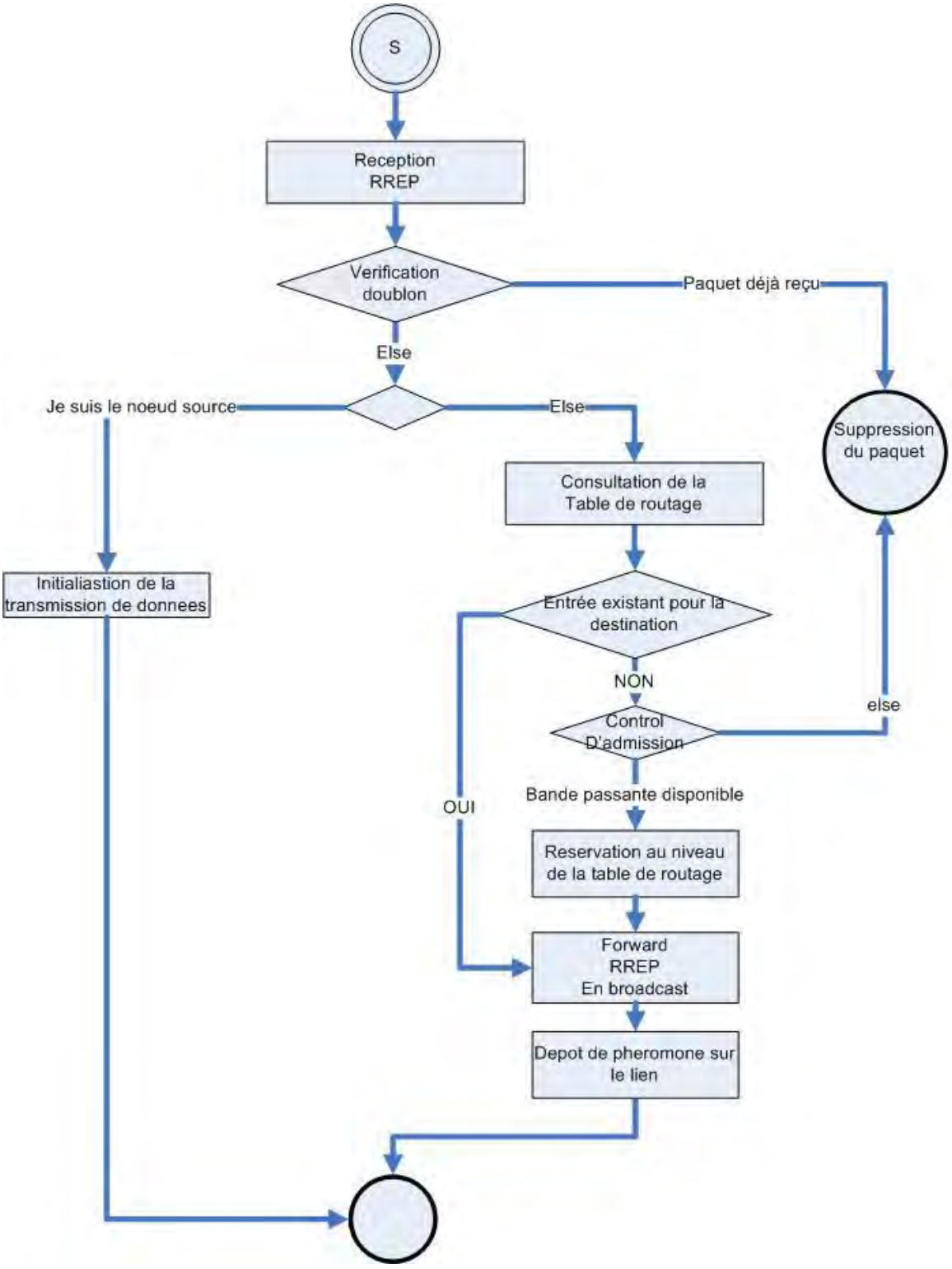



**Figure 19. Comportement du nœud à la réception d'un paquet agent RREP**

Quand le paquet arrive au nœud de destination, il aura traversé tous les nœuds et liens optimaux qui permettent d'atteindre la destination. Cependant, au niveau de la table de routage de ces nœuds, on aura l'ensemble des entrées qui permet de joindre la destination avec des taux de phéromone différents. Celle ayant le taux de phéromone le plus élevé représente le lien le plus rapide pour atteindre la destination et ainsi de suite.

### 1.2.2 La maintenance des routes

La seconde phase de l'algorithme est appelée maintenance de route, elle se déroule de la même manière que le protocole ARA. Une fois que les RREQ et RERP établissent les chemins de phéromone entre la source et la destination, les paquets de données "transmis" sont utilisés pour assurer la maintenance. Par analogie à la nature, les chemins établis ne gardent pas leurs valeurs de phéromone de façon permanente. Quand un nœud Vi relaie un paquet de données vers la destination à un nœud voisin Vj, il incrémente la valeur de phéromone de l'entrée (Vd, Vj, $\varphi$) par $\Delta\varphi$ c'est-à-dire le chemin vers la destination est renforcé par les données de paquets.

### 1.2.3 La détection d'une erreur de route

Si un nœud reçoit un message de ROUTE_ERROR pour un certain lien, il enregistre d'abord ce lien en mettant la valeur de phéromone et le champ bande passante réservée à 0. Plus tard, le nœud recherche un lien alternatif dans sa table de routage. S'il y a un autre itinéraire à la destination elle enverra le paquet par l'intermédiaire de ce chemin. Autrement, le nœud informe ses voisins, espérant qu'ils pourront expédier le paquet vers la destination. Si le paquet n'atteint pas la destination, le nœud de source doit lancer un nouveau procédé de découverte d'itinéraire.

## 1.3 Implémentation et Validation du mécanisme dans NS-2

Afin de valider notre algorithme de routage avec QoS basé sur les Systèmes Multi Agent, nous avons utilisé le Simulateur NS-2 pour l'implémenter et ainsi mesurer les critères de performances du mécanisme de réservation de routes QoS comme le débit des paquets dans le réseau, le délai de transmission des paquets, les entrées des tables de routage des nœuds etc.

### 1.3.1 Le Simulateur NS-2:

#### *A) Définition*

***NS*** est un outil de simulation de réseaux [FAL]. Il est bâti autour d'un langage de programmation appelé ***Tcl*** dont il est une extension. Du point de vue de l'utilisateur, la mise en œuvre de ce simulateur se fait via une étape de programmation qui décrit la topologie du réseau et le comportement de ses composants, puis vient l'étape de simulation proprement dite et enfin l'interprétation des résultats. Cette dernière étape peut être prise en charge par un outil annexe, appelé ***Nam*** qui permet une visualisation et une analyse des éléments simulés.

NS est un simulateur à événements discrets orienté objet. Il est écrit en C++ avec une interface textuelle (ou shell) qui utilise le langage OTcl (Object Tool Command Langage). L'OTcl est une extension objet au langage de commande Tcl. Le langage C++ sert à décrire le fonctionnement interne des composants de la simulation. Pour reprendre la terminologie objet, il sert à définir les classes.

Quant au langage OTcl, il fournit un moyen flexible et puissant de contrôle de la simulation comme le déclenchement d'événements, la configuration du réseau, la collecte de statistiques, etc. L'application NS se compose de deux éléments fonctionnels: un interpréteur et un moteur de simulation. Au moyen de l'interpréteur l'utilisateur est capable de créer le modèle de simulation ce qui revient à assembler les différents composants nécessaires à l'étude. Les composants du modèle de simulation sont appelés objets ou encore instances de classe. Le moteur de simulation effectue les calculs applicables au modèle préalablement construit par l'utilisateur via l'interpréteur.

***NS*** est en réalité un programme relativement complexe écrit en C++ et interfacé via ***Tcl***. Pour modifier le comportement d'objets existants, il est donc nécessaire de modifier le code C++ qui en réalise l'implantation.

NS bénéficie de toutes les possibilités qu'offrent les techniques objets comme l'héritage, le polymorphisme, la surcharge, etc. L'héritage permet d'élaborer des arborescences de classes. Le modèle de simulation est construit à partir d'une arborescence de classes qui en fait se dédouble:

- Une définie en OTcl dite arborescence interprétée. Elle est utilisée par l'interpréteur et est visible par l'utilisateur.
- Une définie en C ++ que l 'on nommera compilée. Elle forme l'arborescence utilisée par le moteur de simulation (que l'on appellera par la suite simulateur). C'est l'ombre de l'arborescence interprétée.

La distribution de NS comprend principalement 3 répertoires [HOR]:

- ns-2, l'application NS. Ce répertoire contient l'ensemble des fichiers .h et .cc de NS.
- nam-1, l'outil de visualisation des résultats de l a simulation: l'animateur réseau.
- tclcl, sources du code assurant la liaison entre l'interpréteur et le simulateur. Citons l'un des principaux fichiers: tcl-object.tcl.

Dans le répertoire ns-2, on trouve les répertoires:

- tcl pour tous les codes interprétés
- bin pour les utilitaires et les exécutables pour la réalisation du binaire ns-2
- lib pour la bibliothèque de libg++
- gen pour les sources générées lors de la réalisation du binaire ns-2 par le makefile.
- test_output pour les résultats des simulations.
- tous les fichiers .h et .cc des classes C++ du simulateur.

