Afrique SCIENCE *08(1) (2012) 1 - 12*
ISSN 1813-548X, http://www.afriquescience.info

Auto adaptation incluant une double mobilité logicielle et physique : Analyse par simulation dans le cadre des réseaux ad hoc

Michel BABRI[1*], Boubakar BARRY[2], Souleymane OUMTANAGA[1] et Gérard PADIOU[3]

[1]Institut National Polytechnique Félix Houphouët-Boigny, BP 1093, Yamoussoukro, Côte d'Ivoire
[2]Université Cheikh Anta Diop, Dakar, Sénégal
[3]École Nationale Supérieure d'Électrotechnique, d'Électronique, d'Informatique et de Télécommunications, Institut de Recherche en Informatique de Toulouse (IRIT-CNRS), Toulouse, France

* Correspondance, courriel : *michelbabri@yahoo.fr*

Résumé

Dans ce papier, nous étudions les possibilités d'auto adaptation que pourrait offrir un calcul réparti incluant une double mobilité : d'une part une mobilité physique des sites supports d'exécution et, d'autre part, une mobilité des composants logiciels métiers ou clients.
Cette mobilité de niveau logiciel est envisagée dans le contexte de mobilité physique des réseaux Ad hoc. Comme illustration, nous décrivons un exemple de service de localisation de services, ceux-ci étant vus comme mobiles. Ce service s'appuie sur la propagation de rumeurs et l'usage de promenades aléatoires par des agents légers dédiés. L'évaluation des performances d'un tel service est faite grâce un simulateur de réseaux Ad hoc.

Mots-clés : *réseaux ad hoc, agents mobiles, protocoles de routage, rumeurs, simulation, Madhoc.*

Abstract

Self-adaptability for logical and physical mobility: simulation-based analysis in mobile ad hoc networks

In this paper we study the self-adaptability of a distributed application in a mobile environment. In such an environment, nodes and code are both mobile.
The logical mobility of code is used in physical mobility situation in mobile ad hoc networks. As an example, we describe a mobile services localization application. This localization is based on hints propagation and random walks. The proposal is evaluated throw simulation.

Keywords : *Manet, mobile agent, routing protocols, hints, simulation, Madhoc.*

1. Introduction

La prise en compte des besoins d'adaptation dynamique lors du développement d'applications réparties à grande échelle, augmente la complexité de celles-ci. La maîtrise d'une telle complexité, passe par la mise à disposition d'une infrastructure distribuée flexible. Nous étudions les possibilités d'auto adaptation que peut offrir un calcul réparti incluant une double mobilité :

- une mobilité physique des sites supports d'exécution,
- une mobilité des composants logiciels métiers ou clients. Ces derniers peuvent prendre différentes formes : agents mobiles, composants mobiles, codes mobiles, services mobiles. Le terme mobile indique la possibilité de déplacement de l'entité d'un site à un autre au cours de l'exécution globale du système.

En considérant par exemple, des composants métiers, on peut envisager qu'un composant ait besoin des services offerts par un autre composant. Dans un contexte mobile, le composant demandeur peut chercher à trouver le composant dont il a besoin en se déplaçant vers un site de rencontre. Une fois sur le même site, les deux composants, l'un momentanément client et l'autre momentanément serveur, peuvent coopérer le plus efficacement possible. Lorsque ses requêtes sont satisfaites, le composant client peut envisager de recommencer ce même schéma comportemental avec d'autres composants. Dans un contexte fortement mobile, l'un des principaux problèmes de base est, pour un agent donné, de localiser un partenaire et trouver le chemin à prendre pour atteindre celui-ci. Dans ce présent travail, nous étudions la réalisation d'un service de localisation pour faciliter la recherche d'un service (composant métier) par un client (composant applicatif).

La mise en œuvre de cette reconfiguration dynamique et/ou auto adaptation d'un calcul en utilisant deux niveaux de mobilité, peut prendre différentes formes décrites précédemment. De nombreux critères peuvent entrer en ligne de compte pour décider de façon pro active ou réactive le déplacement d'une entité : la recherche d'un service comme dans l'exemple précédent, mais aussi, la régulation de charge, la tolérance aux fautes, etc.

Nous envisageons l'utilisation de cette mobilité de niveau logiciel dans le contexte de mobilité physique des réseaux ad hoc. Ceux-ci sont en effet fortement dynamiques et constituent donc une base intéressante pour tester les effets ou apports d'une technique d'auto adaptation utilisant la mobilité logicielle.

