Conception d'une plate-forme d'expérimentation pour réseaux ad hoc et hybrides - Application à l'évaluation d'un protocole d'auto-organisation et de routage

Fabrice Theoleyre, Fabrice Valois

Conception d'une plate-forme d'expérimentations pour réseaux ad hoc et hybrides

Application à l'évaluation d'un protocole d'auto-organisation et de routage

Fabrice Théoleyre[†] — **Fabrice Valois**[§]

[†] *CNRS, Laboratoire Informatique de Grenoble (LIG)*
681 rue de la passerelle, 38402 Saint Martin d'Heres Cedex - France
fabrice.theoleyre@imag.fr

[§] *CITI, INSA Lyon / INRIA Rhône-Alpes*
21, Avenue Jean Capelle, 69621 Villeurbanne Cedex - France
fabrice.valois@insa-lyon.fr

RÉSUMÉ. *L'évaluation des réseaux* ad hoc *est essentiellement faite par simulation bien que la propagation radio soit modélisée de façon non fidèle dans les simulateurs réseau. Il est donc nécessaire de valider les protocoles à l'aide d'une démarche expérimentale. Nous proposons ici l'implémentation et le déploiement d'une plate-forme de tests pour réseaux* ad hoc *interconnectés à internet, permettant la création d'un internet multi-saut. Nous fournissons un descriptif matériel et logiciel détaillé pour constituer une telle plate-forme de tests afin de valider tout protocole pour réseau* ad hoc *ou hybride. Nous évaluons sur cette plate-forme les performances d'un protocole d'auto-organisation et de routage. Les performances obtenues démontrent la faisabilité et la pertinence d'une telle approche.*

ABSTRACT. *Protocols for MANets are often evaluated through simulations, although radio propagation is not modeled accurately in simulators. We must consequently validate the performances of different protocols through an experimental approach. We describe here the implementation and deployment of a complete testbed for ad hoc networks connected to the internet, that creates a multihop wireless internet. We provide a complete description of software and hardware requirements to constitute such a testbed in order to validate a protocol for any ad hoc or hybrid network. We evaluate with this testbed the performances of one self-organization and routing protocol. The performances we obtained validate the feasibility and relevance of this approach.*

MOTS-CLÉS : *réseaux hybrides, plate-forme d'expérimentations, auto-organisation, routage*

KEYWORDS: *hybrid networks, testbed, self-organization, localization*

RSTI - TSI. Volume 28 – n° 5/2009, pages 677 à 701

1. Introduction

Un réseau *ad hoc* (MANET) est littéralement un réseau *prêt à l'emploi* : des terminaux peuvent communiquer spontanément *via* des liaisons radio, sans infrastructure fixe préalable. Le réseau devant fonctionner de façon autonome, sans intervention humaine, les terminaux doivent donc collaborer. Ainsi lorsqu'un terminal souhaite envoyer des données à un destinataire qui n'est pas à portée radio, d'autres nœuds intermédiaires doivent relayer ce paquet. Les réseaux *ad hoc* suppriment la distinction usuelle des réseaux classiques entre les clients et les équipements dédiés routant les informations. Par ailleurs, dans un réseau *ad hoc*, les terminaux sont mobiles, conduisant à des apparitions et disparitions de liens radio, obligeant le réseau à s'auto-organiser pour prendre en compte une telle dynamique. Les réseaux *ad hoc* permettent d'échanger des informations dans un environnement mobile. Ainsi, un entrepôt peut mettre en place un réseau *ad hoc* comme prolongation de son système d'informations, ou une compagnie de taxis peut créer un réseau sur licence libre pour répartir les clients en attente (Huang *et al*., 2005). Les réseaux *ad hoc* peuvent également être connectés à internet *via* un équipement dédié, le point d'accès (AP), jouant le rôle de passerelle entre le monde filaire et la bulle *ad hoc*. De tels réseaux sont appelés *réseaux hybrides* et constituent de véritables *réseaux cellulaires multisaut*. Nous pensons que les réseaux hybrides constituent une évolution naturelle des réseaux de périphérie. Un ensemble de clients d'un opérateur pourrait par exemple collaborer pour relayer les paquets vers le point d'accès de l'opérateur, ces clients pouvant être indistinctement fixes ou faiblement mobiles. Les réseaux hybrides permettraient également de connecter un réseau domotique à internet, intégrant spontanément tout nouvel équipement. Nous nous focaliserons dans cette étude sur ces réseaux hybrides.

Les réseaux *ad hoc* demandent des solutions spécifiques, notamment à cause de la mobilité des nœuds et de l'utilisation du médium radio. Nous pensons que l'auto-organisation du réseau est nécessaire, préalablement à son exploitation par exemple pour le routage. L'auto-organisation est pour nous l'introduction d'une vue logique différente de la topologie physique. Une hiérarchie est introduite dans le réseau, permettant de distribuer plus efficacement les rôles. De plus, cette vue logique peut limiter l'impact de la mobilité en présentant une vue logique plus stable aux couches supérieures. Ainsi, les liens radio instables ne sont pas reportés et sont donc peu utilisés par les fonctions de haut-niveau telles que le routage. Toutes ces raisons expliquent que l'auto-organisation soit actuellement un domaine de recherche très étudié. Nous présenterons en section 3 une structure d'auto-organisation et un protocole de routage associé connectant un réseau *ad hoc* à internet de façon transparente.

Les protocoles pour réseaux *ad hoc* sont essentiellement évalués par simulation, permettant de reproduire un phénomène *in vitro* après modélisation. Les simulations sont intensivement utilisées pour leurs nombreux atouts :

– reproductibilité : la modélisation de l'environnement est fixe et déterministe. Il est possible de reproduire un scénario pour *rejouer* un phénomène particulier créant une anomalie de comportement,

– ré-utilisation : le code de simulation peut être partagé par la communauté et réutilisé facilement,

– facilité d'implémentation : un simulateur offre souvent une facilité accrue d'implémentation, grâce à un haut niveau d'abstraction et des primitives conçues pour la simulation. De plus, l'implémentation étant centralisée, la collecte des statistiques en est facilitée,

– rapidité des tests : un simulateur permet de tester de façon atomique l'influence d'un paramètre isolé de l'environnement,

– flexibilité du code : la couche protocolaire classique est modifiable à l'envi. Ainsi, un concepteur peut collecter des statistiques au niveau de la couche MAC et les faire partager aux couches supérieures,

– passage à l'échelle : un simulateur permet souvent de modéliser le comportement de dizaines à quelques milliers de nœuds différents,

– coût : de nombreux simulateurs sont *open-source*. De plus, il est seulement nécessaire d'investir dans une machine de calcul, même peu puissante. Une plate-forme expérimentale requiert au contraire l'achat d'un grand nombre de terminaux.

A la vue de telles remarques, nous pourrions conclure que les simulations constituent la panacée de l'évaluation de performances. Cependant, un simulateur se base sur une modélisation, *i.e.* une simplification du réel. En particulier, les phénomènes radio tels que les évanouissements, les réflexions ou l'hétérogénéité du milieu de propagation sont rarement pris en compte. Les performances peuvent donc varier de façon appréciable entre un environnement simulé et un environnent réel.