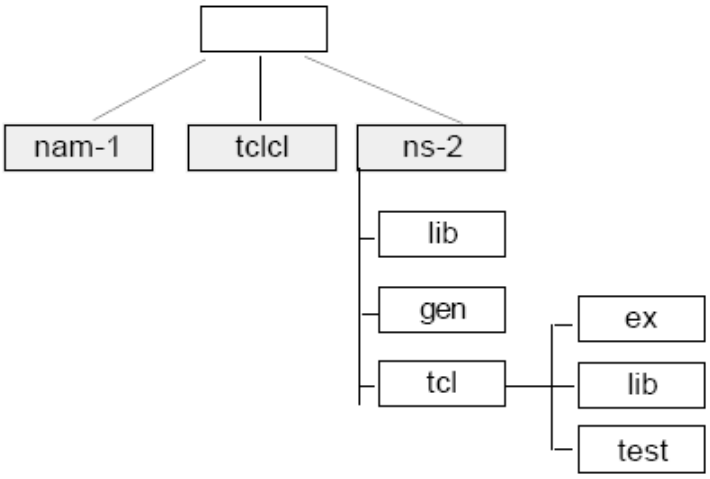



**Figure 20. Arborescence des fichiers de NS-2**

## *B) Architecture Réseau*

Les classes de b ases utilisables permettant de déf inir l'architecture et la topologie du modèle sont les "Node" et "Link". Ils servent à la composition de la topologie du réseau. Elles modélisent les nœuds et les arcs d'un graphe.

### a) Les nœuds

La classe "Node" est une classe OTcl. Elle n'a donc pas d'existence en tant que telle dans le simulateur. Cette classe et ses méthodes sont définies dans le fichier tcl/lib/ns-node.tcl. Un nœud est une c ollection de *classifiers* et *d'agents*. Le c lassifier démultiplexe les paquets. L'agent est habituellement l'entité d'un protocole. L'assemblage des composants est représenté par la figure ci-dessous.

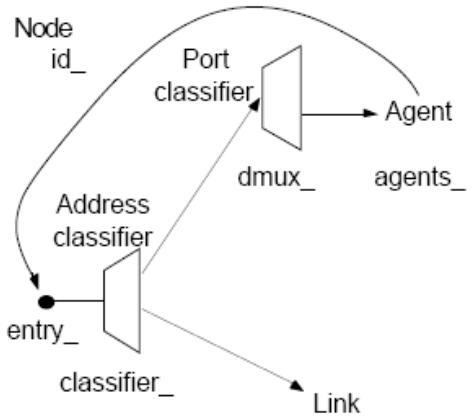



**Figure 21. Composant d'un Node**

Le trafic émis par l'agent est transmis au point d'entrée du nœud. Que ce soit des paquets reçus de ses voisins ou émis par ses agents, le traitement effectué par le nœud reste identique. Quand un nœud reçoit un p aquet, il examine les champs du paquet (habituellement sa destination) et le commute vers la bonne

interface de s ortie. Cette tâche est exécutée par le classifier. La table des adresses du classifier est remplie par la procédure de r outage (add-routes{}). Celle du "port multiplexer" se remplit à chaque ajout d'un agent au nœud.

Un "*classifier*" a pour objectif de r etrouver une référence à un au tre objet de la simulation à partir d'une comparaison sur un critère dont la valeur est contenue dans le paquet. Il a un rôle de démultiplexeur en quelque sorte. Le classifier sert notamment pour un nœud à modéliser la fonction de relayage (forwarding).

Quand un paquet est reçu par un nœud, ce dernier le conserve, détermine sa route et l'envoie à un autre NsObject. Il n'y a pas de durée ou de temps consommé pour ces opérations. Pour prendre en compte ces temps, il convient d'ajouter un élément de délai dans le nœud comme c'est fait pour le lien.

### b) Les Liens

Le lien sert à relier les nœuds. Il modélise le système de transmission. Le lien est principalement caractérisé par un délai de propagation et une bande passante. C'est une classe OTcl qui regroupe un ensemble de composants dérivés de la classe *Connector*. Cette classe et ses méthodes sont définies dans le fichier tcl/lib/ns-link.tcl. Des liens plus sophistiqués peuvent être dérivés de cette classe. Quelque soit le type du lien, il comporte 5 références sur des objets qui le composent. Le lien peut se représenter selon la figure suivante:

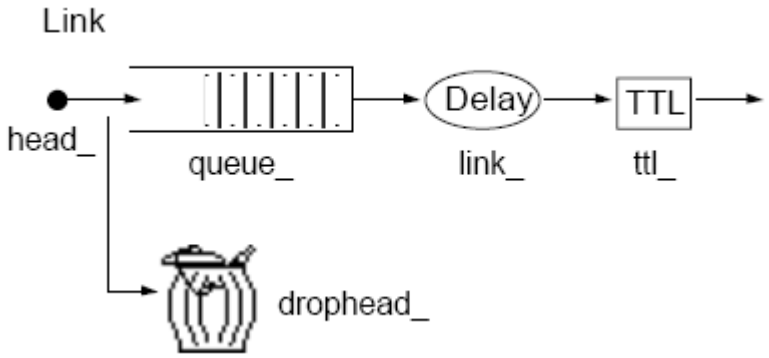



**Figure 22. Composantes d'un lien**

Les 5 instances variables suivantes ont la sémantique:

- head_, point d'entrée du lien, il pointe sur le premier objet du lien.
- queue_, référence la file d'attente de p aquets. Cette file modélise celle que l'on trouve habituellement aux interfaces de sortie.
- link_, référence l'élément qui caractérise le lien souvent en termes de délai et bande passante.
- ttl_, référence l'élément qui manipule la durée de vie de chaque paquet.
- drophead_, référence l'élément qui traite les pertes de paquets au niveau de la file d'attente.

Parmi les variables du lien, une référence est gardée sur le nœud amont (fromNode_) et sur le nœud aval (toNode_).

Tous les objets du lien sont dérivés de la classe C++ Connector. Un Connector, à la différence d'un classifier, envoie les paquets à un seul récepteur. Dans le cas normal le Connector a un voisin connu sous la référence Connector::target_.Si il doit détruire le paquet il le passe à l'objet référencé par Connector::drop_. En somme, le Connector délivre le paquet à son voisin indiqué par target_ ou drop_. Si drop_ est indéfini, le paquet à jeter est supprimé de la simulation. target_ doit être défini sous peine d'obtenir un "Segmentation fault" lors de son utilisation à la simulation. L'initialisation de ces variables se fait par les commandes OTcl target ou drop-target qui sont définies par la procédure Connector::command().

### c) L'Agent

L'agent est un autre composant d'un nœud. Il modélise les constructeurs et les consommateurs de paquets IP. La classe agent fournit des méthodes utiles au développement de la couche transport et à d'autres protocoles du plan de signalisation ou de gestion. Cette classe est à la fois dans l'interpréteur et dans le simulateur. C'est la classe de base pour définir des nouveaux protocoles dans NS. Elle fournit l'adresse locale et de destination, les fonctions pour générer les paquets, l'interface à la classe Application. Actuellement NS comporte de nombreux agents citons: UDP, protocoles de routage, différentes versions de TCP, RTP, etc.

Avec l'évolution de NS, le rôle de l'agent a évolué. Dans des versions passées, l'agent faisait également office de source de trafic. C'est la raison pour laquelle on a la classe Agent/CBR ou Agent/UDP/CBR. Ces deux classes sont identiques. Il n'est pas conseillé d'utiliser ces classes actuellement. La bonne démarche est d'utiliser la classe Agent/UDP et la classe Application/Traffic/CBR. La séparation de la source de trafic de l'agent apporte une plus grande flexibilité.

## 1.3.2 Implémentation de l'algorithme de routage ARA-Q dans NS-2

Pour l'implémentation de notre algorithme de routage Multi Agents avec QoS (ARAQ), nous avons modifié les codes sources ARA (ANTNET) afin de lui ajouter les fonctionnalités de différenciation de flux, de réservation de bande passante au niveau des nœuds pour les requêtes la sollicitant. Notre projet ARAQ comporte 8 fichiers source écrits en C++ définissant le comportement des nœuds et des agents RREQ et RREP durant tout le processus de découverte de route. Ainsi, nous avons fait les modifications suivantes:

- Dans le fichier "**araq_pkt.h**" qui définit le format d'un paquet ARAQ (RREQ et RREP), on a ajouté les champs Id_flux pour la différenciation de flux et b_reservé pour la réservation de la bande passante.
- dans les fichiers sources "**araq.h**" et "**araq.cc**": on définit le comportement des agents RREQ et RREP et leur interaction avec le

control d'admission du nœud. Dans ces classes, nous avons ajouté les méthodes permettant au c ontrol d'admission, en s e basant sur les paramètres fournis par les agents RREQ et RREP, d'accepter ou d e refuser la requête.

```
//////////////////////////////////////////////////////////////////////////
/// initialisation des paramètres de l'agent
/// avec les paramètres par défaut définit dans ns-default.tcl
//////////////////////////////////////////////////////////////////////////
 araq::araq(nsaddr_t id) : Agent(PT_ARAQ), araq_timer_(this), dmux_(0) {

        bind("num_nodes_", &num_nodes_);     // nombre de noeuds de la
topologie
        bind("r_factor_", &r_factor_);       // valeur de phéromone
        bind("timer_ant_", &timer_ant_);     // fréquence de génération des
agents
        bind("b_demande_", &b_demande_);     //bande passante sollicitée
        bind("b_dispo_", &b_dispo_);         //bande passante du lien
        bind("id_flux_pkt_", &id_flux_pkt_); // type de flux
}
```

