

Déclaration et découverte de services dans les réseaux ad-hoc sans fil

Hend Koubaa, Eric Fleury

▶ **To cite this version:**

Hend Koubaa, Eric Fleury. Déclaration et découverte de services dans les réseaux ad-hoc sans fil. Colloque Francophone sur l'Ingénierie des Protocoles - CFIP 2000, Oct 2000, Toulouse, France, 16 p. inria-00107861

HAL Id: inria-00107861
https://inria.hal.science/inria-00107861

Submitted on 19 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Déclaration et découverte de services dans les réseaux ad-hoc sans fil

Hend KOUBAA[*] — Éric FLEURY[**]

[*] *LORIA , Université de Nancy I*
[**] *LORIA , INRIA*
Campus Scientifique - BP 239
54506 Vandœuvre-Lès-Nancy Cedex
Tél : 0383 593 090 - Fax : 0383 278 319
{Hend.Koubaa,Eric.Fleury}@loria.fr

RÉSUMÉ. *L'évolution des services dans le réseau est aujourd'hui incontestable. Il semble indubitable que le nombre de services offerts va aller en croissant mais il est aussi à parier que la demande de services va elle aussi suivre la même croissance. Face à cette évolution il apparaît nécessaire de pouvoir déclarer, découvrir et déployer des services de façon la plus transparente possible et surtout de façon dynamique. Les protocoles qui sont proposés à l'heure actuelle le sont pour des réseaux filaires. Il nous semble intéressant de se pencher sur le cas des réseaux ad-hoc car le domaine de l'informatique mobile est en plein essor. Un réseau ad-hoc est une collection d'entités mobiles, interconnectées par une technologie sans fil formant un réseau temporaire sans l'aide de toute administration ou de tout support fixe. Cet article présente une approche pour la déclaration et la découverte de services adaptée aux réseaux ad-hoc.*

ABSTRACT. *The widespread deployment of inexpensive communication technology, computational resources in networks raises an interesting problem to users : how to locate a specific network service. Moreover, in front of the explosion of available services, the declaration, deployment, and lookup must be as transparent as possible and the configuration mechanism must be dynamic. Most of discovery service protocols proposed are designed and proven to be effective for fixed wired local-area network. It seems interesting to study such protocols in an ad-hoc infrastructure. An ad-hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any central organization or standard support services. This paper presents our approach and the architecture of a discovery service protocol designed for ad-hoc networks.*

MOTS-CLÉS : *services, déploiement, découverte, réseaux ad-hoc, sans fil, mobilité, technologie active*

KEYWORDS: *services, deployment, discovery, ad-hoc network, wireless, active technology*

1. Introduction

Pour utiliser un service présent sur un réseau, un utilisateur doit, encore à l'heure actuelle, connaître l'adresse réseau du service en question. Par « *service* » nous désignons de façon très générique toute application possédant une interface clairement définie capable d'effectuer des traitements ou des actions au profit et au nom d'un utilisateur, souvent nommé « *client* » dans ce contexte.

Les évolutions actuelles des réseaux couvrent un large spectre et permettent d'aller d'une infrastructure offrant un ensemble de services minimum (par exemple des imprimantes) vers des architectures beaucoup plus sophistiquées et programmables permettant d'offrir un ensemble complexe de services (jeux de réalité virtuelle, encodage temps réel vidéo). Il semble indubitable que le nombre de services offerts va aller en croissant mais il est aussi à parier que la demande de services va elle aussi suivre la même croissance. Dans une telle perspective d'évolution, une configuration manuelle telle que nous la connaissons actuellement, d'un grand nombre de services diversifiés, va très vite devenir impraticable. Afin de répondre en partie à ce problème, les protocoles de découverte de services (SDS [CZE 99], SLP [VEI 97], JINI [MIC]) permettent à l'utilisateur de ne plus avoir à se soucier de l'adresse réseau du service désiré. La seule chose qui incombe à l'utilisateur est de donner le type et les attributs du service requis. Le rôle d'un protocole de découverte de services va alors être de se charger de résoudre et de trouver l'adresse de la machine sur laquelle se trouve le service désiré.

Cependant, tous les protocoles de services actuellement proposés concernent des réseaux locaux filaires. Le coût décroissant des technologies et composants réseaux et la multiplication du nombre d'équipements capables de communiquer font que l'informatique mobile est en plein essor. À cette évolution, s'ajoute le fait que des ressources de calcul sont de plus en plus déployées dans les réseaux (filaires ou non) ce qui permet de les utiliser pour offrir de nouveaux services toujours plus innovants. Il semble important de pouvoir continuer à offrir tous ces types de services y compris dans des réseaux ad-hoc : pouvoir découvrir une imprimante depuis un portable quand on est dans une conférence, découvrir une liaison Internet, un serveur de flux audio/vidéo, accéder à un service caméra pour un membbre d'une équipe de sauvetage, avoir accès à un serveur local d'informations dans un champ de bataille...

Nous allons, dans la section 2, présenter dans un premier temps les divers protocoles de découverte de services qui existent afin d'en résumer les caractéristiques générales. Dans la section 3, après une description succincte des réseaux ad-hoc et de leurs spécificités, nous présenterons dans la section 4 les raisons majeures qui font que ces protocoles ne sont pas parfaitement adaptés aux réseaux ad-hoc sans fil. Nous tenterons de répondre à ces divers problèmes en présentant l'architecture d'un protocole de découverte de services développé pour les réseaux ad-hoc. Finalement, nous conclurons en donnant diverses perspectives de recherche que nous menons à l'heure actuelle.

2. Tour d'horizon des protocoles de découverte de services

Le but de cette première partie est de présenter une étude des principales fonctionnalités que doit offrir un protocole de déploiement et de découverte de services et d'arriver à en déduire une architecture schématique générale. Cette étude se base sur des protocoles de découverte de services proposés par différents groupes de recherche. Nous avons considéré principalement trois protocoles. SDS (*Secure Discovery Service System*) [CZE 99] est un travail élaboré dans le cadre du projet Ninja [TEA] développé à l'université de Berkeley. SLP (*Service Location Protocol*) [VEI 97] est un protocole élaboré par J. VEIZADES, E. GUTTMAN, C. PERKINS and S. KAPLAN. SLP est proposé en tant que RFC au sein de l'IETF. Finalement, JINI [MIC] est un développement effectué chez SUN et basé sur Java permettant à un utilisateur de déployer et de découvrir des services.