Après un bref rappel des réseaux Ad hoc, nous décrivons un exemple de service de localisation de services mobiles. Ce service s'appuie sur la propagation de rumeurs et l'usage de promenades aléatoires par des agents légers dédiés. Pour évaluer les performances d'un tel service, nous utilisons le simulateur de réseaux Ad hoc nommé Madhoc [1].

2. Méthodologie

Les réseaux Ad Hoc

Les réseaux sans fil Ad hoc [2] ou encore MANET (Mobile Ad hoc NETwork) se définissent comme des réseaux dont la topologie ne s'appuie sur aucune structure préétablie. Les caractéristiques de tels réseaux peuvent être mises à profit par les participants à une réunion, par les opérations de secours, dans les échanges d'informations. Les caractéristiques principales de ces réseaux sont : l'utilisation des liaisons sans fil, l'autonomie des sites, la mobilité des sites, l'équivalence des sites.

Les considérations précédentes montrent toutes les difficultés rencontrées lors de la conception d'applications réparties à base d'agents mobiles pour réseaux Ad hoc. Pour résoudre ces difficultés, l'on peut procéder soit par des tests grandeur nature ou soit par simulation [3]. Les tests en environnement réel sont très onéreux car ils ont besoin d'infrastructures opérationnelles et de systèmes réels. La simulation, par contre a besoin d'une modélisation des applications, protocoles et ne requiert pas de contraintes matérielles la plupart du temps. La simulation est en définitive, une bonne alternative pour l'étude et l'évaluation des systèmes, protocoles, et applications.

Nous présenterons les résultats de simulation d'une application à base d'agents sur un réseau ad hoc. Cette simulation est réalisée grâce au simulateur de réseaux mobiles Ad hoc appelé Madhoc [4].

3. Spécification du service de localisation

3-1. Description du service de localisation

Nous considérons un réseau ad hoc composé de sites (ou de stations) mobiles, et une application formée de différents agents ou composants. Chaque agent est capable d'accomplir une tâche donnée via une interface exportée. De plus, il est possible que plusieurs agents aient la même expertise. On parle alors d'agents de même type. Le problème à résoudre est de permettre à un agent client de se déplacer vers le site d'un agent particulier ou appartenant à un type, et capable de lui rendre le service requis. Il s'agit donc essentiellement, pour un agent donné, de trouver la localisation du partenaire et le chemin à prendre pour atteindre celui-ci.
L'architecture de notre application est bâtie au tour de deux sortes d'agents: les agents dits commères et les agents dits applicatifs. Les commères jouent un rôle de coordination et de routage. En effet, il revient aux commères d'indiquer la route à prendre pour atteindre un agent applicatif donné, c'est-à-dire trouver et utiliser un service spécifique.
Quant à l'agent applicatif, il est caractérisé par les fonctions qu'il sait exécuter et par sa localisation courante dans le système.

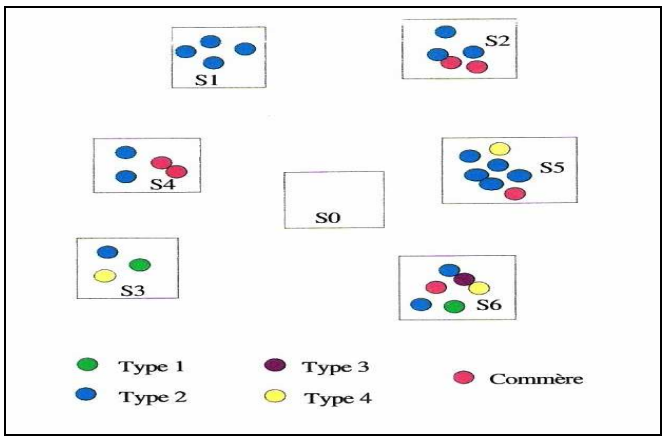

Figure 1 : *Exemple d'application*

La ***Figure 1*** illustre une application pour laquelle le réseau comporte 7 stations mobiles ou sites mobiles ($S_i, i = 0, 1, \ldots, 6$). On y distingue également 4 types d'agents applicatifs. Enfin, on note qu'au moins un agent de chaque type est présent sur le site S_6 (site complet) et qu'aucun agent n'est encore arrivé sur le site S_0 (site vide).
Il est à noter que dans un tel service, les sites sont des entités parfaitement anonymes en ce sens qu'un agent applicatif ne cherche jamais à atteindre un site particulier, mais bien un autre agent. C'est donc les agents ou les types d'agents qui sont nommés explicitement et non pas les sites.