**Figure 23. Méthode d'initialisation de l'objet ARAQ**

Cette méthode ci-dessus appelle le fichier ns-default.tcl et récupère les lignes ou on a défini les paramètres concernant les agents.

```
////////////////////////////////////////////////////////////////////////////////
/// Méthode pour retransmettre un Paquet RREQ vers la destination
////////////////////////////////////////////////////////////////////////////////
void araq::rreq_pkt(Packet* p) {
        struct hdr_ip* ih = HDR_IP(p);                  // ip header
        struct hdr_cmn* ch = HDR_CMN(p);// common header
        struct hdr_araq_pkt* ah = HDR_ARAQ_PKT(p);// ant header

        nsaddr_t parent = ih->saddr();          // nœud parent
        nsaddr_t next = rtable_.calc_next(addr(), ah->pkt_dst(), parent);
        if(next == addr() || next == parent) {
                Packet::free(p);
                return;
        }
        int b_total_next=rtable_.b_total_alloue(next); //b.p totale alloue sur un lien
        int b_restant = b_dispo - b_total_next;
        if(b_demande_< b_restant){
                ch->next_hop() = next;          // ajout du next hop dans le common
header
                ih->saddr() = addr();   // ajout de la source dans ip header
                ih->daddr() = next;             // ajout de la destination dans ip header
                // send packet to next hop node
                rtable_.update_bp(ah->pkt_dst(), next, b_demande_);
                target_->recv(p);
        }else{
                Packet::free(p);
                return;
        }
}
```

**Figure 24. Retransmission d'un RREQ par un nœud dans ARAQ**

- dans les fichiers araq_rtable.h et araq_rtable.cc qui définissent le fonctionnement de la table de routage, on a modifié la méthode d'initialisation de la table de routage afin qu'elle prenne en compte le champ de réservation de la bande pas sante. Cette classe sera implicitement appelée par cotre control d'admission afin de choisir les routes QoS.

```
////////////////////////////////////////////////////
/// Méthode pour ajouter une entrée dans la table de routage
////////////////////////////////////////////////////
void araq_rtable::add_entry(nsaddr_t dest, nsaddr_t next, double phvalue, int
reservation) {
        struct pheromone temp_pheromone;      // enregistrement de phéromone
                                                (annexe)
        temp_pheromone.neighbor = next;                 // adresse du nœud voisin
        temp_pheromone.phvalue = phvalue;     // la valeur de phéromone
        temp_pheromone.reservation=reservation;       // la bande passante
reservée
        rtable_t::iterator iterRt = rt_.find(dest);
        if(iterRt == rt_.end()) {       // destination n'existe pas on ajoute une
```

```
nouvelle entrée
                pheromone_matrix temp;
                temp.push_back(temp_pheromone);
                rt_[dest] = temp;
        }
        else {  // destination existe dans la table, on fait la mise à jour du next hop
                pheromone_matrix *temp = &((*iterRt).second);
                temp->push_back(temp_pheromone);
        }
}
```

**Figure 25. Ajout d'une entrée dans la table de routage de ARAQ**

Pour la maintenance des routes, c'est-à-dire la persistance et la cohérence des routes par le dépôt de phéromone, nous avons conservé la méthode ci-dessus de ARA:

```
////////////////////////////////////////////////////////////////////////////////
/// Method to update routing table
/// Parameters:
/// - adresse noeud de destination
/// - adresse noeud voisisn
////////////////////////////////////////////////////////////////////////////////
void araq_rtable::update(nsaddr_t dest, nsaddr_t next) {

  pheromone_matrix *vect_pheromone;
  pheromone_matrix temp;

  // recherche de la destination dans la table de routage
  rtable_t::iterator iterRt = rt_.find(dest);
  if(iterRt != rt_.end()) {
       vect_pheromone = &((*iterRt).second);
        pheromone_matrix::iterator iterPh = vect_pheromone->begin();
        for(; iterPh != vect_pheromone->end(); iterPh++) {
                double oldph = (*iterPh).phvalue;
                if((*iterPh).neighbor == next)
                        (*iterPh).phvalue = oldph + r*(1 - oldph);  aj out de
phéromone
                else
                        (*iterPh).phvalue = (1-r)*oldph;  //  evaporation  du
phéromone
        }
  }
}
```

**Figure 26. Mise à jour des routes dans ARAQ**

Après ces modifications sur les fichiers source, nous avons intégré le protocole dans NS-2. Pour ce faire nous avons édité quelques fichiers tcl de NS-2

- dans le fichier common/packet.h, nous avons ajouté le nouveau type de paquet PT_ARAQ et le nom "araq" dans le constructeur p_info() de la classe.
- Nous avons édité le fichier trace/cmu-trace.h et trace-cmutrace.cc pour le format de trace du protocole ARAQ
- On définit des priorités aux paquets RREQ et RREP dans queue/priqueue.cc

```
Void  PriQueue : : recv ( Packet  *p , Handler *h)
{
        s t r u c t hdr_cmn *ch = HDR_CMN(p ) ;
        i f ( Prefer_Routing_Protocols ) {
                switch ( ch−>ptype ( ) ) {
                        case PT_DSR:
                        case PT_MESSAGE:
                        case PT_TORA:
                        case PT_AODV:
                                recvHighPriority (p , h ) ;
                                break ;
                        case PT_ARAQ:
                                i f ( ch−>direction () == hdr_cmn : :UP) { //RREP
                                        recvHighPriority(p, h) ;
                                }
                                el se {
                                        Queue : : recv(p, h) ;   //RREQ
                                }
                                break ;
                        d e f a u l t :
                                Queue :: recv (p , h ) ;
                }
        }
        e l s e {
                Queue :: recv (p , h ) ;
        }
}
```

**Figure 27. Priorité aux paquets RREQ et RREP dans ARAQ**

- Ajout du nouveau protocole ARAQ dans tcl/libns-packet.tcl.

- On définit les paramètres par défaut de l 'agent ARAQ dans tcl/lib/ns_default.tcl
  - o r_factor_ : valeur de phéromone
  - o b_demande_ : bande passante sollicitée
  - o Id_flux_ : id flux (BE ou QoS)

- Dans tcl/lib/ns-lib.tcl, on ajoute les méthodes de création de l'agent ARAQ lors de la simulation

```
Simulator instproc create−araq−agent { node } {
        set ragent [new Agent/ARAQ [ $node node−addr ] ]
        $sel f at 0.0 ”$ragent start”
        $node set ragent $ragent
        return $ragent
}
```

**Figure 28. Création de l'agent ARAQ dans NS-2**

- pour prendre en compte les fichiers sources de A RAQ dans NS-2 pendant la compilation, on ajoute leurs fichiers objets dans le Makefile

```
OBJ CC = \
tools/random.o tools/rng.o tools/ranvar.o \
common/misc.o common/timer−handle r.o \
#. . .
araq/araq_common.o araq/araq_rtable.o araq/araq.o \
$ (OBJ STL)
```

**Figure 29. Modification du Makefile de NS-2**

Après, on compile notre nouveau NS-2 avec le protocole ARAQ

### 1.3.3 Validation de l'algorithme

Pour la validation de notre algorithme de routage Multi agents avec QoS, nous avons fait une simulation avec 12 nœuds dont les nœuds de 0 à 5 font une réservation de bande passante et le reste des nœuds font une demande de route Best Effort.

Les paramètres de simulation

- N = 12 nœuds
- Temps de simulation = 11 sec
- Intervalle d'envoi des paquets= 0.03 sec
- Bande Passante des liens = 512 Mb
- Demande de routes QoS pour les nœuds
  **0 ; 1 ; 2 ; 3 ; 4 ; 5**
- Bande Passante réservée 50 Mb
- Demande de routes élastiques pour les nœuds : **6 → 11**

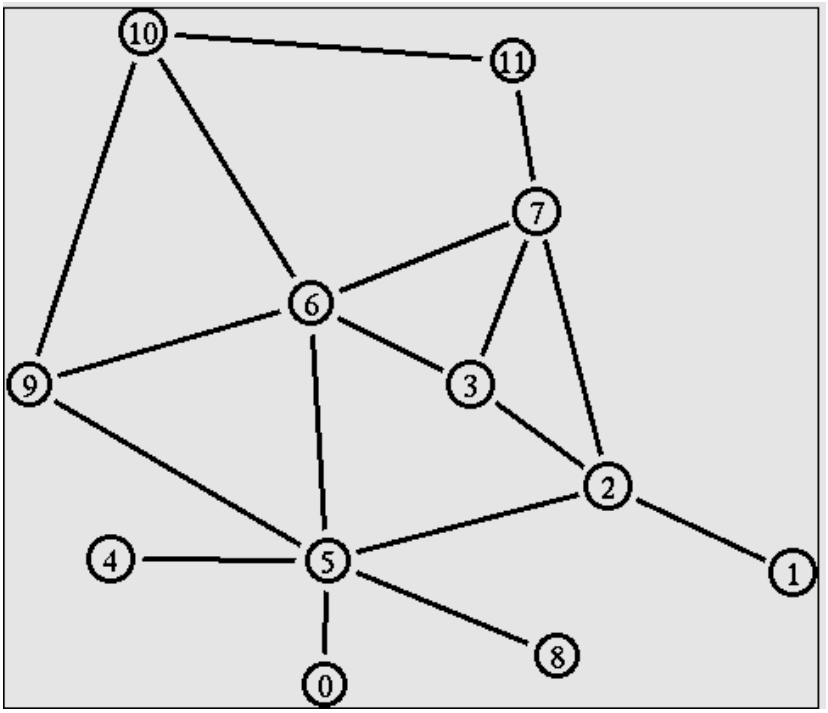



**Figure 30. Topologie de la simulation**

Comme pour les autres algorithmes, nous nous sommes focalisés en premier lieux sur le bon fonctionnement du mécanisme de recherche de routes que ça soit pour les routes élastiques ou QoS. Pour cela, nous avons analysé les entrées des tables de routage au niveau des nœuds.