Les services dans JINI et dans SDS sont restreints à des applications écrites en Java et les concepts proposés sont assez similaires : Java, code mobile, RMI, applications Java, services Java. JINI et SDS s'appuient aussi bien sur un échange d'informations que sur la technologie du code mobile pour déployer le protocole de déploiement et de découverte de services. Le concept général sous-jacent à ces deux approches est de fournir une configuration fluide d'objets réseau qui peuvent se déplacer d'un lieu à un autre et qui peuvent faire appel à n'importe quel autre objet pour assurer certaines opérations. Même si SLP n'emploie pas le concept de code mobile ni l'utilisation de RMI mais s'appuie principalement sur un échange de messages et donc d'informations, l'étude de ces trois protocoles permet de déduire une architecture commune. On a quelque soit l'approche employée la notion de *client* (entités cherchant des *services*), *provider* (entités possédant des *services*) et *server* (entité mettant en relation les *clients* et les *providers*). En détaillant ces trois approches, on peut mettre en valeur six fonctions principales qui caractérisent en quelque sorte la notion de « *protocole de découverte de services* ». Nous listons ces six fonctionnalités et présentons de façon succincte comment elles sont mises en œuvre dans les protocoles considérés.

Localisation du serveur. La notion de serveur fait le lien entre les services/providers et les clients. Localiser un serveur est donc un point clef pour tout protocole de découverte de services. SDS et SLP proposent deux types d'approches similaires permettant aussi bien à un client qu'à un provider de localiser un serveur. Dans la première approche, le serveur s'annonce périodiquement sur un canal multicast attribué au protocole. Dans la seconde approche, c'est le client ou le provider qui initie la localisation du service : par DHCP [PER 99], configuration statique ou multicast convergent dans SLP ou uniquement par multicast convergent dans SDS. Dans JINI, c'est le protocole nommé *Discovery* qui permet à un client ou à un provider de découvrir un serveur en multicastant une requête sur le réseau local.

Déclaration de services. Le deuxième aspect crucial est la façon dont les providers doivent déclarer leurs services. Dans SDS, les services doivent être annoncés périodiquement au niveau du serveur afin d'assurer le rafraîchissement des informations sur

les déclarations de services. Dans SLP et JINI, la déclaration d'un service se fait par enregistrement et le provider spécifie la durée de validité de cet enregistrement.

Envoi des requêtes dans un réseau local. Une fois le serveur localisé et les services enregistrés, reste à régler la façon dont les clients vont formuler et envoyer leur requête. Dans SDS, JINI et SLP et de façon tout à fait prévisible, un client envoie sa requête au serveur qu'il a découvert. Les trois approches traitent différemment le cas où un client n'a pas trouvé de serveur. Dans SDS, les annonces périodiques des services peuvent être écoutés par les clients qui sont donc informés des services de leur réseau local. Dans JINI, le client peut utiliser une technique dite *peer lookup*. Dans ce cas, le client envoie une requête aux providers pour qu'ils s'enregistrent auprès de lui (il joue le rôle de serveur) mais est en droit de ne sélectionner que les services dont il a besoin et de refuser les autres. Dans SLP, une requête peut être multicastée au sein du réseau et tout provider la recevant doit y répondre s'il est en mesure de la satisfaire.

Passage à grande échelle et routage des requêtes entre serveurs. Jusqu'à présent, nous nous sommes placés dans le cas des réseaux locaux. Dans l'hypothèse où une requête ne peut pas être satisfaite localement, *i.e.*, auprès du serveur local, il semble opportun de pouvoir la rediriger vers d'autres serveurs et permettre de trouver une éventuelle réponse positive. Ce passage à grande échelle [PER 98, ROS 98] engendre de nombreux problèmes concernant bien évidement le routage des requêtes d'un serveur à l'autre mais aussi la façon dont les serveurs ont conscience de leur interconnexion, selon quelle topologie et quelles doivent être les informations échangées entre serveurs. De part l'utilisation de DHCP ou de multicast convergent, SLP n'est pas très extensible de nature. SDS traite du passage à grande échelle de façon la plus poussée via l'utilisation de deux techniques : la mise en œuvre d'une hiérarchie entre les serveurs et l'agrégation des informations concernant les services au sein de cette hiérarchie

Robustesse. La robustesse d'un protocole de déploiement et de découverte de services concerne principalement la défaillance d'un serveur et/ou d'un provider. La panne d'un service est gérée dans SDS par les annonces périodiques du provider alors que dans SLP et JINI, la panne d'un service peut porter atteinte à la robustesse du protocole. Dans JINI, quand un serveur n'est pas présent, la technique *peer Lookup* est utilisée. Notons que JINI propose un mécanisme intéressant permettant à un client de s'abonner aux différents évènements qui peuvent survenir au niveau d'un service et ce dernier doit en notifier les clients. Dans SLP, au niveau d'un réseau local, quand un serveur tombe en panne, un client peut découvrir les services présents en envoyant une requête multicast et non plus unicast vers le serveur. Dans ce cas, tous les services présents qui sont en mesure de répondre à la requête doivent le faire.

Sécurité. Découvrir et accéder à des services sur un réseau soulève inévitablement la question de la sécurité et de la confidentialité. Ces aspects ont été pris en compte dans SDS. En effet, les auteurs ont traité le contrôle d'accès des clients aux services non publics et ont utilisé aussi bien l'authentification que le cryptage pour assurer les communications entre les composants du protocole de déploiement et de découverte de services. Dans SLP, la sécurité n'est pas aussi performante que celle présentée dans

SDS. En effet, l'intégrité des informations sur les services est garantie, mais les clients ne s'authentifient pas. Dans JINI, la sécurité n'a pas été traitée.