La contribution de cet article est double. Tout d'abord, nous décrivons une plate-forme générique de tests à travers la description détaillée d'une architecture logicielle et matérielle. Cette plate-forme est utilisable pour l'évaluation d'un protocole de routage quelconque pour réseaux hybrides : un ensemble de terminaux mobiles ou statiques coopèrent pour relayer l'informations vers la ou les terminaux connectés à internet. Nous ne nous intéressons donc pas dans cette étude aux réseaux maillés. Nous utilisons cette plate-forme pour évaluer le comportement d'un protocole d'auto-organisation et d'un protocole de routage pour réseaux hybrides. Les performances sont mesurées dans un environnement radio *indoor* en milieu réel, reflétant donc les performances réelles attendues d'une telle solution.

Nous allons dans une première section exposer un état de l'art sur les plates-formes de recherche existantes évaluant les performances d'un réseau *ad hoc* ou hybride. La section 3 présente succinctement les protocoles d'auto-organisation et de routage dont nous testons ici les performances. La section 4 détaille les propriétés requises pour pouvoir concevoir une plate-forme de tests flexible. Nous présenterons également une description détaillée de l'architecture logicielle et matérielle de la solution développée. La section 5 détaille les résultats des expérimentations, montrant l'efficacité des protocoles implémentés pour gérer un réseau hybride. Enfin, la section 6 expose les limites actuelles des plates-formes d'expérimentations et conclue en donnant quelques perspectives de ces travaux.

2. Etat de l'art

Le lecteur pourra se référer aux études (De *et al*., 2005), (Kiess & Mauve, 2007) présentant une synthèse très complète des expérimentations en sans-fil. Nous en dressons ici un rapide panorama.

2.1. *Approches pour concevoir une plate-forme expérimentale*

Une plate-forme de tests grandeur nature est compliquée à mettre en œuvre en termes par exemple de reproductibilité, d'isolation lors des tests et de complexité matérielle. Aussi, plusieurs approches ont été proposées pour simplifier le déploiement.

Dans (Maltz *et al*., 1999), un filtrage au niveau de la couche MAC afin de pouvoir contrôler statiquement les nœuds voisins est implémenté. Si tous les nœuds sont à une distance radio de 1 saut, le filtrage permet d'implémenter facilement une topologie quelconque. Durant l'implémentation, le concepteur peut tester une topologie particulière tout en maintenant les nœuds dans un espace limité. Naturellement, les performances réelles sont faussées. Ainsi, un tel outil ne doit être utilisé que dans la phase de conception et de validation. Suivant le même principe, Zhang & Li (2002) présentent MobiEmu : n nœuds du réseau *ad hoc* sont représentés par n PC. Par contre, les nœuds sont interconnectés *via* Ethernet, et un nœud central va donner à chacun des membres ses voisins *virtuels*. Kaba & Raichle (2001) proposent également un mécanisme permettant de contrôler facilement la topologie créée. Les auteurs proposent de déployer des dispositifs matériels permettant de câbler les cartes réseaux et donc de contrôler la topologie *via* des atténuateurs, séparateurs et combineurs de signal. Cependant, une telle approche nous semble biaisée : un tel environnement ne reproduit en aucun cas le comportement d'ondes radio ; les évanouissements ou réflexions sont même totalement absents. Ainsi, nous ne pensons pas qu'un tel environnement offre une plus-value par rapport à des simulations. Sanghani *et al*. (2003) proposent de réduire la portée radio en utilisant des câbles BNC pour antennes et en plaçant des atténuateurs pour réduire le signal et donc réduire la superficie de la plate-forme. Cette approche modifie donc là aussi la propagation radio. Raychaudhuri *et al*. (2005) décrivent le fonctionnement d'une plate-forme de tests en grille, déployée à l'université de Rutgers, pour tester le bon fonctionnement de protocoles pour réseaux sans-fil. Les liaisons radio sont *contrôlées* afin de proposer une reproductibilité : l'environnement ne permet donc que de valider un protocole, mais dans un environnement artificiel. Le projet Orbit Lab (D. Raychaudhuri *et al.*, 2009) fournit également un environnement radio réel en grille pour tester les protocoles, utilisant de multiples technologies sans-fil.

2.2. *Types de plate-forme*

Toh *et al*. (2000) proposent une plate-forme multisaut, mais testant seulement les performances d'une chaîne de 4 PC dans un environnement urbain. La chaîne est

une configuration fréquente en réseaux sans-fil, mais les auteurs ne testent que son comportement atomique, et non son intégration dans un réseau plus complexe. Par ailleurs, l'architecture logicielle n'est pas décrite.

Maltz *et al*. (1999) présentent un travail pionnier dans l'expérimental en sans-fil, dans une application de type réseaux véhiculaires. Les auteurs proposent d'évaluer expérimentalement les performances du protocole de routage pour réseax *ad hoc*, DSR. L'expérience comprend 2 nœuds fixes constituant les extrémités du réseau, et 5 nœuds mobiles se déplaçant selon une trajectoire circulaire à la même vitesse. Chaque nœud est équipé d'un récepteur GPS afin de retracer leur parcours après les expérimentations. Les auteurs mettent notamment en exergue les cassures des flux TCP lors des déplacements. Cependant, les pertes de paquets peuvent être élevées (>5 %), même lorsque les trames ne sont envoyées qu'aux voisins radio. (Barron *et al*., 2005) teste également les performances d'un réseau véhiculaire : des nœuds mobiles peuvent se connecter à internet *via* 12 nœuds statiques, disposés le long d'une route. Des nœuds mobiles peuvent éventuellement relayer les données d'autres nœuds mobiles si besoin est. Les auteurs n'ont testé que des flux UDP de longueur au plus de 2 sauts.

D'autres chercheurs ont proposé la création de véritables réseaux maillés de grande taille (> 30). Gray *et al*. (2004) comparent par exemple les performances de quatre protocoles de routage (APRL, AODV, ODMRP et STARA) dans un réseau maillé de 40 nœuds. Les auteurs comparent les performances *indoor* et *outdoor* de ces protocoles. AODV qui présente les meilleures performances en outdoor ne permet d'atteindre qu'un taux de livraison de 50 % : seulement un paquet sur deux arrive à destination. Ces performances très médiocres nous semblent étranges, les résulats que nous obtenons en milieu *indoor* étant bien meilleurs. Par ailleurs, ODMRP permet un taux de livraison de 60 %, mais ce protocole étant dédié au routage multicast, nous ne pensons pas qu'une comparaison avec les protocoles unicast soit judicieuse. Enfin, les expérimentations *indoor* consistent à reproduire articifiellement la topologie observée en environnement extérieur à l'aide d'une des méthodes décrites dans la section précédente (2.1). Ainsi, certains liens radio, bien qu'existant et générant des interférences, ne sont pas utilisés. Bicket *et al*. (2005) présentent MIT Roofnet, la plate-forme actuellement la plus aboutie selon nous. Trente sept nœuds sont déployés dans la ville, et implémentent un protocole de routage spécifique pour acheminer les informations vers les passerelles étant équipées d'une connexion filaire vers internet. Cette plate-forme présente la grande spécificité d'être opérationnelle et en production, *i.e.* des utilisateurs l'utilisent librement. Les auteurs présentent une évaluation fine de la qualité des liens radio de la plate-forme (débit TCP *versus* distance), de la distribution du degré, de la robustesse de la topologie.