**Table de routage nœud 2**

| dest | next | phvalue | b.reser |
|---|---|---|---|
| 0 | 7 | 0.250000 | 50 |
| 0 | 5 | 0.250000 | 0 |
| 0 | 3 | 0.250000 | 150 |
| 0 | 1 | 0.250000 | 50 |
| . | . | . | . |
| . | . | . | . |
| 8 | 1 | 0.250000 | 0 |
| 9 | 7 | 0.237500 | 100 |
| 9 | 5 | 0.237500 | 0 |
| 9 | 3 | 0.287500 | 100 |
| 9 | 1 | 0.237500 | 100 |
| 10 | 7 | 0.323125 | 200 |
| 10 | 5 | 0.225625 | 0 |
| 10 | 3 | 0.225625 | 0 |
| 10 | 1 | 0.225625 | 50 |
| 11 | 7 | 0.323125 | 0 |
| 11 | 5 | 0.225625 | 100 |
| 11 | 3 | 0.225625 | 50 |
| 11 | 1 | 0.225625 | 50 |

**Table de routage nœud 3**

| dest | next | phvalue | b_reser |
|---|---|---|---|
| 0 | 7 | 0.333333 | 150 |
| 0 | 6 | 0.333333 | 200 |
| 0 | 2 | 0.333333 | 0 |
| 1 | 7 | 0.300833 | 50 |
| . | . | . | . |
| . | . | . | . |
| 4 | 2 | 0.333333 | 100 |
| 5 | 7 | 0.316667 | 0 |
| 5 | 6 | 0.366667 | 50 |
| 8 | 2 | 0.333333 | 50 |
| 9 | 7 | 0.316667 | 50 |
| 9 | 6 | 0.366667 | 100 |
| 9 | 2 | 0.316667 | 200 |
| 10 | 7 | 0.316667 | 0 |
| 10 | 6 | 0.316667 | 100 |
| 10 | 2 | 0.366667 | 0 |
| 11 | 7 | 0.348333 | 150 |
| 11 | 6 | 0.300833 | 50 |
| 11 | 2 | 0.350833 | 50 |

**Table de routage noeud 5**

| dest | next | phvalue | b_reser |
|---|---|---|---|
| 0 | 9 | 0.059748 | 100 |
| 0 | 8 | 0.059748 | 50 |
| 0 | 6 | 0.059748 | 0 |
| 0 | 4 | 0.059748 | 50 |
| 0 | 2 | 0.059748 | 50 |
| 0 | 0 | 0.701262 | 100 |
| 1 | 9 | 0.158333 | 0 |
| 1 | 8 | 0.158333 | 0 |
| 1 | 6 | 0.158333 | 50 |
| 1 | 4 | 0.158333 | 50 |
| 1 | 2 | 0.208333 | 0 |
| 1 | 0 | 0.158333 | 50 |
| 2 | 9 | 0.066202 | 0 |
| 2 | 8 | 0.066202 | 100 |
| 2 | 6 | 0.066202 | 50 |
| . | . | . | . |
| 3 | 4 | 0.135751 | 150 |
| 3 | 2 | 0.233251 | 50 |
| 3 | 0 | 0.135751 | 0 |
| 4 | 9 | 0.073354 | 100 |
| 4 | 8 | 0.073354 | 0 |
| 4 | 6 | 0.073354 | 0 |
| 4 | 4 | 0.633228 | 100 |
| 4 | 2 | 0.073354 | 100 |
| 4 | 0 | 0.073354 | 100 |
| 6 | 9 | 0.056760 | 0 |
| . | . | . | . |
| . | . | . | . |
| 8 | 2 | 0.110570 | 0 |
| 8 | 0 | 0.110570 | 100 |
| 9 | 9 | 0.583987 | 50 |
| 9 | 8 | 0.077215 | 0 |
| 9 | 6 | 0.077215 | 0 |
| 9 | 4 | 0.077215 | 0 |
| 9 | 2 | 0.107152 | 200 |
| 9 | 0 | 0.077215 | 100 |
| 10 | 9 | 0.208333 | 50 |
| 10 | 8 | 0.158333 | 0 |
| 10 | 6 | 0.158333 | 0 |
| 10 | 4 | 0.158333 | 100 |
| 10 | 2 | 0.158333 | 50 |
| 10 | 0 | 0.158333 | 0 |
| 11 | 9 | 0.208333 | 150 |
| 11 | 8 | 0.158333 | 100 |
| 11 | 6 | 0.158333 | 150 |
| 11 | 4 | 0.158333 | 0 |
| 11 | 2 | 0.158333 | 0 |
| 11 | 0 | 0.158333 | 0 |

**Table de routage nœud 6**

| dest | next | phvalue | b_reser |
|---|---|---|---|
| 0 | 10 | 0.190000 | 50 |
| 0 | 9 | 0.190000 | 0 |
| 0 | 7 | 0.190000 | 100 |
| 0 | 5 | 0.240000 | 50 |
| 0 | 3 | 0.190000 | 0 |
| 1 | 10 | 0.200000 | 100 |
| 1 | 9 | 0.200000 | 50 |
| 1 | 7 | 0.200000 | 50 |
| 1 | 5 | 0.200000 | 0 |
| 1 | 3 | 0.200000 | 50 |
| 2 | 10 | 0.200000 | 50 |

**Figure 31. Extrait des tables de routage des nœuds**

Ces tables de routage nous donnent une idée sur la validité du protocole. En effet, on voit que durant cette phase de découvertes de route, les nœuds, grâce aux sollicitations des agents RREQ et RREP, arrivent à faire une réservation de bande passante sur les liens qui permettent d'atteindre une destination. Et nous verrons dans le fichier ci dessus des tables de routages que la somme des réservations pour un lien est toujours inferieur à la bande passante disponible sur ce lien.

L'apport multi path des SMA au routage QoS et représenté par le fait que pour une même destination, nous avons plusieurs routes classées par ordre de quantité de phéromone déposée. Cette approche, notons le, apporte une très grande réactivité face à la forte mobilité des réseaux ad hoc.

Comme c'est un algorithme de routage qui tente d'apporter de la QoS avec réservation de la bande passante pour la transmission des données, il serait important de mesurer l'impact du mécanisme de recherche de routes sur la bande passante disponible du réseau, c'est-à-dire sur le taux de saturation du réseau par les paquets lors de la phase de recherche de routes.

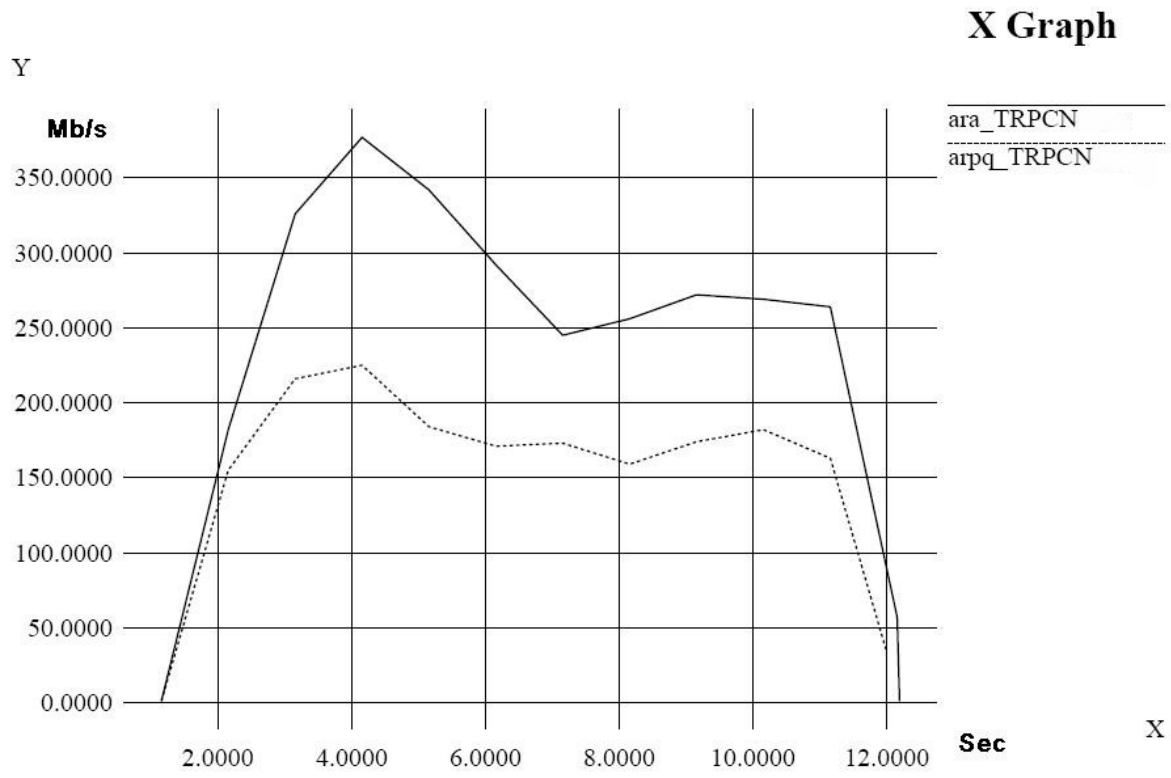



**Figure 32. Débits des paquets reçus à travers le réseau par les nœuds**

Sur cette courbe, on remarque que par rapport aux processus de découverte de route de ARA, notre protocole de routage QoS assure un gain important en terme de bande passante consommé durant la phase de découverte de route. Etant un protocole Multi agent QoS qui fait de la réservation de la bande passante, ARAQ consomme peu de ressources (bande passante) durant la recherche de route car les paquets agents RREQ et RREP ne traversent que les liens des nœuds qui ont suffisamment de bande passante pour satisfaire les requêtes QoS.

Pour finir nous nous sommes aussi intéressés au délai moyen de transmission des paquets dans le réseau afin de comparer le temps de latence du protocole ARAQ par rapport au protocole ARA dont il hérite ces fonctionnalités de base. Car notons-le même s'ils sollicitent de la bande passante, certaines données sont sensibles aussi au délai et la phase de découverte de route ne doit pas influer sur ce facteur.