3. Les réseaux ad-hoc sans fil

L'architecture d'un réseau ad-hoc sans fil peut se caractériser par une absence d'infrastructure fixe préexistante, à l'inverse des réseaux filaires ou des réseaux sans fil cellulaires tels que nous les connaissons dans les télécommunications classiques (GSM par exemple). Les nœuds d'un réseau ad-hoc sont des hôtes mobiles ayant des capacités et une puissance de transmission identiques et possédant une certaine puissance de calcul. Un tel réseau doit s'organiser automatiquement de façon à être déployable rapidement et pouvoir s'adapter aux conditions de propagation, aux trafics et aux différents mouvements pouvant intervenir au sein des nœuds mobiles.

Du fait de l'absence d'une architecture fixe et de la mobilité des nœuds, la topologie d'un réseau ad-hoc peut évoluer à tout instant. Afin que le réseau reste connecté au sens classique du terme (tout sommet peut atteindre tout autre), chaque sommet 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 directement sa destination. Tout nœud joue ainsi le rôle d'hôte et de routeur. Le groupe de travail de l'IETF étudie ce genre de problème. Nous nous sommes plus particulièrement intéressés au réseau ad-hoc de type HIPERLAN, mais les principes de notre étude s'applique à tout autre réseau ad-hoc. La topologie d'un réseau ad-hoc évoluant à chaque instant, nous allons la modéliser par un graphe $G_t = (V_t,E_t)$ où V_t représente l'ensemble des nœuds et E_t modélise l'ensemble des connexions existantes entre les nœuds du réseau. Il existe une arête $e = (u,v) \in E_t$ si les nœuds u et v sont en mesure de communiquer directement à l'instant t.

Une des caractéristiques primordiales des réseaux sans fils est l'utilisation d'un médium de communication partagé qui est le canal de communication hertzien dans HIPERLAN [BAR 97, TCR 95]. Une transmission sélective se révèle donc impossible : lorsqu'un nœud émet, tous ses voisins, *i.e.*, tous les nœuds se trouvant dans la zone de couverture du sommet émetteur, vont recevoir le message et une collision se produit si tout autre sommet voisin tente de communiquer en même temps. Cette caractéristique fondamentale se traduit par la propriété de base suivante valable à chaque instant t au sein de chaque sous graphe complet (clique) du graphe G_t :

Propriété 1 *Pour toute clique C de G_t, au plus un nœud émet un message à l'instant t et lorsqu'un des nœuds de la clique émet un message, tous les autres nœuds de la clique peuvent entendre ce même message.*

Cette propriété nous permet de déduire qu'a chaque instant t il existe une décomposition du réseau en cliques. Une telle décomposition en cliques est représentée par une famille $\mathcal{C} = C_1, \ldots, C_\ell$ de cliques du graphe et vérifie que l'union des cliques de

la décomposition est égale au graphe G_t. Une telle décomposition s'avère très utile pour mettre en œuvre des politiques de routage [COH 00].

4. Protocole de déclaration et de découverte de services dans un réseau ad-hoc

Nous nous intéressons au problème de la découverte de services dans un tel contexte, *i.e.*, ensemble d'entités mobiles constituées de ressources de calcul et d'au moins une interface de communication sans fil. Les échanges d'informations s'effectuent directement entre les hôtes mobiles, sans pouvoir compter sur une infrastructure préexistante (bornes fixes par exemple). La découverte (et à terme le déploiement) de services dans ce contexte doit répondre à plusieurs critères spécifiques au type de réseaux employé. Il faut bien évidement se conformer aux caractéristiques fondamentales d'un tel protocole telles que nous avons pu les décrire dans la section 2. Il faut que le protocole prenne en compte la mobilité (des connexions peuvent s'évanouir et d'autres peuvent se créer à tout moment) qui caractérise de façon intrinsèque les hôtes et ce de façon la plus transparente possible pour les clients : en se déplaçant au sein d'une conférence, il est souhaitable de ne pas perdre le service « *impression* ». Il semble important que le protocole puisse permettre d'effectuer des optimisations notamment sur les ressources réseaux utilisées : il est préférable d'utiliser un service présent dans son voisinage (par exemple un serveur vidéo) plutôt que d'être obligé d'aller le chercher « *loin* » perturbant ainsi beaucoup de monde tout au long du chemin.

Nous le voyons, vouloir mettre en œuvre un protocole de découverte de services au sein d'un réseau ad-hoc complexifie encore d'avantage le problème général et soulève aussi de nouvelles questions. Nous allons dans un premier temps mettre en évidence les raisons majeures qui font que les protocoles présentés dans la section 2 ne sont pas adaptés en l'état et ce en se basant sur les différences fondamentales qui existent entre les deux, même, si l'on tentera de construire un parallèle entre réseaux fixes et réseaux ad-hoc afin de pouvoir conserver le schéma d'architecture général que nous avons mis en évidence : serveur, provider, client.

4.1. *Comparaison entre réseau fixe et réseau ad-hoc*

Afin d'appliquer l'un des protocoles de déploiement et de découverte de services dans les réseaux locaux présentés dans la section 2 nous devons identifier quels sont les composants des réseaux ad-hoc qui vont jouer le rôle des serveurs, des providers et des clients. A priori, il n'y a pas lieu de changer la signification ni des clients ni des providers : à chaque instant t, tout nœud $u \in V_t$ du réseau ad-hoc demandant un service est un client et tout nœud $u \in V_t$ proposant un ou plusieurs services est un provider de services. Considérons maintenant le cas du serveur. Dans un réseau fixe, la mise en œuvre du serveur dans le protocole de services a principalement pour but le passage à l'échelle et doit permettre de transmettre aussi bien les déclarations de services que les requêtes d'un sous-réseau à un autre. Cette comparaison nous amène à faire le parallèle entre la notion de sous-réseaux et la notion de cliques. La question

soulevée est alors de savoir s'il y a lieu de mettre en œuvre des serveurs dans un réseau ad-hoc et dans l'affirmatif, doivent-ils jouer le même rôle ? Par cette interrogation, on retrouve les deux classes d'approches mises en œuvre pour le routage dans les réseaux ad-hoc : approches dites pro-actives et approches dites réactives [JOH 94].