2.3. *Conclusion*

Les performances actuelles des plates-formes sans-fil sont encore médiocres, même sans mobilité : beaucoup de paquets sont perdus, et les débits atteignables sont faibles en multisaut. Il reste donc encore un travail important à fournir avant de créer un

réseau sans-fil multisaut adapté aux besoins multimédias des utilisateurs nomades. Nous nous focalisons ici sur le déploiement d'un petit réseau sans-fil multisaut mais fixe (8 nœuds sans-fil avec une seule interface). Pour vérifier que le protocole SOMoM fonctionne bien dans un réseau hybride, nous avons également testé les performances lorsqu'un nœud mobile est introduit dans le réseau. Nous souhaitons dans un avenir proche déployer plus de nœuds, à la fois en intérieur et extérieur, afin de pouvoir tester le passage à l'échelle, et constituer un réel réseau hybride d'expérimentations, en suivant le modèle maillé de MIT Roofnet déployé sur toute la ville (Bicket *et al.*, 2005). La plate-forme présentée ici constitue donc pour nous une première étape dans l'expérimentation des réseaux hybrides sans-fil.

3. Protocoles évalués sur la plate-forme

Nous avons déployé une couverture sans-fil multisaut en IEEE 802.11 dans le laboratoire, basée sur le protocole SOMoM (Theoleyre & Valois, 2005). Chaque terminal peut se connecter librement à internet sous la condition qu'il exécute un démon permettant de gérer les accès au réseau de façon transparente pour l'utilisateur. SOMoM fonctionne en tirant parti de la structure d'auto-organisation proposée dans (Theoleyre & Valois, 2008). Nous allons ici présenter une très courte description de ces protocoles car elle nous semble nécessaire à la compréhension de cet article.

3.1. *Structure d'auto-organisation*

Theoleyre & Valois (2008) présentent une structure d'auto-organisation : elle permet d'organiser logiquement un réseau *ad hoc*, créant une hiérarchie dans le réseau. Cette vue logique (figure 1) est plus stable que la topologie radio. En outre, la hiérarchie introduite permet de simplifier les fonctions de haut niveau telles que le routage. Theoleyre & Valois (2008) détaillent les algorithmes permettant de construire mais également de maintenir cette structure en forme d'arbre. Dans un réseau hybride, le nœud connecté à internet sert de racine de la dorsale. Dans un premier temps, chaque nœud initie une découverte de voisinage : il envoie périodiquement des `hellos` (figure 1). Ensuite, une élection distribuée permet d'élire les membres de la dorsale. L'ensemble de ces membres (les *dominants*) forment un ensemble connexe, et chaque nœud *normal* du réseau (un *dominé*) est voisin d'au moins un nœud *dominant*. La topologie construite est orientée : chaque dominé est rattaché à un dominant et un dominant maintient lui-même un père à un saut de lui dans la dorsale. Nous avons également proposé des procédures distribuées permettant de maintenir cette dorsale en cas de changement de topologie.

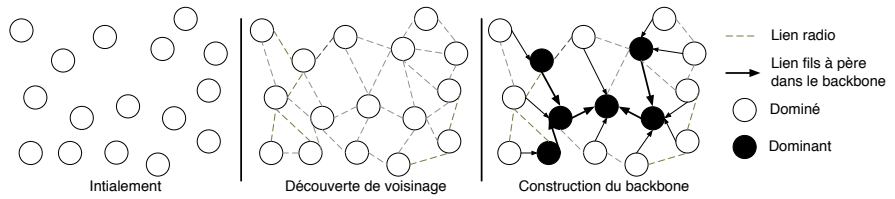

Figure 1. *Structure d'auto-organisation pour réseaux hybrides*

3.2. *Protocole de connexion sans-fil multisaut* SOMoM

Self-Organized Mobility Management Protocol (SOMoM) (Theoleyre & Valois, 2005) permet de tirer parti de la stabilité et de la hiérarchie introduites par cette structure d'auto-organisation pour router les informations dans le réseau hybride. Le protocole présente notamment les caractéristiques suivantes :

– dans le sens *montant* (vers internet), le protocole maintenant la structure d'auto-organisation permet de créer gratuitement une route par défaut vers le point d'accès, *via* les liens de parenté dans l'arbre. Ainsi, lorsqu'aucune route n'est présente dans la table de routage, le paquet est envoyé sur cette route par défaut, se rapprochant d'un saut de la racine de la dorsale (figure 2),

– lorsqu'un nœud reçoit un paquet, il enregistre le nœud par lequel le paquet lui arrive afin d'ajouter la route vers la source correspondante dans sa table de routage. Ainsi, si un paquet de réponse arrive plus tard sur la route inverse, chaque nœud intermédiaire possédera déjà une route vers la source initiale,

– si une connexion est initiée par un hôte sur internet, aucune route n'est présente. Ainsi, le point d'accès va stocker le paquet dans une file d'attente et déclenche une découverte de route, envoyée dans la dorsale. Seuls les dominants de la dorsale relaient un paquet de découverte de route, minimisant ainsi le trafic de contrôle. Lorsque le paquet de requête est reçu par la destination, elle renvoie une réponse de route. Cette réponse est relayée *via* la route par défaut, créant ainsi une route inverse dans chaque nœud intermédiaire, comme pour un paquet de données habituel. Souvent, une seule découverte de route est nécessaire pour l'intégralité d'une connexion : les autres paquets suivent automatiquement la route mise en cache,

– SOMoM propose des procédures pour mettre à jour les tables de routage lorsque des changements surviennent dans la structure d'auto-organisation. En particulier, lorsque la dorsale est cassée puis réparée, une route peut changer puisqu'elle suit les liens fils→père de la dorsale. Des procédures permettent donc de supprimer automatiquement les routes obsolètes lorsqu'une reconnexion de la dorsale survient.

En conclusion, SOMoM permet d'interconnecter de façon transparente un réseau sans-fil multisaut à internet. Les performances en simulation étant intéressantes, nous proposons ici d'étudier son comportement *in vivo*.

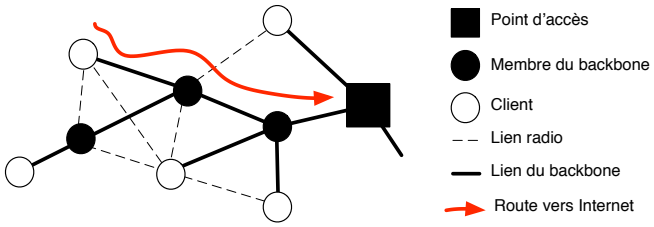

Figure 2. *Comportement général de* SOMoM

4. Implémentation et déploiement

Nous allons décrire ici l'architecture matérielle et logicielle de l'environnement de tests que nous avons déployé.