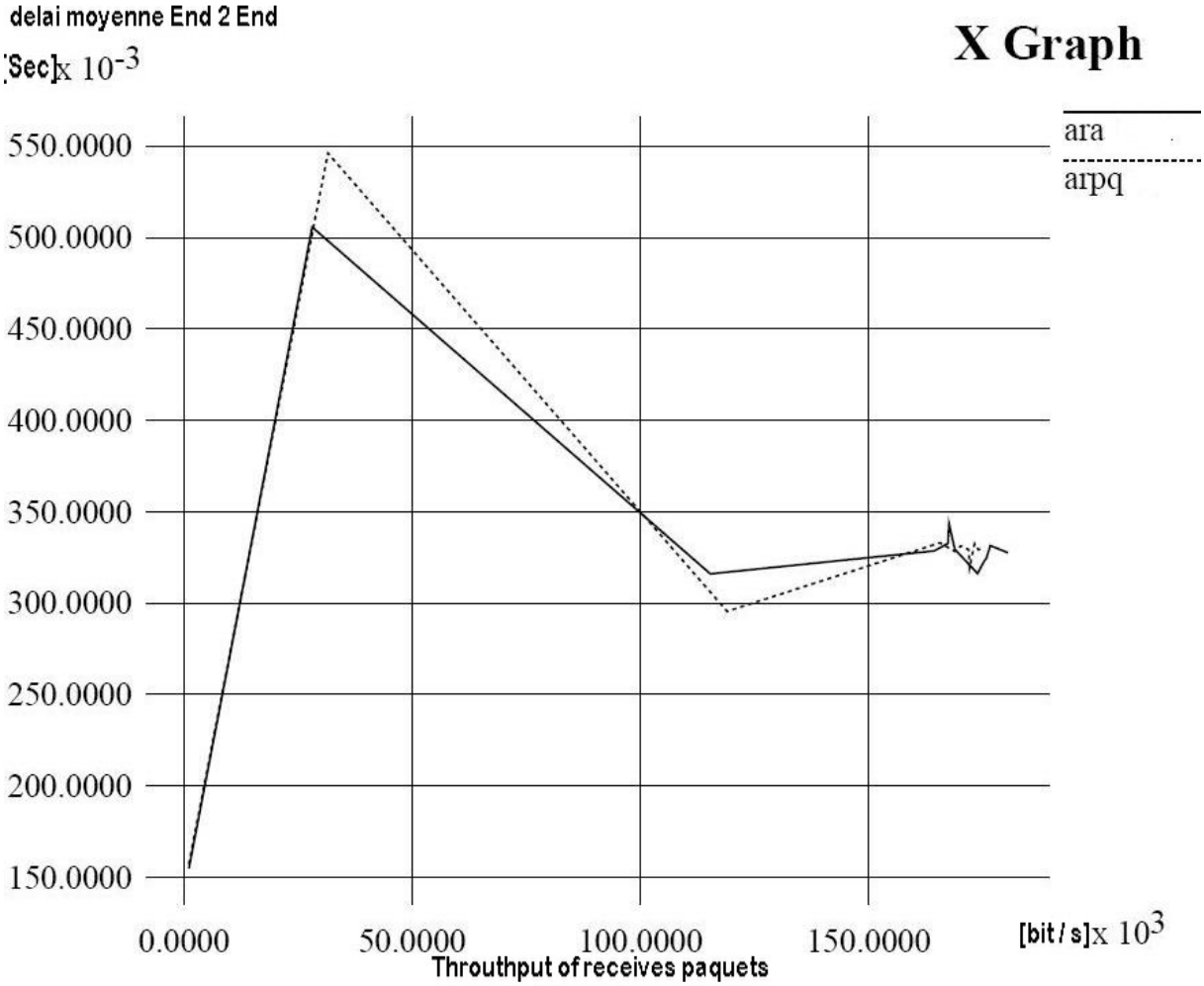



**Figure 33. Délai moyen de transmission des paquets**

Sur cette figure, on remarque au début de la phase de découverte de route, ARAQ à un délai moyen légèrement plus important qu'ARA. Ceci s'explique par le fait qu'une route optimale n'est pas toujours le plus court chemin. Cependant grâce aux SMA et au processus de marquage des routes par de la phéromone, on voit que le courbe va converger ver un délai moyen. Avec tous les chemins optimaux qu'ils ont empruntés tout au long de ce processus, les agents seront choisir grâce aux phéromones les routes les plus optimales en termes de ressources (bande passante) et aussi de délai d'acheminement. En conclusion, nous pourront dire que le protocole ARAQ n'influe pas trop sur le délai de transmission des paquets RREQ et RREP lors de la phase de découverte de routes.

Cette implémentation de notre protocole de Routage multi agent avec QoS, ARAQ dans NS-2, nous a permis de le valider dans un environnement de simulation dédié au réseau en général et ad hoc en particulier.

Avec ces différents paramètres étudiés pendant la simulation, nous avons montré quelques avantages et inconvénients de notre protocole. A savoir

- Le choix des chemins optimaux
- Support multi path
- Pertinences et cohérences des routes grâce au dépôt de phéromone
- Gain en bande passante

Notre exemple de simulation nous a permis de déceler certains manquements de notre proposition surtout pour la gestion des paquets de trafics de NS-2 dus à un problème de port communication du classifier (voir 4.3.1.B. a).

# *Chapitre 6: Mécanisme de sécurité pour le protocole de routage avec qualité de service basé sur les Systèmes Multi Agent : S-ARAQ (Secure ARAQ)*

Le routage avec qualité de service cherche à trouver des routes ayant suffisamment de ressources disponibles (bande passante) pour satisfaire une requête. Ce but fondamental du routage avec qualité de service montre que le bon fonctionnement de ce processus repose sur la disponibilité des ressources et aussi une bonne coopération des nœuds afin de fournir ces ressources et participer aux routages.

L'état de l'art sur la sécurité dans les réseaux ad hoc nous a permis de déceler quelques attaques qui peuvent être menées sur notre protocole de routage ARAQ. Ces attaques, pour la majeure partie de type DoS, visent à forcer un nœud à consommer ces ressources mais aussi incitent un nœud à avoir un comportement égoïste. Ces attaques sont d'autant plus faciles que la première vulnérabilité de ces réseaux est liée à la technologie sans fil sous-jacente. Quiconque possédant le récepteur adéquat peut potentiellement écouter ou perturber les messages échangés.

Dans ce qui suit, nous allons détailler ces différentes attaques qui peuvent être mené sur le protocole ARAQ, ensuite nous vous présenterons notre mécanisme de sécurité afin d'assurer la continuité du service de routage Qos de notre protocole.

## 6.1 Les attaques sur le protocole de routage Multi Agent ARAQ

Les attaques de type dénis de service apparaissent comme les attaques les plus faciles à réaliser par un attaquant. La criticité de telles attaques dépend fortement du contexte d'utilisation mais sont très sensibles aux services QoS notamment aux routages avec qualité de service.

### 6.1.1 Comportement égoïste des nœuds

Les mécanismes de routage sont d'autant plus critiques dans les réseaux ad hoc que chaque entité participe à l'acheminement des paquets à travers le réseau. Ainsi, l'égoïsme pousse un nœud à ne pas participer au bon fonctionnement du réseau dans le but de préserver ses ressources. Notons que notre protocole de routage avec Qualité de Service basé sur les SMA s'appuie sur la collaboration sans condition de ses éléments. Un nœud refusant de jouer son rôle peu mettre en péril le service de QoS.

Cette attaque peut se réaliser en mentant délibérément sur la disponibilité des ressources sur le nœud. A chaque fois qu'une requête de demande de route QoS arrive, le nœud fourni un résultat négatif du control d'admission pour éviter de réserver sa bande passante pour transmission des données vers la destination.

### 6.1.2 La consommation de ressources

Si un nœud malicieux a la capacité d'usurper l'identité d'un nœud valide du réseau, il peut initier un mécanisme de découverte de route QoS avec un faux paquet Route Request afin d'inciter un ou plusieurs nœuds à consommer leurs ressources en bande passante. Ainsi, chaque nœud qui reçoit ce faux paquet fera la réservation croyant qu'il vient d'un nœud valide. De ce fait, les nœuds visés n'auront pas assez de ressources disponibles pour satisfaire les requêtes valides de demande de route QoS des nœuds.

En retransmettant un paquet RREQ ou RREP, un nœud malicieux peut modifier les valeurs de la quantité de phéromone à déposer ou de la bande passante à réserver. L'effet immédiat de cette attaque est pour la première le partitionnement (l'isolement) d'une partie du réseau en diminuant la valeur de phéromone ou de forcer les paquets à passer par lui en augmentant la valeur de phéromone. En changeant la valeur de la bande passante du paquet RREQ ou RREP, le nœud malicieux va forcer ses nœuds voisins à supprimer les paquets car n'ayant pas assez de bande passante pour satisfaire cette fausse requête.

## 6.2 Mécanisme de sécurité pour l'intégrité et l'authentification des paquets RREQ et RREP de ARAQ

Afin de lutter contre ces attaques, deux solutions sont à envisager.

- Un mécanisme assurant l'authentification saut par saut et l'intégrité de paquets RREQ et RREP.
- Un système favorisant une bonne coopération des nœuds c'est-à-dire un système de surveillance du comportement des nœuds.

En ce qui concerne notre proposition, nous allons nous focaliser sur le mécanisme assurant une authentification et une intégrité des paquets agents RREQ et RREP car étant la base de toutes autres solutions de sécurité.