Si l'on décide de répondre par la négative, deux choix s'offrent à nous : soit chaque provider diffuse périodiquement la totalité des services qu'il offre et dans ce cas, chaque client est tenu de mettre à jour la liste des services qu'il écoute ; soit les services ne sont pas diffusés et c'est le client qui diffuse sa requête et dans ce cas, chaque provider étant en mesure de satisfaire une requête se doit d'y répondre ou du moins prévenir le client de sa présence. L'inconvénient de la première approche est qu'elle génère une charge de trafic inutile due à la diffusion périodique dans tout le réseau des services. Un nœud n'est pas forcément intéressé par les déclarations de services en provenance d'une clique éloignée si il a le même service dans son propre voisinage. De même, la deuxième approche nécessite la diffusion des requêtes dans l'ensemble du réseau et ce chaque fois qu'un nœud cherche un service. Ces deux approches nous semblent mal adaptées car elle font toutes deux une utilisation de la diffusion sans chercher à optimiser les ressources réseaux et chacune d'elle laisse le choix du service au client ce qui ne permet pas de mettre en œuvre des politiques d'optimisation globale des ressources (service le plus proche) ou de répartition de charge [STE 99, FOX 97]. Afin de palier au problème de la diffusion , il nous semble important d'introduire un troisième composant jouant le rôle de serveur que nous avons choisi de le nommer *médiateur* (pour se démarquer de l'approche réseau fixe). Nous présentons dans la section suivante 4.2 quelles vont être ses fonctions et quelles sont les propriétés qu'il doit garantir.

4.2. *Définition du composant médiateur*

Le médiateur va jouer un rôle très similaire à celui du serveur. Pour éviter les diffusions dans tout le réseau, un médiateur ne couvrira qu'une zone de services réduite : un médiateur sera responsable de tous les services qu'il écoute, donc, de tous les services présents dans les cliques auxquelles il appartient. En effet, si un médiateur $m \in C_1 \cap C_2 \cap \ldots \cap C_k$ il reçoit par définition tous les messages émis dans les cliques C_i pour $1 \leq i \leq k$. La figure 1 illustre l'analogie que l'on peut faire entre médiateur et serveur. Son rôle est de servir d'intermédiaire entre les clients et les providers de services. Néanmoins, du fait que nous sommes dans un contexte de réseau ad-hoc, nous ne pouvons pas supposer l'existence d'un médiateur fixe connu de tous. De plus, le ou les médiateurs étant des entités comme les autres, ils sont mobiles et peuvent à tout moment s'évanouir. Il nous faut donc spécifier où doivent se placer les médiateurs, comment gérer le fait qu'ils peuvent bouger au sein du réseau, décrire la façon dont les services doivent se déclarer et réciproquement la façon dont un client recherche un service.

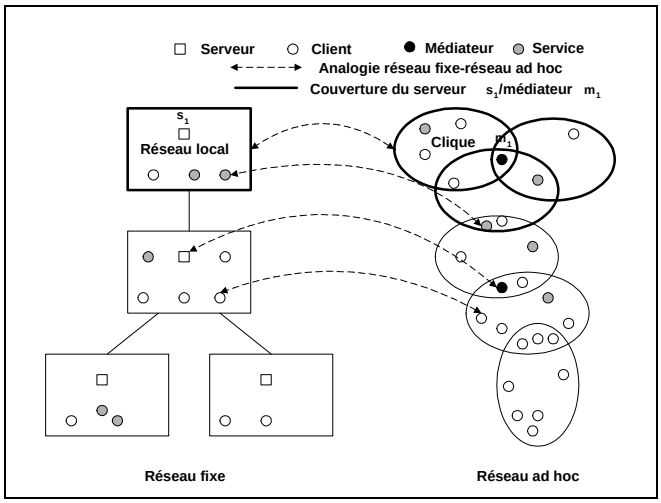


FIG. 1 –. *Analogie réseau fixe-réseau ad-hoc.*

4.3. *Déclaration de services*

Supposons dans un premier temps qu'il existe un médiateur capable d'écouter les services présents dans $\mathcal{C} = C_1 \cup C_2 \cup \ldots \cup C_k$. Chaque service d'un provider $p \in C_1 \cap C_2 \cap \ldots \cap C_k$ ne doit être déclaré qu'au sein de la ou des cliques auxquelles il appartient. Étant donné que tout sommet $u \in \mathcal{C}$ reçoit les messages d'annonce de p, il peut en profiter pour mettre à jour la liste des services qui sont présents dans son voisinage direct. Chaque nœud d'une clique est donc au courant de tous les services déclarés par les nœuds de cette clique. Cette approche garantit à un client de trouver un service immédiatement au sein de sa clique si ce dernier est présent et permet de bénéficier du message d'enregistrement émis par un provider pour informer le reste des autres sommets de l'existence des services présents dans leur voisinage direct.

4.4. *Gestion du médiateur*

L'analogie que nous venons de présenter (cliques/réseau local et serveur/médiateur) n'implique pas la gestion d'un médiateur par clique. Notre but est de n'avoir des médiateurs que dans les cliques où il y a des providers, donc des services déclarés. Il n'est pas nécessaire de perturber inutilement une partie du réseau qui ne possède aucun service. La notion de médiateur est introduite pour :

1. permettre aux services de se déclarer afin d'éviter de diffuser les déclarations dans tout le réseau ;
2. permettre aux clients de chercher un service auprès d'eux.

Ils jouent donc bien un rôle d'intermédiaires entre les services et les clients. Nous décrivons maintenant les propriétés qui doivent être garanties par les médiateurs au sein d'un réseau ad-hoc.