4.1. *Propriétés requises*

Nous nous sommes efforcés de suivre les propriétés suivantes, vitales selon nous pour constituer une plate-forme de tests adéquate :

– la plate-forme doit être assez flexible pour simuler un protocole quelconque. En particulier, l'implémentation doit être découpée en briques élémentaires et réutilisables. Par exemple, la découverte de voisinage est une fonction récurrente dans les réseaux hybrides et peut être détachée des autres parties du code. De plus, l'implémentation doit être la plus indépendante possible d'un système d'exploitation ou d'un noyau pour pouvoir évoluer plus facilement,

– les résultats lors d'une évaluation de performances doivent être le plus possible fidèles à la réalité. Ainsi, des tests doivent être lancés en nombre, en tenant compte de paramètres n'apparaissant souvent pas dans les simulations (personnes présentes ou portes fermées modifiant la propagation radio, liens radio intermittents…) Autant que possible, les tests doivent être automatisés afin de maximiser la reproductibilité et la vérification des performances réelles par un tiers dans d'autres conditions,

– l'évaluation de performances doit s'attacher tant à des tests atomiques (temps de convergence après la mobilité d'un nœud) qu'à des métriques globales quantitatives (délai d'acheminement d'un paquet) ou qualitatives (ressenti d'un utilisateur pour la navigation sur le web),

– bien qu'elle soit complexe à évaluer, la mobilité doit être introduite. Elle peut simplement consister à éteindre/allumer des nœuds statiques ou observer l'impact d'un nœud mobile,

– naturellement, le code d'implémentation doit être mis à disposition de la communauté à des fins de vérification, d'amélioration et de réutilisation.

4.2. *Architecture logicielle*

Nous avons choisi de déployer un système Linux sur l'ensemble des nœuds actuels de la plate-forme. Ce système d'exploitation, ouvert, est largement utilisé par la communauté, le code pouvant être réutilisé sur d'autres plates-formes. Enfin, de nombreux outils existent pour la mesure des performances réseau (e.g. débit des flux TCP ou UDP). Nous avons implémenté un démon permettant de construire la dorsale virtuelle décrite dans la section 3.1, ainsi que le protocole de routage décrit dans la section 3.2. Ce démon est fonctionnel, la topologie virtuelle construite de façon valide, et le protocole de routage tirant parti de la dorsale virtuelle fonctionne parfaitement. Certaines fonctions comme la mise à jour des systèmes sont réalisées *via* ces liaisons radio multisaut. Nous allons dans la suite de cette section détailler certains points de l'implémentation.

4.2.1. *Niveau d'exécution*

La philosophie de Linux est d'implémenter en mode noyau le minimum de fonctionnalités. Cette propriété est peu suivie par les concepteurs de protocoles de routage pour les réseaux *ad hoc* et nuit à l'évolution du code et sa réutilisation. Ainsi, Ad Hoc Support Library (ASL) (Kawadia *et al*., 2003) permet à un démon d'être implémenté dans l'espace utilisateur mais contient lui-même des modules noyau, par exemple pour la gestion d'une table de routage modifiée. Chakeres & Belding-Royer (2004) présentent trois méthodes pour coder un protocole de routage :

– utilisation de Netfilter (Welte & et al., 2005). Cette fonctionnalité du noyau permet d'activer des filtres sur les paquets IP et de passer les paquets à d'autres programmes pendant la traversée de la couche IP. Ces règles peuvent éventuellement rentrer en conflit avec les règles ajoutées par l'utilisateur,

– le démon peut être codé dans l'espace noyau tel le démon de routage Kernel-AODV (Klein-Berndt & et.al, 2004),

– le programme écoute les trames passant au niveau MAC : toute requête ARP est capturée et déclenche une découverte de route. Cette méthode requiert donc d'interférer avec le fonctionnement d'ARP.

Nous avons choisi une troisième solution, exécutant la totalité du code en espace utilisateur. Il a été testé sur les noyaux 2.6.14 et 2.6.12. Nous détaillons les échanges de paquets en détails un peu plus loin.

4.2.2. *Programmation multithread*

Le démon nécessite de surveiller plusieurs tables (table de voisinage, table de routage…) et d'éliminer les données obsolètes. Bien que la maintenance soit événementielle, il n'existe aucun moyen pour obtenir en temps réel des informations exactes sur le réseau. Un nœud ne peut qu'implémenter des temporisateurs pour les entrées obsolètes, étant donné que le médium radio n'est fiable ni pour les transmissions en *broadcast*, ni pour celles en *unicast*. De même, les procédures de maintenance de

la structure virtuelle se déclenchent sur certains évènements eux-mêmes découlant de temporisateurs[1]. Nous avons donc choisi de développer le démon en multithread, chaque thread possédant une action spécifique, avec l'utilisation de sémaphores pour les ressources partagées. Une approche multi-processus a été écartée à cause d'un besoin plus important en mémoire.

4.2.3. *Adressage et configuration*

Dans la première étape de ce déploiement, nous avons choisi d'assigner statiquement les adresses aux nœuds du réseau. Un nœud se voit attribué une adresse IP avec un masque de 32 bits (le *masque réseau*). Ainsi, une route dans la table de routage du noyau est spécifique à un nœud. Cependant, toutes les adresses des nœuds *ad hoc* sont dans un même préfixe IP logique, partageant ce que nous avons appelé le *masque somom*. Par exemple, un nœud reçoit l'adresse IP `192.168.1.15` avec le *masque réseau* `255.255.255.255` et le *masque somom* `255.255.255.0`. Ainsi, toutes les adresses *ad hoc* seront comprises dans l'intervalle `192.168.1.1-254`.

Le *masque somom* trouve son utilité avec le point d'accès : il doit distinguer les destinations dans l'aire *ad hoc* de celles présentes dans internet. Lorsque la destination est connue, une route avec un masque de 32 bits est déjà présente dans la table de routage : le paquet est donc directement envoyé. Si au contraire le point d'accès ne possède encore aucune route, il vérifie que l'adresse IP de la destination appartient au sous-réseau de somom (obtenu à partir de l'adresse IP et du *masque somom*).

4.2.4. *Table de routage*

La table de routage classique du noyau Linux a été conçue pour les réseaux filaires : les routes présentent une grande stabilité et les changements constituent des exceptions. Au contraire, dans un réseau *ad hoc*, la mise à jour des tables de routage est fréquente : les nœuds sont mobiles, nécessitant de mettre à jour continuellement leur vue de la topologie. Cependant, nous avons préféré ne pas modifier la table du routage du noyau pour des raisons de portabilité. Ainsi, un thread est chargé de maintenir une table de routage interne au processus, gérer les *timeouts*, et synchroniser lui-même la table de routage du noyau.

4.2.5. *Pile protocolaire*

Le modèle OSI stipule que les couches protocolaires doivent être indépendantes afin de pouvoir favoriser l'interchangeabilité entre plusieurs couches. L'indépendance des couches permet une plus grande flexibilité, mais au prix d'une baisse de performances : aucune information n'est partagée. Il existe donc un compromis entre partage d'informations augmentant significativement les performances et indépendance

1. La non réception de certains paquets de maintenance constitue par exemple un évènement déclencheur d'une procédure de reconnexion de l'auto-organisation. Nous voyons bien qu'une telle occurrence constitue un *non évènement*, qu'il faut donc déclencher sur un temporisateur réarmé à chaque réception de paquets particulier.