3-2. Localisation d'un agent

Pour résoudre le problème de localisation d'un agent et l'acheminement de ce dernier vers son partenaire, notre approche est fondée sur les principaux points suivants :

- partitionnement des informations de localisation;
- utilisation de la notion de rumeur;

- utilisation de promenades aléatoires (random walks) pour la propagation des rumeurs.

Le service de localisation est conçu à partir d'un répertoire partitionné contenant la liste des agents. Chaque site contient un répertoire local qui enregistre à tout moment les agents présents sur ce site. Toutes les mises à jour de ces répertoires locaux n'impliquent aucune communication.

3-2-1. Description du comportement d'un agent « poursuivant »

Quand un agent arrive sur un site, son identité est enregistrée dans le répertoire local et chaque agent consulte ce répertoire pour savoir si l'agent cible qu'il recherche est sur ce site. Si l'agent cible n'est pas trouvé, l'agent « *poursuivant* » appelle une primitive *toward* pour obtenir en retour une destination de migration. Cette primitive renvoie soit une destination effective et l'agent tente de migrer vers celle-ci, soit aucune destination et, dans ce cas, l'agent peut par exemple décider de migrer au hasard vers un site voisin. Ce comportement peut être traduit par le code suivant :

```
/* consultation locale */
 Agent cible = lookup(nom|_cible);

 if (cible == null) {
   /* l'agent cible n'est pas là */
  Node dest = toward(cible);
   if (dest != null) (move(dest);
     else  move();          /* migration au hasard */
 }
```

Nous raffinons ce comportement générique selon deux stratégies :

- une stratégie bloquante : la primitive *toward* retourne toujours un voisin accessible: dans ce cas, cette primitive peut retarder l'agent appelant jusqu'à ce qu'un tel voisin existe;
- une stratégie non bloquante : la primitive renvoie immédiatement un voisin accessible ou une référence nulle s'il n'y en a aucun.

3-2-2. Interface applicative

Le service exporte deux primitives :

- *Agent lookup (String cible)* : cette primitive permet de vérifier si un agent cible est actuellement présent sur le site de l'agent appelant. Si l'agent cible n'est pas présent, la primitive renvoie une référence nulle. Il est alors nécessaire d'appeler la primitive suivante;
- *Node toward (String cible)* : cette primitive permet de trouver vers quel site l'agent appelant doit migrer pour trouver l'agent cible. Cette primitive assure une fonction de routage vers le nœud cible. Selon l'implantation du service, cette primitive peut soit renvoyer un site voisin accessible après éventuellement un certain délai (stratégie bloquante) soit renvoyer une référence nulle immédiatement (stratégie non bloquante).

3-2-3. Relations de localisation

La localisation des agents peut être traduite par deux sortes de relations entre agents répartis sur le réseau :

- une relation de visite
- une relation de voisinage

3-2-3-1. Relation de visite

Une relation de visite est définie sur l'ensemble des sites pour chaque agent :

$$S \xrightarrow{A} S \qquad (1)$$

où S est le domaine des noms de sites et A est le domaine des noms d'agents. Une occurrence de cette relation, $S \xrightarrow{A} S'$ définit le site voisin $s' \in S'$ vers lequel un agent a de A s'est déplacé lorsqu'il a quitté le site $s \in S$ pour la dernière fois.
Une relation de visite peut être enregistrée à chaque fois qu'un agent quitte un site. Étant donnée la mobilité des sites, si le site s change de voisins, cette valeur peut indiquer un site qui n'est plus voisin de s. Cette information est alors obsolète et peut être oubliée. Cette notion, qualifiée de *pointeur de poursuite*, est utilisée par les protocoles de routage de messages inter-agents [5].

3-2-3-2. Une relation de voisinage

Une relation de voisinage est définie sur l'ensemble des sites pour des ensembles d'agents. Elle consiste à mémoriser les usagers qui sont momentanément sur les sites voisins. Ce type d'information est assimilable à une propagation de rumeurs. Cette relation est notée :

$$S \xleftarrow{\lfloor A \rfloor} S \qquad (2)$$

où $\lfloor A \rfloor$ est l'ensemble des ensembles d'agents engendré à partir de A.
Pour un site s, la relation $s \xleftarrow{a_1,a_2,\ldots} s'$ signifie que $\{a_1, a_2, \ldots\}$ constitue un ensemble d'agents qui sont supposés être présents sur le site voisin s' par le site s. L'agent qui a amené cette information est le dernier à avoir vu les a_i sur le site s', cet agent ayant ensuite migré vers s. Comme précédemment, cette information peut être erronée si certains a_i ont bougé ou si le site s' cesse d'être voisin de s.
La relation de visite est liée à celle de voisinage de la façon suivante: pour tout agent a, sa dernière migration du site s vers le site s' apporte au site s la connaissance suivante: le site s sait que l'agent a est présent sur le site s', c'est-à-dire, en terme de relations :

$$\left(s \xrightarrow{\ldots,a,\ldots} s'\right) \Rightarrow s \xleftarrow{\ldots,a,\ldots} s' \qquad (3)$$