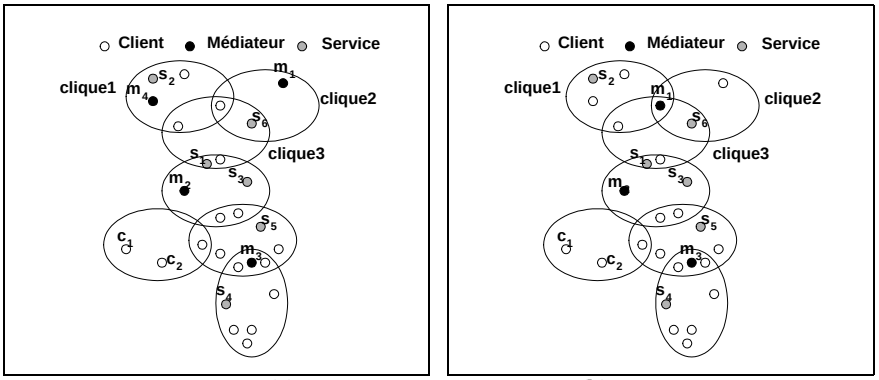


FIG. 2 –. *(a) : Exemple de configuration de médiateurs dans un réseau ad-hoc. (b) : Exemple de configuration optimisée de médiateurs dans un réseau ad-hoc*

Propriété 2 *Afin de jouer leur rôle, les médiateurs doivent garantir un certain nombre de propriétés listées ci-dessous. Pour un sommet $u \in V_t$, on note $\mathcal{C}_u = \{C_i | u \in C_i, 1 \leq i \leq k\}$ l'ensemble des cliques auxquelles il appartient. Notons que tout sommet appartient au moins à une union de cliques (qui peut se réduire à lui seul).*

1. Si un nœud u du réseau est provider alors il existe un médiateur $m \in \mathcal{C}_u$ dans son champ d'écoute qui le prend en charge. De plus, chaque provider connaît le ou les médiateurs qui le prennent en charge ;

2. Si un médiateur du réseau n'écoute plus aucun service alors il ne peut pas être médiateur ;

3. Il ne peut pas y avoir plusieurs médiateurs annonçant l'écoute d'un même ensemble de services offerts par un même ensemble de providers ;

4. La position du médiateur doit permettre de minimiser le trafic. Prenons l'exemple illustré sur la figure 2. Dans la figure de gauche (a) , m_1 est médiateur du service s_6 et m_4 est médiateur du service s_2. Dans la figure de droite (b) m_1 est médiateur des services s_1, s_2 et s_6. Ceci permet de minimiser localement le nombre de médiateurs et donc a priori le nombre d'émissions nécessaires pour que des médiateurs puissent s'échanger des informations. Le placement des médiateurs est très complexe dès que l'on veut appliquer des critères de minimisation globale (vouloir minimiser le nombre de médiateurs par exemple est un problème NP complet puisqu'il s'apparente au MINIMUM DOMINATING SET *[GAR 79]).*

Une fois cet ensemble de propriétés énoncé, il nous faut maintenant étudier les mécanismes à mettre en œuvre afin de garantir qu'elles soient respectées au sein du réseau ad-hoc et donc prendre en compte la mobilité de tout sommet (médiateur, client, provider).

– La propriété 2.1 visant à garantir que tout service est couvert par au moins un médiateur est maintenue par le fait qu'un médiateur va devoir annoncer régulièrement, dans son voisinage direct, la liste des services qu'il couvre ainsi que l'identité du provider qui a déclaré le service. Les providers de services vont donc devoir inclure dans leur déclaration périodique leur identité ainsi que l'identité des services qu'ils offrent. Notons que le cas ou un provider change de clique est couvert par le mécanisme décrit dans la section 4.3. Si un provider est le seul à proposer des services au sein d'un voisinage ou s'il n'entend aucun médiateur, il va « *s'auto proclamer* » médiateur ;

– La propriété 2.2 permettant de réduire la diffusion de trafic inutile est facilement garantie par le fait qu'un médiateur va écouter les annonces de services faites par les providers de son voisinage. Si il ne reçoit pas les annonces de rafraîchissement correspondant aux services alors il les supprime et s'il s'avère qu'il ne gère plus aucun service alors il se retire ;

– La propriété 2.3 tend aussi à minimiser la diffusion de trafic inutile. Pour un ensemble donné de cliques, une multiplicité des médiateurs ne peut être justifiée que par la différence des services qu'ils écoutent. Chaque médiateur va devoir écouter non seulement les annonces diffusées par les providers mais aussi celles émises par les autres médiateurs qui sont dans son voisinage. Un médiateur qui se rend compte que la liste des services qu'il couvre est inclue dans la liste de services couverte par un autre médiateur doit se retirer en cessant d'envoyer sa déclaration en tant que médiateur. Si l'on considère l'exemple précédent illustré par la figure 2, les médiateurs m_1 et m_4 devront se retirer ;

– La propriété 2.4 est la plus délicate car elle est qualitative. Un peu sur le principe de « *l'auto-proclamation* » utilisée pour garantir la propriété 1, on va l'appliquer dans le cas ou un autre sommet découvre qu'il couvre un nombre strictement supérieur de services que ceux diffusés par un médiateur. Dans ce cas, après une « *auto évaluation* », il considère qu'il est mieux à même de gérer les services qu'il couvre et se déclare médiateur. Les autres médiateurs, devront alors appliquer la propriété 3 et se retirer.

Notre protocole assure que tous les services sont déclarés au niveau des médiateurs. Donc, l'ensemble des médiateurs offre une image complète des services déclarés dans tout le réseau. Il nous reste à résoudre la façon dont un client va découvrir un service. Le cas où le service recherché est présent dans la clique est trivial et a déjà été traité dans la section 4.3. Le problème se pose réellement quand un client est dans une clique où aucun médiateur n'est présent et donc aucun médiateur (propriété 2.2), ou lorsque le service recherché n'a pas été enregistré auprès des médiateurs présents. Il faut donc trouver un moyen permettant de relier ces médiateurs afin de pouvoir en retirer l'information nécessaire.