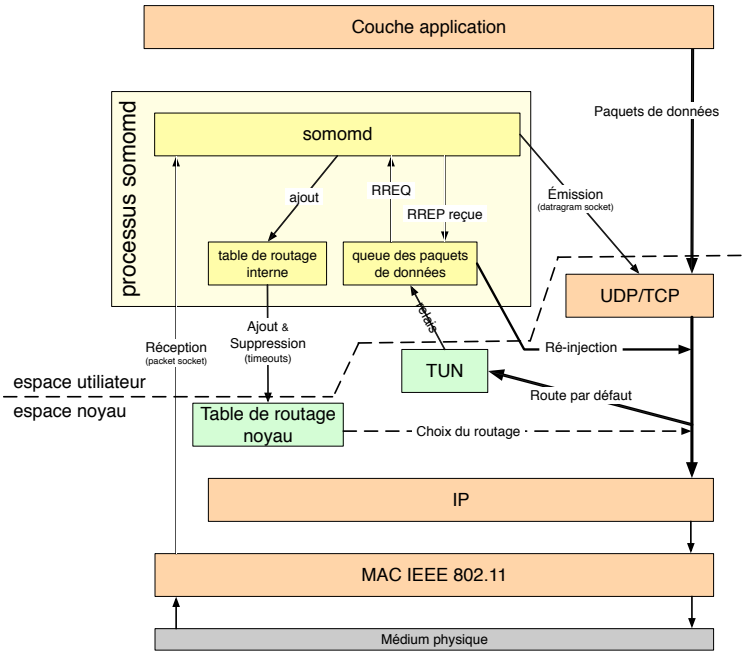

Figure 3. *Architecture logicielle générale de* SOMoM *pour le point d'accès*

des couches. Ainsi, certains protocoles de routage, par exemple DSR, présupposent l'existence d'une API de notification de livraison de paquets par la couche MAC. Une telle fonctionnalité permet de réduire les retransmissions et autorise les reconstructions locales de routes.

Nous avons donc décidé de *casser* la pile protocolaire, approche souvent dénommée sous le terme de *cross-layer*. L'architecture retenue est représentée sur la figure 3. Le démon s'exécute au niveau logique au-dessus de la couche réseau, et emprunte un port UDP pour l'envoi des paquets de contrôle. Par contre, le protocole ayant besoin d'informations sur tous les paquets transitant *via* la couche IP, le démon implémente une *packet socket*. Cette fonctionnalité offerte par Linux permet de capturer l'ensemble des trames venant de la couche MAC, avec leurs en-têtes de niveau 2 et 3. Le démon extrait les paquets IP, et les informations qui l'intéressent par exemple pour mettre à jour sa table de routage. Si le paquet est destiné au port UDP utilisé par le démon, il est ensuite directement transmis aux threads chargés du routage et de la structure virtuelle. Ainsi, lorsqu'un paquet est relayé par un nœud, SOMoM peut mettre à jour ses informations à la volée.

En particulier, dans SOMoM, un nœud peut ajouter une entrée dans sa table de routage pour la source de chaque paquet reçu lorsqu'elle appartient au *sous-réseau*

somom (calculé à partir de l'adresse IP et du *masque somom*). Ainsi, une route inverse est apprise gratuitement à chaque fois qu'un paquet est relayé par un nœud.

4.2.6. *Prise en compte du réactif*

Le comportement réactif n'est pas intégrable de façon triviale dans un démon Linux en mode utilisateur. En effet, le noyau devrait relayer tous les paquets sans route vers le processus réactif afin qu'il initie une découverte de route. Nous avons choisi d'utiliser la fonctionnalité TUN/TAP de Linux. Une interface virtuelle du type `/dev/tunX` est créée derrière laquelle un processus utilisateur peut s'enregistrer afin que le noyau lui relaie les paquets envoyés *via* cette interface. En créant au sein de la passerelle une route pour le sous-réseau somom (correspondant à la bulle *ad hoc*) menant vers cette interface virtuelle, le noyau relaie de façon transparente tous les paquets sans route vers le démon SOMOM. De plus, lorsque plusieurs routes possibles sont présentes, Linux choisit, comme la rfc 1518 (Rekhter & Li, 1993) le stipule, la route avec le préfixe le plus long. Ainsi, les routes spécifiques, avec un préfixe de 32 bits sont choisies en priorité, la route menant vers l'interface *tun* n'étant choisie qu'en dernier recours.

Le démon récupère les paquets, les stocke dans une file d'attente temporaire en espace utilisateur, dans un segment mémoire alloué à SOMOM. Puis il génère une découverte de route. Si aucune réponse de route n'arrive au bout d'un temporisateur (une seconde dans notre cas), le démon renvoie une nouvelle demande de route. Dans notre implémentation, si un paquet n'obtient aucune réponse de route, il est supprimé de la file d'attente au bout de cinq secondes. Si la passerelle reçoit une réponse de route, elle enregistre la route dans le noyau, extrait dans sa queue locale les paquets destinés à cette entrée, et réinjecte normalement les paquets dans la couche IP. La nouvelle entrée dans la table de routage sera utilisée pour router les paquets vers la bonne destination.

Parallèlement, chaque client possède une route par défaut menant vers son père dans la dorsale. Si une route spécifique avec un préfixe de 32 bits existe, elle sera utilisée, sinon, le paquet remontera dans la dorsale jusqu'à atteindre la passerelle. Celle-ci sera ensuite chargée de délivrer le paquet vers internet après avoir fait de la translation d'adresse. Éventuellement, si la destination est en fait présente au sein de la bulle *ad hoc*, la route *via* l'interface *tun* sera empruntée.

4.2.7. *Génération de traces*

Le démon de routage s'interface avec `syslog` afin de garder une trace de tous les événements survenus, et donc de collecter des statistiques (e.g. trafic de contrôle, changement de voisinage, paquets de données relayés, pertes des paquets).

4.3. *Équipements des nœuds*

Les terminaux utilisés sont des mini-PC silencieux. Cependant, les capacités sont assez puissantes pour stocker les *logs*, déployer un système d'exploitation Linux com-

plet, installer un analyseur réseau... Nous avons choisi d'utiliser dans un premier temps des nœuds statiques pour des raisons de facilité de tests et de configuration. Cependant, la topologie radio peut changer sous l'influence d'obstacles tels que des personnes dans le couloir ou une porte fermée. De plus, nous avons étudié l'impact d'ajout et de suppression de certains de ces nœuds statiques. Dans un deuxième temps, un nœud mobile a été introduit.

Les nœuds du réseau possèdent tous une interface radio IEEE 802.11 a/b/g. Cependant, pour que tous les clients mobiles puissent se connecter au réseau hybride, nous avons utilisé la bande de fréquences des 2,4 GHz, plus largement déployée. Naturellement, des tests de fonctionnement ont également été réalisés dans la bande des 5 GHz pour valider le comportement du protocole. IEEE 802.11 présente une portée radio différente en unicast et broadcast selon le débit utilisé. Ainsi, un paquet `hello` envoyé en broadcast peut être reçu correctement sur un lien radio sans qu'aucun paquet ne puisse être reçu en unicast. De plus, le multidébit présente des anomalies de performances (Heusse *et al.*, 2003). Ainsi, nous avons bloqué les cartes à un débit de 1 Mbps. Afin de simplifier l'administration, tous les mini-PC possèdent une carte réseau filaire additionnelle. Le trafic de management ne vient donc pas perturber les expérimentations.