Nous nous intéressons à la propagation de la relation de voisinage en tant que rumeur parmi les sites pour obtenir des chemins entre deux agents quelle que soit leur localisation. Par exemple, si les relations suivantes sont connues :

$$s_1\{t\} \xleftarrow{a,b,c,d} s_2\{a,a\} \wedge s_2\{a,b\} \xleftarrow{c,d} s_3\{c,d\} \qquad (4)$$

L'agent poursuivant t présent sur le site s_1 et cherchant l'agent c, peut être dirigé vers l'agent cible par le chemin $\{s_1;s_2;s_3\}$.
Pour réaliser cette propagation, nous adoptons une approche épidémique [6] pour gérer les rumeurs. L'évaluation de la relation de voisinage à l'aide de rumeurs est la principale caractéristique de notre approche.
En fait, les deux relations définissent des chemins pour atteindre un agent cible et peuvent être combinées. Par exemple, un agent poursuivant t sur un site s_1 peut se déplacer sur le chemin $p = \{s_1; s_2; s_3; s_4\}$ pour trouver l'agent cible a présent sur le site s_4 si les relations suivantes sont connues :

$$s_1\{t\} \xleftarrow{\ldots,a,\ldots} s_2 \wedge s_2 \xleftarrow{\ldots,a,\ldots} s_3 \xrightarrow{a} s_4\{a\} \qquad (5)$$

Chaque site ne peut obtenir qu'une connaissance partielle et approximative de ces relations. Dans une situation idéale, chaque site pourrait savoir quel site parmi ses voisins permet d'atteindre par le plus court chemin un agent donné.
Malheureusement, dans un environnement réparti, il est impossible à un site de connaître l'état global courant du système à cause de la mobilité des sites et des agents qui rend les rumeurs obsolètes au fur et à mesure du temps qui s'écoule.

4. Implantation du service et mise en œuvre de la simulation

4-1. Implantation du service

L'implantation du service a pour objectif de fournir une connaissance la plus exacte et globale possible des relations précédentes à chaque site.
Pour atteindre cet objectif, tout site gère une table dont chaque entrée contient un couple $hint(A) =< A,n >$ concernant l'agent A et fournissant le site voisin vers lequel migrer pour atteindre l'agent A.
Grâce à cette table, un agent poursuivant peut migrer vers l'agent cible final d'un pas de plus.
Cette table implante les deux relations. Par exemple :

- une relation de visite $S \xrightarrow{a} S'$ est enregistrée sur le site s par une entrée $< a, s' >$.
- une relation de voisinage $S \xleftarrow{a,b,c} S'$ est enregistrée par un ensemble d'entrées $\{< a,s' >, < b,s' >, < c,s' >\}$.

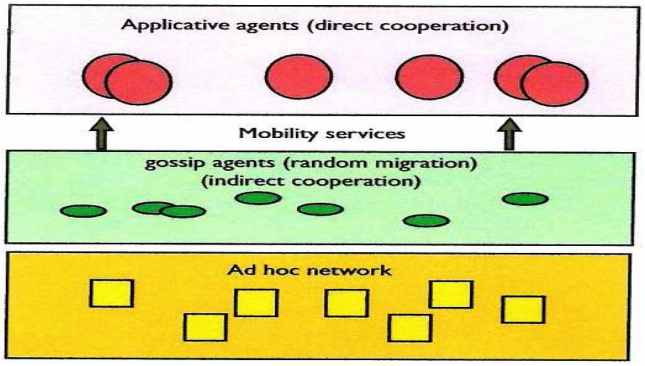

Figure 2 : *Architecture de propagation de rumeurs*

Dans un tel système dynamique, une entrée peut devenir rapidement obsolète. C'est pourquoi, un problème clé est la validité des entrées et leur stratégie de mise à jour critique.
Face à ce problème, nous adoptons une stratégie s'appuyant sur une architecture d'agents mobiles illustrée par la ***Figure 2***. Dans cette architecture, on distingue deux niveaux d'agents mobiles: un premier niveau d'agents mobiles "légers" qui propagent les rumeurs et un second niveau constitués des agents applicatifs que l'on peut qualifier d'agents "lourds". Ce sont ces agents applicatifs qui utilisent le service de localisation assuré par les agents légers.