4.5. *Gestion globale des médiateurs*

Un client doit être en mesure d'accéder à un médiateur puisque dans notre approche il représente une sorte de point d'entrée vers l'ensemble des services offerts. Dans les sections suivantes nous allons spécifier la façon dont un client va trouver un médiateur et le degré d'interaction nécessaire dont les médiateurs ont besoin pour pouvoir résoudre une requête. Dans l'état actuel de nos travaux, nous nous sommes intéressés au routage applicatif sous-jacent qui existe dans le réseau ad-hoc [TCR 95, COH 00, BAR 97, JAC 98] car il permet de ne pas se soucier, dans un premier temps, de la mobilité des sommets. Donc, sous l'hypothèse que l'on a trouvé un médiateur, le routage d'une requête est supposé réalisable même si le client et/ou le médiateur se déplacent. De même, nous ne traiterons pas ici de la spécification des requêtes qui est a priori indépendante de la nature du réseau sous-jacent même si nous envisageons de traiter par la suite des types de services dédiés aux réseaux ad-hoc.

4.5.1. *Agrégation des médiateurs*

L'idée sous-jacente est de créer une structure à laquelle vont participer tous les médiateurs afin de mettre en commun toute ou une partie de leur connaissance. L'un des buts étant d'optimiser le nombre des ressources du réseau, nous nous sommes acheminés vers l'utilisation d'une adresse multicast [COH 00, BOM 98, WU 98]. Cette structure devra pouvoir servir aux clients pour pouvoir accéder à tous les services du réseau ad-hoc sans pour autant perturber les parties du réseau (cliques) où ne se trouve ni médiateur ni client. Le problème qui se pose alors est la gestion de ce groupe de multicast, c'est-à-dire, doit-on l'utiliser à des fins d'échanges d'information entre les médiateurs et/ou plus simplement afin de propager les requêtes. Face à cette alternative, nous pensons qu'il est préférable de réaliser un compromis entre la propagation des déclarations des services (si on a un grand nombre de médiateurs, le trafic généré au sein de l'arbre multicast peut devenir important) et la propagation des requêtes (si chaque requête de chaque client est diffusée dans l'arbre cela risque aussi de créer beaucoup de trafic). Ce compromis, dépendant à priori du nombre de clients et du nombre des médiateurs, peut être assuré en tenant compte de différents critères. D'une part on veut minimiser le trafic généré ce qui peut se faire en minimisant la portée des requêtes ainsi que celle des déclarations des services. D'autre part, pour une requête émise, on cherche à restreindre les duplications des réponses afin de limiter le trafic bien sûr mais aussi afin de ne pas déporter le choix au sein des clients mais d'en conserver cette maîtrise au niveau des médiateurs puisqu'ils bénéficient d'une structure les regroupant qui peut servir à terme à mettre en œuvre des politiques globales, de répartition de charge par exemple.

4.5.2. *Acheminement des requêtes*

Dans l'approche que nous proposons et qui fait usage d'une agrégation de médiateurs, l'acheminement d'une requête vers le service le plus proche se fait suivant la

hiérarchie suivante :

1. Si le service existe dans une des cliques auxquelles appartient le client alors il est présent dans son cache et le client peut directement envoyer sa requête au provider associé au service demandé. Notons que l'utilisation d'un cache augmente la vitesse à laquelle une requête peut être routée vers un provider mais en contre partie ne permet pas une gestion globale des providers ;
2. Sinon, il s'adresse aux médiateurs qui sont dans son voisinage si ceux ci existent car ces médiateurs peuvent être en écoute d'autres services qui ne sont pas dans la même clique que le client. Si l'un des médiateurs est capable de satisfaire la requête, il renvoie au client l'adresse du provider associé. Sinon, se pose le problème de l'acheminement de la requête entre les médiateurs ;
3. Si aucun médiateur n'est présent dans son voisinage, se pose le problème de pouvoir acheminer la requête vers un médiateur distant.

Étant données les hypothèses et les données dont disposent les médiateurs et les clients, il semble logique de devoir utiliser l'adresse multicast pour résoudre les deux derniers problèmes. Cependant, l'acheminement de la requête va être fortement liée à la gestion des déclarations de services au sein de ce même arbre de multicast et nous allons traiter les deux problèmes conjointement dans la section suivante.

4.5.3. *Gestion des déclarations de services*

L'idée d'agrégation des médiateurs via une adresse multicast permet a priori de chercher un service dans tout le réseau ad-hoc sans avoir à diffuser cette requête puisque les médiateurs représentent un ensemble dominant des providers.

Dans Ninja [TEA], l'agrégation des serveurs est assurée via une hiérarchisation construite comme un arbre recouvrant tous les serveurs. Cette agrégation permet aussi une diffusion restreinte des déclarations de services. Dans un réseau ad-hoc où les médiateurs sont toujours susceptibles de bouger et de disparaître, vouloir assurer une certaine hiérarchie prédéterminée entre les médiateurs est une tâche difficile surtout si cette hiérarchie a pour but de minimiser le trafic et d'offrir une gestion intelligente des déclarations de services au niveau des médiateurs.

Il est clair que dans un réseau fixe, la hiérarchie topologique entre les réseaux locaux est offerte par DNS. Malheureusement, dans un réseau ad-hoc cette notion de sous réseau n'existe pas et la notion de clique qui peut sembler similaire de part le parallèle que nous avons fait ne permet pas de nous baser sur une hiérarchie topologique en ad-hoc. C'est donc aussi pour cette raison que nous nous sommes intéressés à la gestion d'une adresse multicast regroupant les médiateurs indépendamment de leur topologie, c'est-à-dire, une gestion restreinte à l'échange des informations concernant les services. Nous présentons maintenant différentes approches possibles auxquelles nous avons pensé pour générer l'arbre des médiateurs. La première consiste à gérer de façon complètement distribuée les déclarations de services, *i.e*, les médiateurs ne s'échangent pas leur information alors que la seconde au contraire se base sur l'échange entre médiateurs des services qu'il couvrent.

4.5.4. *Gestion décentralisée des déclarations de services*

Si l'on suppose que les déclarations de services ne sont pas échangées entre les médiateurs, l'arbre multicast va être utilisé pour diffuser la requête. Tout médiateur qui reçoit une requête qu'il est capable de traiter, va pouvoir y répondre de façon positive.