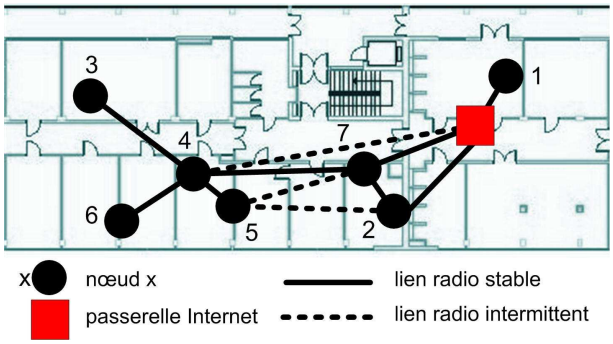

Figure 4. *Topologie du réseau hybride déployé*

4.4. *Plate-forme*

La plate-forme expérimentale est constituée de 8 mini-PC déployés dans le laboratoire, en environnement *indoor*. La topologie représente un compromis afin de tester tant des transmissions sur une longue route que la redondance du réseau en cas de défaillance d'un nœud (figure 4). Un nœud (le carré sur la figure 4) agit en tant que passerelle pour internet implémentant les fonctions de pare-feu, NAT... Certains des liens radio présentent une grande stabilité (représentés en trait plein) : le débit est peu variable au cours du temps, le rapport signal sur bruit (SNR) restant stable et acceptable. Par contre, d'autres liens radio (représentés en pointillés) sont plus fluctuants :

ils laissent périodiquement passer des paquets, puis rapidement stoppent tout envoi fiable.

5. Tests de performances dans un environnement *indoor*

La plate-forme est entièrement opérationnelle et permet de mettre en place de façon transparente une connexion internet multisaut. Nous avons tout d'abord étudié l'impact d'un changement de topologie sur l'auto-organisation puis les performances au niveau du routage IP (débit TCP & UDP, délai de bout en bout, délai de découverte de route). Le code source du démon d'auto-organisation et de routage est disponible sur la plate-forme *sourceforge* (Theoleyre & Valois, 2006). Ce démon est actuellement fourni avec une installation automatique sur Debian. Le lancement du démon créera spontanément un réseau hybride, avec une connexion internet si au moins un des nœuds a spécifié dans son fichier de configuration une interface filaire vers internet. Afin d'avoir des résultats significatifs, nous avons répété 10 fois chaque expérience et reporté les moyennes. Nous avons représenté sur les graphes les intervalles de confiance à 95 %.

5.1. *Auto-organisation*

Nous avons dans un premier temps évalué l'impact d'un changement de topologie sur l'auto-organisation : les algorithmes doivent rapidement mettre à jour leurs informations et converger vers un état légal. Nous avons notamment mesuré l'impact de l'ajout et de la suppression d'un nœud dans le réseau. Ce comportement atomique reflète la robustesse à la mobilité.

Nous avons distingué plusieurs scénarios qui, selon nous, reflètent le comportement général du protocole. Le premier cas est l'apparition d'un nœud dans le réseau. L'algorithme converge rapidement car au bout d'une seconde, le nouveau nœud est intégré dans l'auto-organisation, prêt à être utilisé. Nous proposons donc un deuxième scénario (figure 5) dans lequel nous ajoutons simultanément 2 nœuds (tableau 1). Au total, 3,2 secondes sont nécessaires à l'auto-organisation pour se reconfigurer. Au final, 0,8 secondes de plus sont nécessaires à la resynchronisation des informations pour la dorsale (identité de la passerelle internet, etc.).

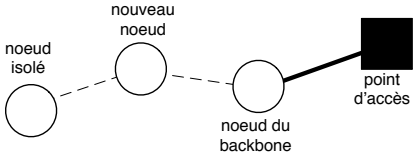

Figure 5. *Scénario d'apparition de deux nouveaux terminaux*

Étape	Temps de convergence (en s)
État	2,6
Père de la dorsale valide	3,2
Mode non dégradé	4,0

Tableau 1. *Temps de convergence en cas d'ajout de deux nœuds*

Nous avons ensuite mesuré l'impact de la disparition d'un nœud intermédiaire du réseau (figure 6) : la dorsale de l'auto-organisation se retrouve donc déconnectée. Nous avons mesuré les temps nécessaires à chaque phase (tableau 2). 4 secondes sont nécessaires pour que les nœuds détectent tous une disparition : ce temps correspond au *timeout* d'un voisin, lorsqu'aucun `hello` n'est reçu. La phase de reconnexion dure 1,2 secondes supplémentaires (paquets de reconnexion de la dorsale, etc). Enfin, le fonctionnement *normal* de l'auto-organisation est effectif moins d'une seconde après. Au final, 7 secondes seulement sont nécessaires pour une reconnexion complète. Naturellement, ce temps de convergence peut être largement réduit avec une détection plus rapide de disparition (trop d'erreurs lors de l'envoi de paquets de données en unicast, `hellos` très rapprochés, détection au niveau MAC, etc).

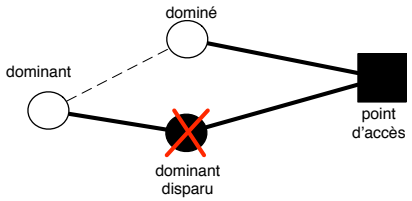

Figure 6. *Scénario de disparition d'un terminal*

Étape	Temps de convergence (en s)
Détection de la disparition	4,0
Père dans la dorsale valide	6,2
Mode non dégradé	7

Tableau 2. *Temps de convergence en cas de disparition d'un nœud*

5.2. *Connectivité multisaut*

5.2.1. *Ping*

L'outil *ping* nous a permis d'étudier le comportement général de SOMOM. Nous avons dans un premier temps mesuré le délai suivant la longueur des paquets des *ping*

envoyés (figure 7). Lorsque la taille des paquets augmente, le délai de bout en bout augmente également : la transmission *via* le médium radio requiert plus de temps puisque la bande passante est limitée. De plus, l'approche *store-and-forward* augmente cet effet : le paquet doit être reçu dans son intégralité avant d'être relayé. Par ailleurs, nous pouvons voir que les délais moyens, minimum et maximum sont assez proches : la gigue est très réduite. Naturellement, un tel délai augmente dans un réseau plus chargé, des collisions pouvant se produire entre les différentes transmissions. Nous avons donc mesuré le délai d'un flux seul afin d'isoler l'influence des différents paramètres.

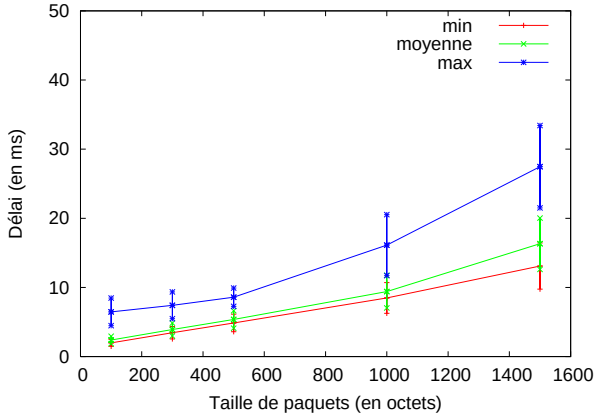

Figure 7. *Délai de bout en bout suivant la taille des paquets*

Puique la topologie reste dans cette partie inchangée, nous avons ensuite fixé des routes statiques sur la plate-forme puis nous avons comparé les performances générales avec celles de SOMoM, qui calcule dynamiquement les routes et les met à jour en fonction de la topologie réelle (tableau 3). Nous pouvons vérifier que, quel que soit le terminal en communication, les délais sont similaires. SOMoM, malgré la découverte dynamique de route, permet d'obtenir des délais semblables à une solution statique, tout en présentant une flexibilité et une adaptation maximale aux changements de topologie. On peut même remarquer que dans certains cas, SOMoM semble plus efficace : le protocole s'adapte à la qualité réelle du lien radio, passant éventuellement *via* une autre route lorsque le lien devient faible. Nous pouvons remarquer que le nœud 5 ne possède que des liens radio instables, impactant de façon drastique les performances : les délais sont élevés, et la gigue est importante. Enfin, certains délais très élevés (e.g. 484 ms pour des paquets de 1500 octets avec SOMoM de la source 1) sont dus à des interférences passagères qui entraînent l'apparition de pertes de paquets pour certains liens radio et nécessitent donc une reconfiguration du routage (personne se déplaçant en cours d'expérimentation).