En utilisant des agents légers mobiles migrant aléatoirement dans le réseau lui aussi mobile, nous pouvons propager les rumeurs à travers le réseau global. De telles promenades aléatoires ont été déjà proposées comme base dans divers algorithmes : protocoles de maintenance et/ou recherche de topologie dans les réseaux pair à pair non structurés [7], protocoles de communication de groupes dans les réseaux ad hoc [8] et protocoles de routage dans les réseaux sans fils [1].
Quant à la relation de voisinage, elle est associée aux agents applicatifs: quand un agent applicatif A quitte un site s pour aller sur le site s', il crée ou modifie une entrée de la table locale du site de départ. Celle-ci contient le couple $< A, s' >$. De telles entrées peuvent être interprétées comme des pointeurs de poursuite [9] et peuvent se substituer à des entrées obtenues par des rumeurs.

4-2. Mise en œuvre de la simulation

Notre application comporte deux sortes d'agents, les agents appelés « commères » qui propagent les rumeurs et les agents applicatifs. Un agent commère est un agent qui collecte des informations sur les agents applicatifs qu'il trouve sur les sites visités par lui.
Un agent applicatif est dédié à une tâche précise qu'il exécute au profit d'un utilisateur.
A un instant donné, le site de localisation d'un agent (commère ou applicatif) possède un ensemble de voisins qui forment ce que nous appelons son voisinage (éventuellement vide). Celui-ci est dynamique, en ce sens que de nouveaux sites peuvent apparaître, d'autres disparaître, et les liens entre sites peuvent également évoluer.
Nous utilisons un mécanisme de diffusion pour trouver le site de localisation de l'agent applicatif concerné. Afin d'éviter un écroulement très rapide du réseau, il convient d'assurer au mieux cette diffusion tout en réduisant la charge globale, en termes de messages. Nous avons vu que l'idée est d'utiliser des agents légers.
Le ou les agent(s) commères parcourent le réseau (de façon aléatoire) et mettent à jour les informations concernant la localisation des différents agents. A cet effet, l'agent commère gère une liste des agents qu'il a vus sur les sites déjà visités.
Il s'agit à l'étape actuelle, de décrire les différents protocoles, en termes de formats des messages et des algorithmes utilisés pour simuler les agents légers et applicatifs avec le simulateur Madhoc.

4-2-1. Description de l'environnement de simulation: Madhoc

Madhoc est un simulateur conçu pour des réseaux ad hoc mobiles hétérogènes à large échelle. Les stations sont différentes aussi bien du point de vue de leur nature que de leur technologie de communication. Il s'agit d'ordinateurs portables, de téléphones mobiles etc.
Madhoc supporte actuellement les protocoles tels que le WiFi (IEEE802.11b), Bluetooth et le sans fil USB. Toutefois, ces protocoles ne sont pas spécifiés dans le détail. Ils sont décrits en termes de bande passante totalement partagée par tous ; de distance de couverture, de taille des paquets, de coût de transfert des données.
Une application Madhoc est définie comme un ensemble de programmes s'exécutant sur des stations mobiles et qui communiquent dans un environnement soumis à de fréquentes variations.

4-2-1-1. Principe de fonctionnement du simulateur

Le simulateur propose un protocole de communication de messages. Le protocole de communication proposé dans Madhoc est bien entendu un protocole de diffusion vers les voisins accessibles. Une première étape de modélisation est donc de définir comment seront modélisés, simulés les agents et leur mobilité. Le

simulateur permet de programmer des applications répliquées sur les sites mobiles et qui peuvent donc s'échanger des messages lorsque leurs sites supports sont voisins.
A chaque pas de simulation, le simulateur exécute, sous forme atomique, une étape de chaque application active où qu'elle soit. Cette étape est décrite quelle que soit l'application simulée dans une méthode de nom prédéfini *void doIt (double time)*. Une telle étape présente toujours la même structure correspondant à un pas de calcul réparti diffusant :

- consommer les messages envoyés à l'application et en attente de réception;
- exécuter des traitements locaux selon les messages reçus;
- envoyer des messages selon le résultat des traitements locaux.