Envoi de la requête du client vers l'adresse multicast. Étant donné que les requêtes sont multicastées en direction de l'arbre et s'il l'on ne rajoute aucun mécanisme d'élection entre les médiateurs qui sont en mesure d'y répondre, plusieurs médiateurs peuvent répondre à la même requête. L'unicité de la réponse n'est pas garantie dans ce cas. Ceci peut poser des problèmes d'encombrement inutiles au niveau des clients surtout si les diverses réponses sont toutes identiques. De plus on perd la maîtrise du choix du provider qui incombe au client.

4.5.5. *Diffusion des déclarations de services*

Afin de palier les inconvénients de l'approche proposée ci-dessus, nous allons considérer que tout médiateur est informé de l'ensemble des services présents dans le réseau. Pour chercher un service il suffit donc à un client d'être en mesure de contacter un seul médiateur. Ceci doit permettre de garantir l'unicité de la réponse. De plus, le choix du provider fait partie intégrante du protocole de gestion de l'arbre des médiateurs et n'est plus laissé à la charge des clients. Cela permet d'assurer une plus grande indépendance entre le client et le protocole de découverte de services. Une façon simple pour que chaque médiateur soit informé de l'ensemble des services couverts par les autres est de diffuser ses informations dans l'arbre de multicast.

Envoi de la requête du client vers l'un des médiateurs. Comme tous les médiateurs possèdent les mêmes informations il faut que l'un d'entre eux prenne en charge la requête. L'idéal est bien entendu d'avoir un équivalent du ANYCAST introduit dans IPv6. Nous donnons diverses approches permettant de répondre de façon plus ou moins satisfaisante à ce problème :

– Une approche simple consiste à envoyer un message au groupe multicast des médiateurs leur demandant de retourner leur adresse (on peut ajuster la probabilité avec laquelle vont répondre les médiateurs en fonction du nombre de sommets dans le groupe). Le client peut alors en choisir un et lui adresser sa requête ;

– Une autre approche tout aussi simple est de considérer que seul un médiateur est autorisé à répondre, par exemple celui qui possède la plus petite adresse au sein du groupe des médiateurs (il est assez facile de l'élire puisque chacun diffuse régulièrement l'ensemble de ses services) ou alors le « core » du multicast si on se base sur une structure d'arbre partagé. L'inconvénient est que l'on a une structure distribuée et que l'on utilise seulement un membre de cette structure pour traiter toutes les requêtes ;

– Si le protocole de multicast offre une latence d'adhésion performante, un client peut se déclarer comme médiateur et donc joindre le groupe des médiateurs bien que ne déclarant aucun service (ou alors un service fictif). Cette participation dure uniquement le temps nécessaire pour que le client reçoive toutes les déclarations des autres

médiateurs et dès qu'il les a récupérés il quitte le groupe des médiateurs. Cette dernière solution n'est pas très viable car elle risque de dégénérer en une structure où l'arbre de multicast que l'on voulait restreindre (il ne doit couvrir que les médiateurs, *i.e.*, les cliques où se trouve des services) risque de se transformer en un arbre recouvrant la majorité des sommets du graphe ;

– Il semble en fait plus judicieux de mettre en œuvre un multicast convergent qui s'arrête dès qu'il a atteint un membre du groupe des médiateurs. Le problème est que plusieurs membres peuvent être atteints en même temps. Il faut donc soit mettre en œuvre un mécanisme d'élection entre les différents médiateurs soit considérer que la requête sert dans un premier temps a connaître l'identité des médiateurs les plus proches (multicast convergent) et dans un deuxième temps le client envoie effectivement sa requête de service vers l'un des médiateurs qui lui a répondu. Malgré le fait que les médiateurs s'échangent leurs informations, cette approche a l'avantage de répartir les requêtes sur l'ensemble des médiateurs si les clients sont distribués dans le réseau. De plus, une fois qu'un client a demandé un service, il connaît a priori l'adresse d'un médiateur et donc peut lui adresser directement ses requêtes de service.

Alors que cette approche peut paraître plus puissante puisque tous les médiateurs possèdent les mêmes informations, elle entraîne plusieurs problèmes dus au fait qu'aucun niveau de structuration des déclarations des services n'est présent ce qui rend difficile l'optimisation de l'envoi d'une requête du client. Néanmoins, nous allons dans la section suivante indiquer des solutions envisageables si l'on emploie les concepts introduits dans les réseaux actifs [AND 00, WET 99].

4.6. *Vers des réseaux ad-hoc actifs*

L'architecture du protocole proposé dans ce papier gagnerait beaucoup à l'utilisation des concepts introduits dans les réseaux actifs. En effet, la gestion des médiateurs peut être grandement optimisée si l'on peut introduire du code au sein des sommets effectuant la gestion de l'arbre multicast par exemple. Dans l'approche où les déclarations de services sont diffusées au sein du groupe, cette diffusion peut être optimisée si un sommet de l'arbre multicast ne rediffuse que l'union des services qu'il vient d'apprendre et de ceux qu'il connaissait déjà. Ceci sous-entend que le protocole de multicast permet d'appliquer un traitement, même restreint sur les paquets d'informations qui sont diffusés. On peut aussi remplacer le multicast convergent de la dernière approche par une très légère modification du protocole d'enregistrement à un groupe de multicast (`JOIN`) ou plus exactement à la façon dont un nouveau membre s'attache à l'arbre. En effet, le multicast convergent que l'on utilise sert uniquement à découvrir un nœud du réseau faisant partie de l'arbre ce que fait par nature un `JOIN`. Il est alors très simple en utilisant les réseaux actifs, par exemple ANTS, de modifier le code d'un `JOIN` pour que dès qu'il arrive sur un membre de l'arbre il renvoie un paquet à destination du client.

En fait, dans les diverses approches présentées pour gérer les médiateurs, aussi bien pour la propagation des déclarations de services que des requêtes, il apparaît très

utile d'employer une structure de multicast pour unifier un ensemble d'entités (les médiateurs) mais le besoin de pouvoir par exemple modifier la diffusion d'un paquet ou son contenu est apparue à plusieurs reprises et milite en faveur de l'introduction d'un protocole de multicast actif, au sens où ce dernier doit donner à l'application les moyens de modifier le traitement par défaut qui est effectué sur les paquets diffusés.