Taille paquet	Proto	Noeud source						
		5	2	7	3	6	4	1
100	SOMoM	23 (18)	4 (0,2)	5 (4)	5 (0,5)	15 (32)	15 (9)	9 (17)
	statique	46 (74)	4 (20)	4 (0,8)	25 (12)	52 (0,5)	10 (22)	9 (0,7)
1500	SOMoM	485 (329)	29 (1)	35 (3)	34 (2)	132 (50)	118 (13)	144 (484)
	statique	649 (229)	29 (0,5)	34 (2)	64 (18)	119 (8)	121 (19)	82 (3)

Tableau 3. *Délai de bout en bout en milli-secondes mesuré* via *l'outil* ping*, les valeurs sont présentées sous la forme "moyenne (écart-type)"*

5.2.2. *Flux TCP*

Puisque les flux TCP représentent une part importante du trafic internet, nous avons mesuré avec *iperf* le débit atteignable pour des flux TCP de 8 secondes (figure 8). Nous pouvons voir que le débit augmente lorsque la taille des paquets augmente. En effet, certaines trames de contrôle (acquittements, backoffs, etc.) doivent être transmises dans IEEE 802.11 avant toute trame, quelle que soit sa taille. Ainsi, un paquet de taille plus importante permet de réduire le ratio de la bande passante réservée au contrôle. Nous pouvons de plus distinguer deux groupes de flots : les flots de 1 saut permettent d'obtenir le meilleur débit tandis que les autres flux, puisqu'ils nécessitent d'être relayés, présentent un débit moindre. Par ailleurs, nous pouvons voir que le débit TCP atteint un maximum sur les liens multisaut à cause des collisions et retransmissions. Au contraire, pour les liaisons radio simple saut, une taille de paquet très grande ne semble pas une limite.

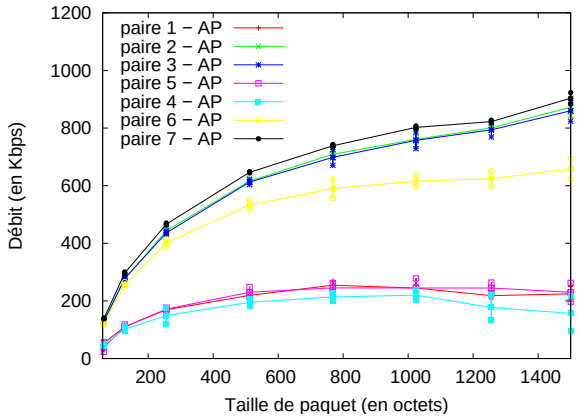

Figure 8. *Débit d'un flux TCP suivant la taille du paquet*

Nous avons mesuré le débit TCP atteignable durant un temps de 8 secondes sur toutes paires source/destination du réseau. Chaque mesure est reportée par un point sur la figure 9. Nous pouvons voir que les liens radio présentent une hétérogénéité élevée en termes de débit : tandis qu'un lien radio présente souvent un débit de 700 kbps,

il peut descendre jusqu'à 100 kbps à cause par exemple d'obstacles ou interférences temporaires. Cependant, 80 % des liens présentent un débit compris entre 500 kbps et 650 Kbps, les valeurs beaucoup plus basses représentant des exceptions (les liens à moins de 200 Kbps représentant moins de 8 % des cas). Par ailleurs, les débits sur les flux multisaut sont inférieurs, ce qui est une conséquence logique puisqu'un paquet occupe dans ce cas-là plusieurs fois le médium radio. Par contre, les débits en 2, 3 ou 4 sauts sont assez comparables. Le débit passe donc bien à l'échelle vis-à-vis de la longueur des routes. En d'autres termes, SOMoM découvre des routes stables, ne cassant pas de façon intempestive au cours du temps.

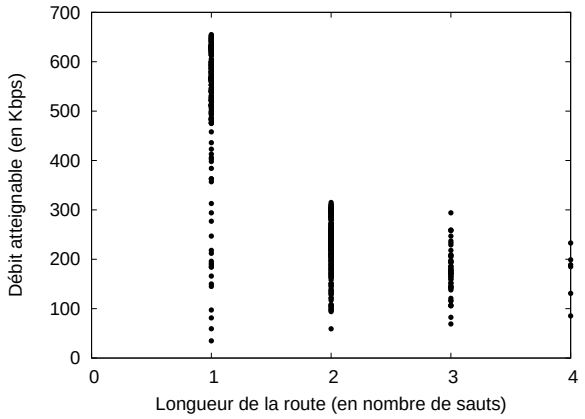

Figure 9. *Débit d'un flux TCP suivant la longueur de la route*

5.2.3. *Flux UDP*

Nous avons également mesuré le débit des flux UDP (figure 10). Puisqu'UDP est non fiable, nous avons supposé qu'un flux est *atteignable* si au moins 95 % des paquets émis sont bien reçus par la destination.

Les débits de UDP et TCP sont similaires. Nous pouvons distinguer quelques liens radio qui semblent présenter un débit UDP moindre. Ce sont essentiellement des liens radio connaissant de fortes pertes de paquets. UDP ne proposant pas de retransmission, le débit atteint semble donc plus faible. Par ailleurs, nous pouvons observer qu'une taille optimale de paquet existe. En effet, si le paquet est trop petit, le trafic de contrôle est trop élevé, s'il est trop long, la probabilité de collision augmente. La taille optimale est d'environ 1 300 octets mais dépend de la topologie exacte, les nœuds cachés pouvant par exemple créer plus de collisions et favoriser les paquets courts. UDP semble donc moins bien gérer les collisions que TCP.

Nous avons reporté sur la figure 11 les débits obtenus en UDP et TCP pour le flux 4 vers 7. Nous pouvons remarquer que le flux UDP est moins stable qu'avec TCP pour des tailles de paquets élevées : les collisions sont importantes, aboutissant à de nombreuses pertes de trames. Au contraire, TCP semble moins bien réagir qu'UDP lorsque

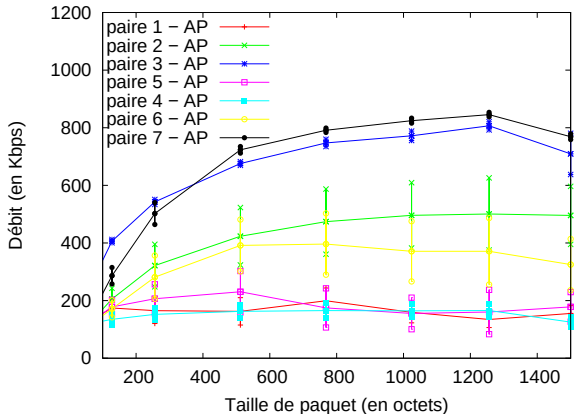

Figure 10. *Débit d'un flux UDP suivant la taille du paquet*

de petites trames sont envoyées : TCP arrive mal à adapter sa fenêtre de congestion à cause du délai de bout en bout assez élevé, occasionné par les interférences à l'intérieur d'un même flot. UDP, étant plus agressif, arrive à acheminer plus de paquets.