Dans ces conditions, la simulation des agents se traduit donc sous la forme d'une application mixant les deux types d'agents. Les objets agents des deux types, commères et applicatifs, sont représentés sous la forme de tableaux statiques accessibles par toute occurrence d'application où qu'elle soit.
Les agents sont créés lors de la création des objets applications.
Les objets agents sont créés au démarrage de la simulation. Les mouvements des agents vont se traduire par l'échange de messages.
Tous les messages simulant la migration des agents sont identifiables par un numéro. Un agent applicatif pourra chercher à trouver un agent cible en désignant ce dernier par son numéro. De plus, les rumeurs sur la localisation des agents utiliseront ces numéros.
Dans chaque objet application vont être gérées les structures de données suivantes :

- la table de routage des agents applicatifs mise à jour à partir des rumeurs et des départs d'agents quittant le site;
- la liste des agents applicatifs présents sur le site;
- la liste des agents commères présents sur le site.

4-2-1-2. Message simulant la mobilité des agents commères

Un agent commère collecte des informations à partir des sites visités. Ces informations sont contenues dans une liste. Dans la simulation, c'est cette liste qui va circuler de site en site via des messages qui simuleront la mobilité des agents commères. Chaque élément de cette liste constitue une rumeur sous la forme d'un numéro d'agent applicatif et d'un compteur de propagation. Ce compteur de propagation indique la longueur du chemin déjà parcouru par la rumeur.

Type_Agent (ici commère)	Num_commère	Site_destination	Num_appli, Cpt

Figure 3 : *Format du message d'un agent commère*

Comme nous l'avons dit, lorsqu'un tel message transportant des rumeurs est émis lors de l'exécution d'un pas de simulation, celui-ci est diffusé à tous les sites voisins. Pour éviter une explosion du nombre de messages, le message "commère" contient un numéro de site de destination qui précise le site sur lequel l'application pourra continuer la propagation des rumeurs. Tous les autres sites du voisinage ayant reçu le message se contenteront d'exploiter les rumeurs contenues dans le message sans le propager. On évite ainsi une inondation du réseau qui surviendrait si chaque voisin récepteur devait propager à son tour vers ses voisins.
Le nombre *cpt*, associé à un agent applicatif doit être incrémenté à chaque pas de propagation de la rumeur et rester inférieur à une limite LIMIT qui correspond à la longueur maximum autorisée du chemin parcouru par une rumeur. Dès que cette limite est dépassée, la rumeur est extraite de la liste.

4-2-1-3. Message applicatif

Un agent applicatif exécute indéfiniment un pas de calcul simple: ayant choisi un agent cible (par tirage aléatoire d'un numéro), il tente de migrer vers le site où se trouve cet agent. Au niveau du simulateur, ce pas de calcul est exécuté naturellement dans la méthode *doIt* pour chaque agent présent sur le site de l'application. Un message simulant la migration est envoyé vers la destination de l'agent. Sa réception permettra d'enregistrer le déplacement de l'agent applicatif. Ce message correspond à un agent applicatif du système, et comprend simplement le type de l'agent (donc le type applicatif) et son numéro.

Type_Agent (ici applicatif)	Num_Agent

Figure 4 : *Format du message d'un agent applicatif*

4-2-2. Algorithme d'un pas de simulation

Un pas de simulation se décompose en trois étapes :

- recevoir les messages en attente qui sont soit des listes de rumeurs soit des agents applicatifs;
- traiter les agents commères présents (arrivés) sur le site : il faut simplement essayer de les faire migrer vers un site voisin;
- traiter les agents applicatifs présents (arrivés) sur le site vérifier pour chaque agent applicatif si l'agent cible qu'il cherchait n'est pas présent localement : pour chaque rencontre possible, compter celle-ci et faire choisir un autre agent cible à chaque agent ayant réussi une rencontre; faire migrer le(s) agent(s) applicatif(s) (arrivés) sur le site;

Plus précisément, on a donc les étapes suivantes :

- Recevoir : Suivant le type de l'agent fournisseur ou applicatif reçu, la table de routage est mise à jour et les agents applicatifs arrivants sont enregistrés comme présents.

```
Recevoir(){
  S'il s'agit d'un message commère {
     Ajouter l'agent commère à la liste locale si le site est déclaré destination;
     Exploiter les rumeurs pour mettre à jour la table de routage;
  }
  sinon { // il s'agit d'un message applicatif
     Enregistrer l'agent sur le site local
  }
}
```