Le protocole S-ARAQ (Secure ARAQ) utilise les moyens classiques pour assurer l'intégrité et l'authentification des paquets échangés par les nœuds durant les phases de découverte de routes QoS et Best Effort sont l'utilisation des signatures numériques et des fonctions de hachage. Les signatures numériques s'appuient sur la cryptographie à clé publique et sera utilisées dans notre proposition pour assurer l'authentification des champs non modifiables des paquets RREQ et RREP de ARAQ à savoir les adresses source et destination, l'identificateur de paquet, l'identificateur de flux QoS, la valeur de phéromone et la valeur de la bande passante demandée. La fonction de hachage permettra d'assurer l'intégrité des champs mutables à savoir l'adresse de l'émetteur, l'adresse du prochain nœud et la liste des nœuds traversés tout au long de leur acheminement à travers le réseau pendant la phase de recherche de route.

### 6.2.1 Mécanisme pour l'intégrité

Pour l'intégrité des champs modifiables des paquets RREQ et RREP de ARAQ, nous allons utiliser la fonction de hachage [HACH]. Elle permet de réduire une donnée binaire de n'importe quelle taille en un condensé de taille fixe. Elle ne comporte aucun élément secret. On l'utilise très souvent pour la préparation des données en vue d'une signature numérique pour assurer l'authentification. Les algorithmes standards sont SHA (secure hash algorithme) [SHA] qui donne une empreinte de 20 octets, MD5 (message digest) [MD5] qui fournit quant à lui un condensé de 16 octets.

Une fonction de hachage doit avoir les propriétés suivantes:

- elle doit être simple à calculer à l'aide d'un algorithme efficace
- c'est une fonction à sens unique; on ne doit trouver de valeur qui produise un condensé donné
- on ne doit pas pouvoir trouver de collisions, c'est-à-dire deux valeurs qui ont le même condensé.

Pour assurer l'intégrité de ces champs, nous allons utiliser l'algorithme de hachage SHA-1. L'algorithme de hachage **Hash** prend en entrée un message M (contenant les champs modifiables : adresse de l'émetteur, adresse next hop et liste des nœuds) et fourni un condensé (haché) $H_S$

$$Hs = Hash(M) \qquad (7)$$

### 6.2.2 Mécanisme pour l'authentification

Pour assurer l'authentification des paquets RREQ et RREP de ARAQ, nous allons nous appuyer sur les techniques de signatures numériques [SIG].

L'objectif d'une signature numérique apposé à un message a pour objectif de permettre au destinataire d'authentifier l'origine de ce message et de lui prouver son intégrité. Leur implémentation fait appel aux fonctions de hachage et aux clés et asymétriques.

Une signature numérique sera appliquée sur les champs non modifiables (adresse source et de destination, bande passante demandée, valeur de phéromone, la fonction de hachage utilisée pour le calcul du condensé…). Cette signature numérique aura pour but d'authentifier l'origine des paquets et permettra ainsi de contrer les attaques de types usurpation d'identité et aussi les attaques de types déni de service vu dans la partie 5.2.2 visant à falsifier les paquets RREQ et RREP.

La signature numérique de ces champs sera basée sur un schéma de signature de type hache-et-signe basé sur RSA [RSA]. On dit qu'un schéma de signature est de type hache-et-signe si le message est d'abord haché en utilisant une fonction de hachage, dont le résultat est ensuite signé en utilisant un schéma de signature comme RSA. La combinaison de SHA-1 et de RSA fournit un schéma

efficace de signature numérique. Notre schéma de signature sera défini de la façon suivante:

Supposons que le nœud S avec comme clé publique $\mathbf{KU_S}$ et privée $\mathbf{KR_S}$ veut envoyer un paquet (RREQ ou RREP) au nœud D,

Au niveau du nœud source:

- L'algorithme de hachage **Hash** prend en entrée un message M (contenant les champs non modifiables) et fourni un condensé (haché) $H_S$

$$Hs = Hash(M) \qquad (6)'$$

- L'algorithme de signature **Sign** prend en entrée le condensé $H_S$ et sa clé privée $\mathbf{KR_S}$ et retourne la signature

$$Si_S = Sign_{KRs}(H_S) \qquad (7)$$

Au niveau du nœud de destination

- L'algorithme de vérification **Verify** prend en entrée $H_S$, la signature $Si_S$ et la clé publique de S $\mathbf{KU_S}$. Il renvoie un bit noté $Verify_{KUS}(H_S, Si_S)$, égal à 1 si la signature est acceptée, et 0 sinon.
- On requiert que si

$$Si_S \leftarrow Sign_{KRs}(H_S), \text{ alors } Verify_{KUS}(H_S, Si_S) = 1. \qquad (8)$$

Le condensé calculé, la fonction de hachage ainsi que la signature sont ajoutés aux paquets RREQ et RREP et envoyé à travers le réseau durant la phase de découverte de route. Chaque nœud qui reçoit un paquet l'authentifie et vérifie l'intégrité de ces champs avant de faire la réservation de route Best Effort ou QoS, sinon le paquet est simplement supprimé.

| Type | Id_flux (ToS) | size | TTL | F. hachage | Adresse IP émetteur | Adresse IP next hop |
|---|---|---|---|---|---|---|
| RREQ ID | | | | | | |
| Adresse IP de Destination | | | | | | |
| Adresse IP de la Source | | | | | | |
| Numéro de Séquence | | | | | | |
| bande passante demandée | | | | | | |
| Δφ (taux de phéromone) | | | | | | |
| Liste des nœuds | | | | | | |
| hash | | | | | | |
| Signature numérique | | | | | | |

**Figure 34. Format d'un paquet RREQ dans S-ARAQ**

## 6.3 Description du mécanisme de sécurité S-ARAQ

Le mécanisme S-ARAQ (Secure ARAQ) tente d'offrir une intégrité et une authentification aux paquets agents RREQ et RREP en se basant sur les algorithmes SHA-1 et RSA jugés très optimales pour le calcul des empreintes et des signatures numériques. Ce mécanisme permettra de lutter contre les attaques de types usurpation d'identité et consommation de ressources (falsification des paquets RREQ et RREP qos). Afin de comprendre le déroulement de notre mécanisme, voici une explication détaillée des différentes étapes.

Dans notre mécanisme de sécurité S-ARAQ, le nœud source, à l'initialisation de la phase de découverte de route,

- Calcule l'empreinte des données modifiables du paquet agent RREQ, c'est-à-dire les champs adresse de l'émetteur, adresse next hop et liste des nœuds, avec l'algorithme de hachage SHA-1.

- Calcule l'empreinte des champs non modifiables à savoir les adresses source et de destination, la bande passante demandée, la valeur de phéromone, la fonction de hachage du calcul de l'empreinte et le signe avec l'algorithme RSA.

- Ajoute la signature et le condensé au paquet RREQ qui est ensuite envoyé en diffusion dan le réseau.

Au niveau d'un nœud intermédiaire,

- On vérifie en premier la signature du paquet agent RREQ. S'il est valide, on vérifie l'intégrité des champs modifiables en recalculant le haché grâce à la fonction de hachage afin de le comparer à celui compris dans le paquet RREQ.

- Si ces conditions sont versifiées, le nœud fait le même traitement du paquet RREQ de ARAQ c'est-à-dire vérifie l'admissibilité de la requête avant de retransmettre le paquet et ajoute l'entrée dans la table de routage, sinon le paquet est supprimé.

- Avant de retransmettre le paquet vers son voisin, le nœud calcule le nouveau haché correspondant à l'adresse du prochain nœud qui a changé.

Arrivée à la destination, le nœud fait la même vérification qu'un nœud intermédiaire. Si l'authentification et l'intégrité des champs sont vérifiées, le nœud de destination formate le paquet RREP en prenant le soin de générer le haché et la signature. En suite le paquet est envoyé à la source.

Le fonctionnement de notre mécanisme de sécurité pour ARAQ peut être résumé sous forme d'algorithme que voici.