Un autre avantage des réseaux programmables est qu'ils permettent aux couches supérieures (applicatives par exemple) d'avoir accès à des informations plus spécifiques aux couches réseaux mais qui peuvent s'avérer utiles pour l'application. Par exemple, dans le cas où un client a émis une requête et qu'il dispose de plusieurs réponses, plusieurs critères d'optimisation sont envisageables tels que la répartition de charge, la proximité du service... Nous pensons que l'actif jouera un rôle important pour permettre par exemple d'avoir accès, au niveau applicatif du protocole de déploiement et de découverte de services, à l'information sur la distance du service par rapport au client.

5. Conclusion

Dans cet article, nous avons présenté notre approche pour un protocole de déploiement et de découverte de services dans un environnement ad-hoc sans fil. La contrainte de mobilité des membres ad-hoc nous a amené à opter pour une approche qui permet de couvrir les services déclarés dans tout le réseau. Cette approche consiste à faire participer les membres responsables de services (qu'on a appelés médiateurs) à une adresse multicast globale. Nous avons mis en avant l'apport introduit par l'utilisation des réseaux actifs car ils permettent d'optimiser les échanges des déclarations de services ainsi que le traitement des requêtes de client au sein de l'arbre des médiateurs.

Notre travail à venir porte sur l'optimisation et la mise en œuvre du protocole ad-hoc proposé dans cet article et pensons utiliser la technologie active pour aboutir à cette optimisation. Nous désirons aussi aborder d'autres aspects de la découverte de services qui ne sont pas traités dans cet article : définition de services spécifiques à un réseau ad-hoc, nécessité de renégocier un service du fait de la mobilité du provider et/ou du client, composition de services [ELA 98].

6. Bibliographie

[AND 00] ANDREY L., CHRISMENT I., FESTOR O., FLEURY E., « *Sytèmes multimédia communicants* », Chapitre Infrastructures pour le multimédia : ALF et les réseaux actifs, IC2, Hermes, 2000, (To appear).

[BAR 97] BARTON S. K., Ed., *Wireless Personal Communication: Special issue on the High PErformance Radio Local Area Network (HIPERLAN)*, vol. 4, Kluwer Academic Publishers, January 1997.

[BOM 98] BOMMAIAH, MCAULEY, TALPADE, LIU, « AMRoute: Adhoc Multicast Routing Protocol », INTERNET-DRAFT n° draft-manet-amroute-00.txt, August 1998, IETF.

[COH 00] COHEN J., FLEURY E., GUSTEDT J., « JUMBO : protocole de routage unicast et multicast dans les réseaux ad-hoc sans fils », rapport n° (en cours), 2000, INRIA.

[CZE 99] CZERWINSKI S. E., ZHAO B. Y., HODES T. D., JOSEPH A. D., KATZ R. H., « An Architecture for a Secure Service Discovery Service, », *Proc. IEEE Mobicom'99*, Seattle, Washington, USA, 1999.

[ELA 98] ELAN AMIR S. M., KATZ R., « An Active Service Framework and its Application to Real-time Multimedia Transcoding », 1998, Submitted for publication.

[FOX 97] FOX A., GRIBBLE S. D., CHAWATHE Y., BREWER E. A., GAUTHIER P., « Cluster-Based Scalable Network Services », *Proceedings of the 16th Symposium on Operating Systems Principles (SOSP-97)*, vol. 31,5 de *Operating Systems Review*, New York, 5–8 1997, ACM Press, p. 78–91.

[GAR 79] GAREY M., JOHNSON D., *Computers and Intractability: A Guide to the theory of NP-Completeness*, a Serie of books in mathematical sciences, Freeman édition, 1979.

[JAC 98] JACQUET P., MUHLETHALER P., QAYYUM A., « Optimized Link State Routing Protocol », Internet Draft n° draft-ietf-manet-olsr-00.txt, November 1998, IETF MANET Working Group.

[JOH 94] JOHNSON D. B., « Routing in Ad Hoc Networks of Mobile Hosts », *IEEE Workshop on Mobile Computing Systems and Applications*, December 1994.

[MIC] MICROSYSTEMS S., « Jini technology specifications. white paper. », http://www.sun.com/jini/specs/.

[PER 98] PERKINS C., « Wide Area Service Location Protocol », http://www.bell-labs.com/mailing-lists/wasrv/, March 1998.

[PER 99] PERKINS C., GUTTMAN E., « RFC 2610: DHCP Options for Service Location Protocol », June 1999, Status: PROPOSED STANDARD.

[ROS 98] ROSENBERG G., GUTTMAN E., MOATS R., SCHULZRINNE H., « WASRV Architectural Principles », Internet Draft n° draft-rosenberg-wasrv-arch-00.txt, February 1998, IETF.

[STE 99] STEVEN D. GRIBBLE MATT WELSH E. A. B., CULLER D., « The MultiSpace: an Evolutionary Platform for Infrastructural Services, », *Proc. Usenix Annual Technical Conference*, Monterey, CASeattle, June 1999.

[TCR 95] TC-RES E., « Radio Equipement and System (RES); High PErformance Radio Local Area Network (HIPERLAN), Type 1, Functional specification », rapport n° ETS 300 652, December 1995, European Telecommunication Standards Institute, Draft.

[TEA] TEAM T. N., « The Ninja Project », http://ninja.cs.berkeley.edu/.

[VEI 97] VEIZADES J., GUTTMAN E., PERKINS C., KAPLAN S., « RFC 2165: Service Location Protocol », June 1997, Status: PROPOSED STANDARD.

[WET 99] WETHERALL D., « Active network vision and reallity: lessons from capsule-based systems », *Operating Systems Review*, vol. 35, n° 4, 1999, p. 64–79, 17th ACM Symposium on Operating Systems Principles (SOSP'99).

[WU 98] WU C., TAY Y., TOH C.-K., « Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) », INTERNET-DRAFT n° draft-ietf-manet-amris-spec-00.txt, November 1998, IETF MANET Working Group.