- Migration des agents applicatifs

Cette méthode assure la migration des agents applicatifs :

```
MigrerAppli (Collection<Station> voisinage) {
  Pour tout agent local
    Si le site de destination est dans le voisinage courant // il existe un lien
       Construire un message applicatif
       Extraire l'agent de la liste des présents;
       Envoyer le message applicatif vers un site voisin;
    }
Pour chacun des agents partis mettre à jour la table de routage;      }
```

- Migration des agents commères

Cette méthode se charge de faire migrer les agents commères présents sur le site.

```
Migrercommère (Collection<Station> voisinage){
  Pour tout agent commère {
      Construire un message commère;
      Choisir une nouvelle destination;
      Envoyer le message;
  }
```

4-3. Une optimisation du premier algorithme

La prise en compte d'une rumeur est liée à son âge. En effet, lors du passage d'un agent commère sur un site, les informations de localisation vont vieillir d'une unité si la direction donnée existe encore.

On a cherché à augmenter le nombre de rencontres des agents. Pour cela, on introduit la notion de type d'agents. Un agent cherche un type d'agent mais plusieurs agents du même type peuvent exister. Autrement dit, on passe à une localisation par pages jaunes plutôt que par pages blanches.

Le concept de type d'agents permet de restreindre la recherche. On peut s'attendre à trouver plus facilement un service en cherchant l'un quelconque des agents pouvant l'assurer plutôt qu'un agent particulier.

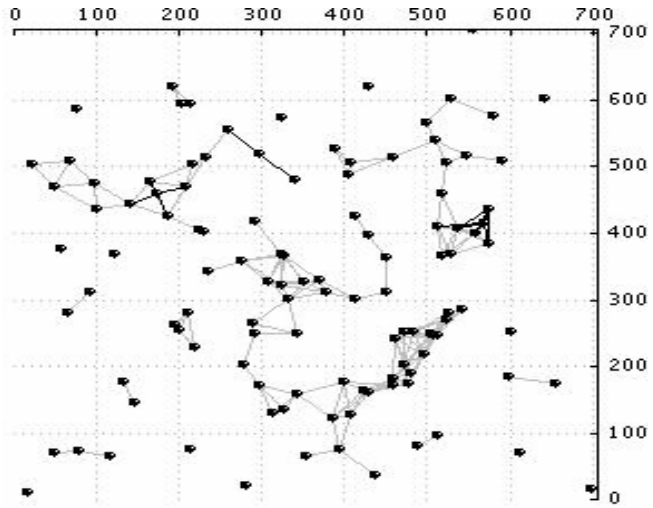

Figure 5 : *Une image du réseau*

La ***Figure 5*** présente une image du réseau. Les points représentent les différentes stations du réseau. Celles-ci peuvent être des téléphones mobiles, des ordinateurs portables et autres matériels. Un agent peut migrer de station en station à la recherche d'un service offert par un autre agent. Dans le même temps la station hôte peut se déplacer.

Les principaux paramètres de la simulation sont :

- LIMIT: qui désigne la longueur maximale d'un chemin de propagation d'une rumeur;
- PROP_F: la proportion d'agents commères par rapport au nombre total d'agents
- NB_TYP: le nombre de types d'agents applicatifs.

5. Analyse des résultats de différentes simulations

Nous avons effectué plusieurs mesures en faisant varier certains des paramètres comme indiqués plus haut. La ***Figure 6*** en donne la courbe.
Il en ressort un certain nombre d'observations :

- Propagation des rumeurs : nous avons vu que les rumeurs peuvent être propagées pendant un nombre plus ou moins grand de migrations d'un agent commère. Pour un chemin maximal de propagation trop grand (LIMIT), il y a trop de migrations pour peu de rencontres. Cela parce qu'au moment où un agent atteint un site intermédiaire, celui-ci peut ne plus avoir de connexion vers le site destination: le chemin peut avoir disparu à cause de la durée de propagation de la rumeur qui est devenu trop longue. Les valeurs 4 et 5 donnent les meilleurs résultats.
- Régulation du nombre d'agents commères : il faut également un bon compromis pour la proportion d'agents commères. En effet, avec peu de commères, la mise à jour des destinations est trop lente et induit un accroissement du nombre de migrations. A l'inverse, un nombre élevé de commères inonde le réseau avec les mêmes informations véhiculées par plusieurs d'entre eux.