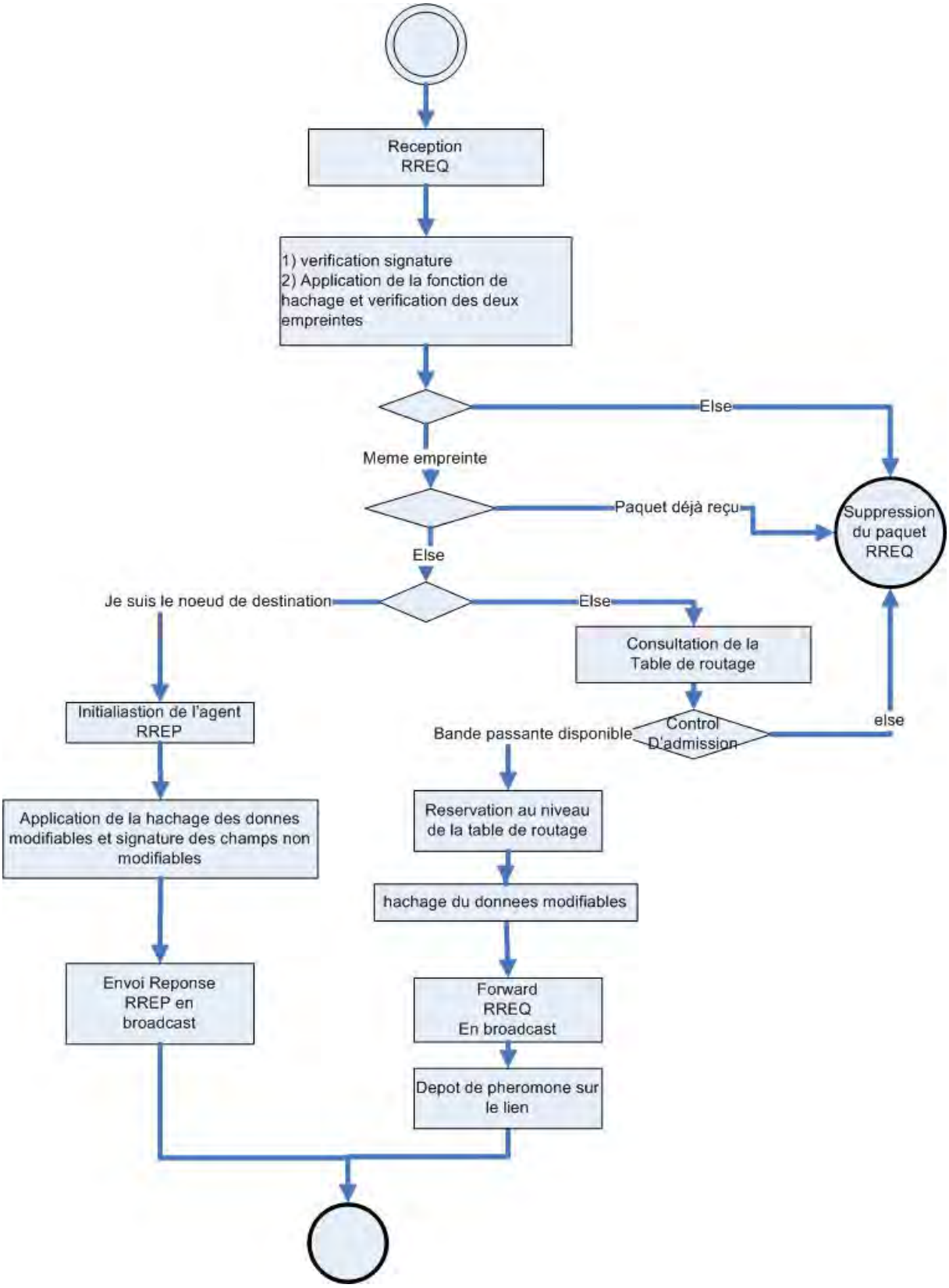



**Figure 35. Fonctionnement d'un nœud pour assurer l'intégrité et l'authentification des paquets RREQ et RREP**

La mise en place de ce mécanisme permettra de parer aux attaques de type usurpation d'identité, émission de faux paquets agent RREQ, etc.

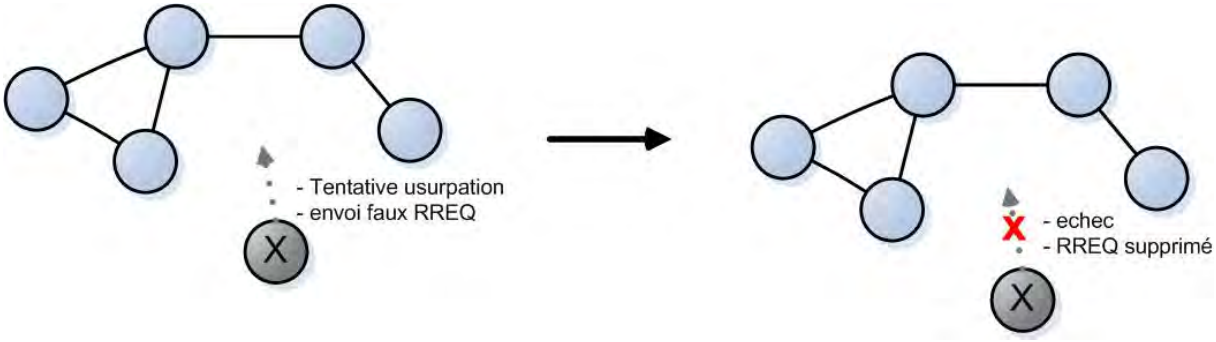



**Figure 36. Tentative d'attaque d'un nœud extérieur**

Pour les algorithmes cryptographiques utilisés, les cryptographes ont démontré que ça pourrait être possible pour le SHA-1 algorithme de hachage d'être rompu. Certains ont présenté une collision avec 58 tour-SHA-1, retrouvée avec 233 opérations de hachage. A la force brutale de recherche, il faudrait $2^{80}$ opérations. Toutefois, les experts font valoir que cela pourrait ne pas se produire pendant un certain temps car l'attaque avec ces $2^{80}$ opérations est à la limite de ce qui est réalisable et pouvait probablement être abordée via un calcul distribué à l'échelle planétaire. Néanmoins, l'Institut national des normes et de la technologie (NIST) a déjà des normes de plus en plus difficiles à briser. Il s'agit des fonctions de hachage: SHA-224, SHA-256, SHA-384 et SHA-512.

Pour RSA, jusqu'à 2005, le plus grand nombre factorisé, selon l'état de l'art en matière de calculs distribués, était long de 663 bits. Les clés RSA sont habituellement de longueur comprise entre 1024 et 2048 bits. Quelques experts croient possible que des clés de 1024 bits seront cassées dans un proche avenir (quoique ce soit controversé). On présume donc que RSA est sûr si la taille de la clé est suffisamment grande. On peut trouver la factorisation d'une clé de taille inférieure à 256 bits en quelques heures sur un ordinateur individuel, en utilisant des logiciels librement disponibles. Pour une taille allant jusqu'à 512 bits, et depuis 1999, il faut faire travailler conjointement plusieurs centaines d'ordinateurs. Par sûreté, il est couramment recommandé que la taille des clés RSA soit au moins de 2048 bits.

En s'appuyant sur les travaux de SAODV, nous avons pu proposer un mécanisme de sécurité à notre protocole de routage multi agent avec Qos ARAQ afin d'assurer la disponibilité des ressources. Ce protocole sécurisé **S-ARAQ** assure l'authentification des paquets et de la source pour lutter contre l'attaque Usurpation d'identité et l'intégrité des paquets luttant ainsi à l'émission de faux RREQ et RREP.

Cependant, ce protocole peut être victime d'une attaque DoS de la part d'un nœud malveillant qui envoie beaucoup de faux paquets RREQ ou RREP dans le but d'inonder le nœud. Le nœud victime passera tout son temps à vérifier ces faux

paquets et ne sera plus en mesure de fournir un service QoS pour le fonctionnement du protocole ARAQ.

L'autre inconvénient du système d'authentification proposé est la non gestion des nœuds corrompus. En effet un nœud authentifié est un nœud de confiance et peut avoir un mauvais comportement entraînant un disfonctionnement du processus de routage avec QoS.

Pour résoudre ces inconvénients un système de surveillance du comportement des nœuds ou de gestion de la réputation peut être ajouté au mécanisme de sécurité.

# Conclusion et Perspectives

La facilité de déploiement et l'absence d'une administration centralisée des réseaux ad hoc en font une solution présentant de nombreux avantages aussi bien dans les domaines civils que militaires. Ces dernières années, de nombreuses recherches ont été menées sur ces types de réseaux et plus récemment, sur la problématique de la Qualité de Service dans ces réseaux pour le routage des données multimédias telles que la vidéo, l'audio, etc. Cependant, comme nous l'avons vu dans tout ce document, les propriétés propres aux réseaux ad hoc rendent difficile la garantie des ressources comme la bande passante.

Le mécanisme de routage avec QoS basé sur les SMA (ARAQ) proposé et implémenté sous le simulateur NS2 permet de rechercher des routes admissibles qui garantissent la transmission des flux QoS avec la bande passante désirée. En effet, grâce à leurs propriétés Multi Agent modélisés à partir du comportement des colonies de fournis [DOR], les paquets RREQ et RREP arrivent à alimenter au niveau des tables de routage des nœuds les demandes de réservations de bande passante lors de la phase de découverte de route afin de transmettre les flux QoS et Best effort de la source vers leur destination. Cet ensemble de routes optimales sont obtenus grâce au control d'admission effectué par chaque nœud du réseau afin d'éliminer les routes considérées comme « non conformes aux exigences ». Les résultats de simulation montrent que les routes trouvées sont bien les routes satisfaisant les exigences de Qualité de Service et de routes optimales. Mais aussi que les tables de routage détiennent bien les entrées des réservations qui ont été faites au niveau des nœuds par le control d'admission afin de satisfaire les requêtes des différents flux (QoS et best effort) lors de la phase de découverte de route. Notre simulation a montré aussi que notre approche apporte un gain important en terme de Bande passante pendant la découverte de route et que les différents calculs effectués par le control d'admission dans ARAQ pour l'estimation et la réservation de la bande passante n'influent pas trop sur le délai de transmission des paquets RREQ et RREP.

D'autre part les attaques de type déni de service, comme la consommation de bande passante par l'envoi de faux paquets RREQ de demande de route QoS, qui peuvent paralyser notre architecture de qualité de service nous ont poussé à proposer un mécanisme efficace de sécurité pour notre protocole ARAQ. Ce mécanisme S-ARAQ, basé sur l'approche du protocole SAODV, assure une authentification de la source et une intégrité des paquets RREQ et RREP et surtout du champ bande passante demandé (bp_demandé). L'algorithme ainsi proposé repose son fonctionnement sur le modèle du schéma de signature Hash – Sign et permet d'offrir une sécurité efficace et optimale à notre architecture de Qualité de Service.

Cependant l'implémentation de notre algorithme de sécurité pour ARAQ (SARAQ) n'a pu être réalisée du fait de l'inexistence de plateforme de simulation

réseau à l'image de NS2 intégrant à la fois les Systèmes Multi Agent, les algorithmes de cryptages et un analyseur de paquets (etheral, tcpdump). Ceci est l'un des causes de la jeunesse de ce nouvel axe de recherche qu'est l'apport de la QoS basés sur les SMA dans les réseaux ad hoc. Mais nous espérons que dans un avenir proche ces plateformes verront le jour.

Il est à souligner à propos de cette mémoire que deux points n'ont pu être gérés et nous espérons qu'ils feront l'objet de nouveaux axes de recherche. Il s'agit d'une part de la gestion proprement dite de la transmission des données et du processus de maintenance différenciée des entrées des tables de routage. En effet, les flux Best Effort peut occuper une bonne partie de la bande passante disponible, empêchant les flux QoS d'effectuer leurs transmissions avec les garanties requises. Pour palier à cette situation, il faudra, par exemple, réguler le débit des flux Best Effort afin d'augmenter le taux d'acceptation des flux QoS à venir ou existante.

D'autre part, il y'a le problème de gestion des clés pour le mécanisme de sécurité S-ARAQ. En effet dans un environnement ad hoc où on ne connait pas tous les participants, il faudra mettre en œuvre une architecture de gestion de clés robuste et optimale qui n'engendre pas de trafics couteux lors des échanges de clés entre les nœuds

Les travaux sur ce sujet de mémoire, nous a permis de cerner toute la problématique autour de la qualité de service et de la sécurité dans les réseaux ad hoc. Mais aussi, il nous a offert une meilleure compréhension de l'apport crucial des Systèmes Multi Agent dans la gestion du routage avec qualité de service.

# Références :

## Webographie

[CNR] http://www.urec.cnrs.fr/metrologie/routage.html
[IET] http://www.ietf.org
[MAN] http: //www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08
[AOD] http://moment.cs.ucsb.edu/AODV/aodv.html
[DSR] http://www.cs.cmu.edu/~dmaltz/dsr.html
[RSA] http://fr.wikipedia.org/wiki/Rivest_Shamir_Adleman
[SHA] http://fr.wikipedia.org/wiki/SHA-1
[MD5] http://fr.wikipedia.org/wiki/MD5
[HACH] http://fr.wikibooks.org/wiki/Les_fonctions_de_hachage_cryptographiques

## Bibliographie

[ABE] Abed Ellatif Samhat, Tijani Chahed, Fortaleza, "IntServ sur DiffServ: Etude du rapport micro-flux/agrégat", Février 2003

[ANJ] Farooq ANJUM, Petros Mouchtaris, "Security for Wireless ad hoc network", WILEY 2007

[BLA] M. A. Shurman, S. M. Yoo, and S. Park, "Black hole attack in wireless ad hoc networks", in ACM 42nd Southeast Conference (ACMSE'04), pp. 96-97, Avril 2004.

[BOU] François Bousquet, Christophe Le Page, Jean-Pierre Müller, "Modélisation et simulation Multi-Agent", Information Interaction Intelligence (Actes des 2ème assises nationales du GdR I3, Nancy, décembre 2002)

[BUC] S. buchegger, J. Y. Le boudec, "Performance analysi of the confidant protocol", proceeding ACM 3rd international symposium on mobile ad hoc network and computing, 2002

[BUT] Levente BUTTYAN et J.P. HUBAUX, "Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks", ICCA, Institut de technologie de Suisse, Janvier 2001

[BRE] Pablo BRENNER, "A technical tutorial on the IEEE 802.11 protocol", 1997

[CES] César A. Mantilla, J. L. Marzo, "A QoS Framework for Heterogeneous Wireless Networks using a Multiagent System", Published at Proceedings of European Wireless 2004 . Barcelona, Spain. February 2004

[CHA] Chaudet & Guérin Lassous, "Routage QoS et réseaux ad hoc : de l'état de lien à l'état de nœud", Rapport de recherche Inria, Rhône-Alpes, Janvier 2003

[CHAO] Hakima CHAOUCHI, Maryline LAURENT-MAKNAVICIUS, "La sécurité dans les réseaux sans fil et mobiles 3, technologies émergentes", LAVOISIER, 2007

[DAH] B. DAHILL, Levine B., E. Royer, C. Shields, "A Secure Routing Protocol for Ad Hoc Networks", proceedings of the 10th conference on net work protocols (ICNP), November 2002.

[DOR] M. Dorigo and G. Di Caro, "The Ant Colony Optimization meta-heuristic", New Ideas in Optimization, pages 11–32. McGraw Hill, London, UK, 1999.

[FAL] Kevin FALL, The ns manual, the VINT Project, MAI 2007

[GAY] Valérie GAYRAUD, "La sécurité dans les réseaux sans fil ad hoc", Actes du symposium SSTIC03, 2003

[GKA] Raghav BHASKAR, Group Key Agreement in Ad hoc Networks, RR4832, INRIA, Mai 2003

[GUI] Philippe GUILLOT, "Introduction à la cryptographie", Université paris 8

[GUN] Mesut GUNES, Udo SORGES, Imed BOUAZIZI, "ARA – the Ant-Colony based Routing Algorithm for MANET", University of Technology Aachen, Germany

[HEL] Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, IT-22, no. 6, pp. 644-654, 1976.

[HOR] Eric HORLAIT, Pascal ANELLI, NS-2 : Principes de conception et d'utilisation, laboratoire LIP6

[HU] Yi-Chun HU, Adrian PERRIG, David B. JOHNSON, "ARIADNE: A Secure On-Demand Routing Protocol for Ad Hoc Networks", September 2002

[LEB] Jerome LEBEGUE, Bernard JOUGA, Christophe BIDAN, "Etat de l'art sur la sécurité des réseaux ad hoc", avril 2005

[MAR] Sergio MARTI, T. J. GIULI, Kevin LAI, and Mary BAKER, "Mitigating routing misbehavior in mobile ad hoc network", In ADV Mobile, 2001

[MIC] P. MICHARDI, "Mécanisme de sécurité et de coopération entre les nœuds d'un réseau mobile ad hoc", actes du symposium SSTIC06, Institut EURECOM, 2006

[PAP] PAPADIMITRATOS P., HAAS Z.J., "Secure Routing For Mobile Ad hoc Networks", SCS Communication Networks and distributed Systems Modeling and Simulation Conference, 2002

[PER] E. Perkins, Pravin Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", University of Maryland 1994

[PER] Perrig A et Al., "Efficient Authentication and Signing of Multicast Streaming over Lossy Channels", Procceding IEEE Symposium, security and privacy, IEEE Press

[RFC1058] C. HEDRICK, "Protocol d'Information de Routage (RIP) ", Network Working Group, juin 1988.

[RFC2501] Scott Carson et Joseph Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations (RFC 2501) ", Naval Research Laboratory University of Maryland, January 1999

[RFC2386] E. Crawley, R. Nair, "A Framework for QoS-based Routing in the Internet", Network Working Group, Aout 1998.

[RFC3561] Perkins, et al., "Ad hoc On-Demand Distance Vector (AODV) Routing, RFC 3561", University of California, Santa Barbara S. Das, University of Cincinnati 2003

[RFC3626] T.Clausen, P. Jacquet, "Optimized Link State Routing Protocol (OLSR), RFC 3626", INRA, octobre 2003

[RFC4728] D. Jhonson Rice University, Y. Hu UIUC and D. Maltz , "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4 (RFC 4728)", Microsoft Research, February 2007

[RUS] Yih-Chun Hu, Adrian Perrig, and David B. Johnson, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols", In *Proceedings of the 2003 ACM Workshop on Wireless Security*, pages 30–40, San Diego, CA, USA, 2003. ACM Press.

[SAN] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, Elizabeth M. Belding-Royer. "A Secure Routing Protocol for Ad Hoc Networks," In Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP). November 2002

[SAR] "De l'apport d'une évaluation précise des ressources pour la qualité de Service des réseaux ad hoc basée sur IEEE 802.11", Thèse à l'INSA de Lyon, Juillet 2007

[SIG] Benoit DEPAIL, "La signature numérique", exposés de système réseaux, 2006

[SPA] Otto SPANIOL, Mesut GUNES, "Routing Algorithm for mobile Multi-Hop Ad Hoc Networks", International Workshop NGNT

[STA] Frank Stajano and Ross Anderson, "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks", AT&T Laboratories Cambridge and University of Cambridge Computer Laboratory, 15th September 1999

[TAY] Tayeb Lemlouma, "Le routage dans les réseaux mobiles ad hoc", Université des Sciences et de la Technologie Houari Boumèdiene, Alger Septembre 2000.

[UIT 91] "UTI-T X.800, Réseaux de communication de données : interconnexion de systèmes ouverts (OSI); sécurité et applications", Architecture de s écurité pour l'interconnexion en systèmes ouverts d'applications du CCITT, 1991

[WOR] Yih-Chun Hu, Adrian Perrig, and David B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks", In *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, San Francisco, CA, USA, April 2003.

[WU] Kui Wu And Janelle HARMS , "QoS Support in Mobile Ad Hoc Networks", Crossing Boundoudaries – the GSA journal of University of Alberta, November 2001

[ZAP] Manel Guerrero Zapata, "Secure Ad hoc On-Demand Distance Vector (SAODV) Routing", draft-guerrero-manet-saodv-00.txt, Mobile Ad Hoc Networking Working Group, INTERNET DRAFT, 12 August 2001

[ZHO 99] Lidong ZHOU, Zygmunt J. HAAS, "Securing Ad hoc Networks", IEEE network magazine, novembre /decembre 1999

[YI] Seung YI, Prasad NALDURG, R. KRAVETS, "A Security-Aware Ad hoc Routing protocol for wireles network", 6th world multi-conference on systemics, cybernetics and informatics, 2002

Appendices:

**Exponentiation modulaire**: En mathématiques, plus précisément en arithmétique modulaire, l'**exponentiation modulaire** est un type d'élévation à la puissance (exponentiation) exécutée modulo un entier. Elle est particulièrement utilisée en informatique, spécialement dans le domaine de la cryptologie.
Généralement, les problèmes d'exponentiation modulaire prennent forme avec une base donnée *b*, un exposant *e*, un entier *m*. On souhaite calculer *c* comme ceci :

$$c \equiv b^e \pmod{m}$$