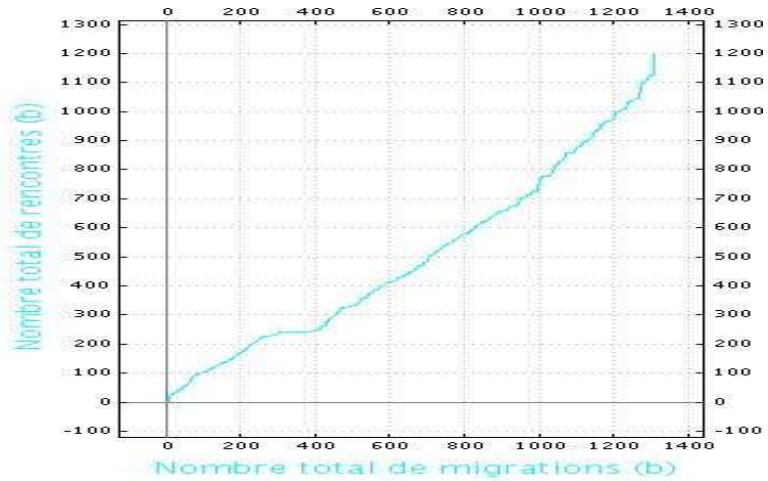

Figure 6 : *Résultat de la simulation*

L'introduction de la notion de type d'agents a permis d'améliorer de façon substantielle les différents résultats obtenus. En effet avec un nombre moindre de migrations, on obtient un plus grand nombre de rencontres. C'est-à-dire qu'un service recherché sera atteint beaucoup plus rapidement.
On observe un nombre de rencontres plus élevé que celui des migrations, comme résultat de la recherche de services locaux (par les agents localisés sur le même site).
Les résultats de la ***Figure 6*** ont été obtenus avec 206 agents applicatifs initialement répartis au hasard dans le réseau pour les valeurs suivantes : LIMIT= 4, PROP_F=30 et NB_TYP = 20.
Il y a donc, pour chaque type d'agent applicatif, environ une dizaine d'agents.

6. Conclusion

Dans cet article, nous avons proposé un service de localisation distribué utilisant la double mobilité dans les réseaux mobiles Ad hoc. Notre principale contribution a été de proposer un algorithme qui permette aux agents composant un calcul de s'auto organiser lors de la recherche d'un service. Des simulations ont été

effectuées pour démontrer l'efficacité de notre proposition. L'introduction de la notion de type d'agents a permis d'améliorer de façon substantielle les différents résultats obtenus. La prise en compte de délais s'est faite par la capacité offerte à un agent de migrer de façon aléatoire.

Références

[1] - M. BUI, S. K. DAS, A. K. DATTA, and D. T. NGUYEN, "*Randmized mobile agent based routing in wireless networks*". International Journal of Fondations of Computer Science, vol.12(3) (2001) 365-384.

[2] - C. E. PERKINS, "*Ad Hoc Networking*", Addison-wesley, 75 Arlington Street, Suite 300, MA 02116, Boston (2001).

[3] - L. HOGIE, P. BOUVRY, and F. GUINAND, "*An overview of MANETs simulations*" In L. Brim and I. Linden, editors, Electric Notes in Theretical Computer Science, in the proceedings *of MTCoord'05*, of LNCS, Namur, Belgium vol.150 (2006), 81-101 Elsivier.

[4] - L. HOGIE, P. BOUVRY, and F. GUINAND, "*The madhoc simulator*", http://www-lih.univ-lehavre.fr/~hogie/madhoc/.

[5] - J. H. AHN, "*Decentralized inter-agent message forwarding protocols for mobile agent systems*", In Computational Science and Its Applications ICCSA 2004, International Conference, vol. 3045, Springer-Verlag (2004) 376-385.

[6] - A. DEMERS et al., "*Epidemic algorithms for replicated database maintenance*", In 6th Symposium on Principles of Distributed Computing, (1987) 1-12.

[7] - C. GKANTSIDIS, M. MIHAIL, and A. SABERI, "*Random walk in peer-to-peer networks*", In INFOCOM. The Conference on computer Communications. IEEE (2004).

[8] - S. DOLEV, E. SCHILLER, and J. WELSH, "*Random walk for self-stabilizing group communication in ad hoc networks*", In Proceedings of the twenty-first annual symposium on Principles of distributed computing (PODC'02), New York (2002) ACM Press, 259-259.

[9] - J. DESBIENS, F. RENAUD, and M. LAVOIE, "*Communication and tracking infrastructure of a mobile agent system*", In The thirty-First Annual Hawaii International Conference on System Science, vol.7 (1998) 54-63.