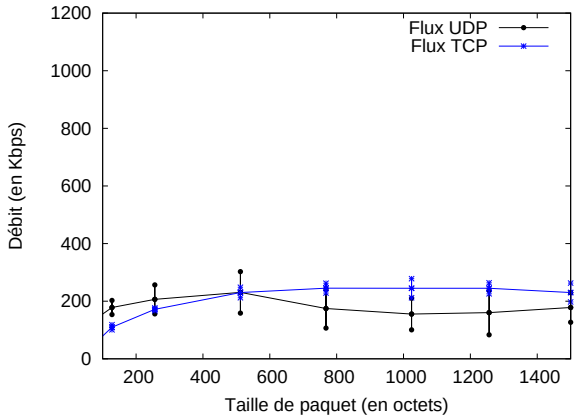

Figure 11. *Comparaison du débit obtenu avec UDP et TCP suivant la taille du paquet pour le flux 3→7*

5.2.4. *Découverte de routes*

Nous avons ensuite mesuré le délai nécessaire à la découverte d'une route (tableau 4). Pour une route de 3 sauts, 800 millisecondes sont en moyenne nécessaires pour un *ping* (découverte de route, retransmissions potentielles, réception de la réponse de route, aller/retour du *ping*). Cependant, une découverte de route est rare : lorsqu'un nœud initie lui-même la communication, la route inverse est gratuitement

créée dans les caches de chaque nœud intermédiaire. De plus, ce délai est nécessaire seulement pour le premier paquet d'un flux. Ainsi, nous pensons qu'un tel délai reste largement acceptable pour un réseau hybride classique.

Type	nœud 3	nœud 5	nœud 6
Délai	779	894	779
Écart type	246	384	352

Tableau 4. *Délai aller/retour (en ms) quand une découverte de route est nécessaire*

5.2.5. *Nœud mobile*

Nous avons ensuite introduit un nœud mobile dans le réseau hybride. Nous l'avons déplacé d'une extrémité du réseau à l'autre, vers le point d'accès, à environ 3 km/h (cf. carte sur la figure 4). Nous avons mesuré le débit TCP au cours du déplacement du nœud. L'expérience a été renouvelée plusieurs fois, la figure 12 représentant un des résultats types obtenus. Bien que les débits obtenus puissent varier selon les expériences (à cause par exemple d'une personne présente dans le couloir), la forme de la courbe et les délais de convergence ne variaient que très peu. Au tout début des mesures, le nœud se trouve à une extrémité du réseau et possède un lien radio faible, ce qui explique le débit fluctuant. Lorsque le nœud s'arrête, le débit devient plus stable. Lorsque le nœud change de père, le débit chute logiquement : des reconfigurations au niveau du routage sont nécessaires, perturbant TCP. Des mécanismes de *soft-handoffs* au sein de l'auto-organisation permettraient sans doute d'améliorer les performances : un nœud pourrait par exemple maintenir deux pères lors de son déplacement, diminuant le temps de reconfiguration lorsqu'un des pères se trouve hors de portée de communication. Enfin, lorsque le mobile se trouve proche du point d'accès, le débit devient élevé car il n'est plus relayé et ne consomme donc que peu de ressources radio.

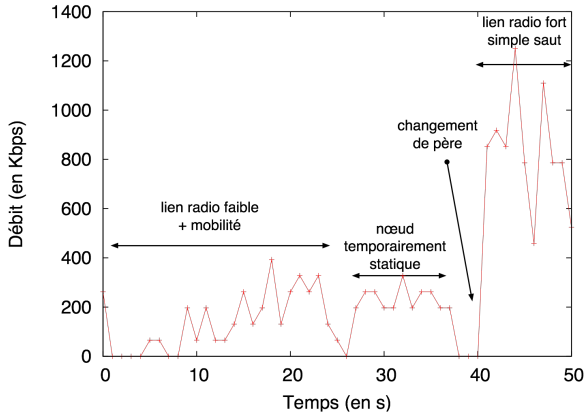

Figure 12. *Débit TCP au cours du déplacement d'un nœud mobile*

5.3. *Vision utilisateur*

Finalement, nous avons réalisé des tests qualitatifs d'utilisation réelle du réseau hybride. Le réseau multisaut remplit bien son rôle. Lors de l'utilisation du web (flux HTTP), la navigation est fluide : le trafic est faible et TCP permet de gérer de façon transparente les retransmissions. Au contraire, des flux FTP chargent beaucoup le réseau sans-fil, créant des collisions, et perturbant le trafic de contrôle. En conséquence, des nœuds considèrent de façon erronée que certaines routes sont cassées (par exemple lorsque des `hellos` subissent beaucoup de collisions). Ainsi, les flux de données se retrouvent coupés. La perte de paquets de contrôle peut donc avoir des conséquences importantes, créant un trafic en rafale. Nous avons un fonctionnement périodique : le trafic est dans un premier temps élevé, beaucoup de paquets de contrôle subissent des collisions. Ces collisions créent des cassures de routes, stoppant les flux de données. Enfin, les routes sont reconstruites et les flux repassent. Selon nous, il est vital que dans l'avenir, IEEE 802.11 permette d'offrir une certaine qualité de service pour le trafic de contrôle, nécessaire au bon fonctionnement du réseau.

6. Une plate-forme de tests : retour sur expérience

Cette plate-forme expérimentale constitue une première étape dans l'évaluation expérimentale des performances d'un protocole de routage et d'auto-organisation pour réseaux hybrides. Cependant, il serait intéressant de déployer plus de nœuds et de trouver une méthode flexible pour la gestion des nœuds mobiles. Par ailleurs, des nœuds embarqués à plus faible capacité doivent également pouvoir être installés afin de mesurer les contraintes en termes de CPU, de mémoire ou d'énergie. Cette plate-forme nous a permis par ailleurs de mettre en exergue les conclusions suivantes :

– certains liens radio sont faibles et instables. Une porte fermée suffit à changer la topologie radio. De même, la portée radio n'est pas binaire : certains liens longs présentent une fiabilité faible mais non nulle (Stojmenovic *et al.*, 2005). Si un `hello` est correctement reçu, le nœud considère que le lien radio est utilisable, alors que ses pertes sont en réalité importantes. Nous avons donc modifié SOMoM : un lien radio est considéré valide si plus de 2 paquets `hello` consécutifs sont reçus. Cependant, une métrique d'efficacité de lien radio telle que celle décrite par De Couto *et al.* (2003) devrait être implémentée,

– bien que cette plate-forme expérimentale soit constituée de nœuds homogènes, des liens unidirectionnels peuvent apparaître, à cause d'antennes non omnidirectionnelles, ou de puissances d'émission différentes. Le protocole doit donc clairement distinguer les liens bidirectionnels de ceux unidirectionnels, comme SOMoM le fait,

– l'environnement radio est très hétérogène. En d'autres termes, les graphes de disque unité représentent une mauvaise représentation des réseaux *ad hoc*. Deux mini-PC peuvent être proches sans qu'ils possèdent un lien radio les interconnectant. Un réseau maillé permettant l'accès à internet à des utilisateurs mobiles doit donc être dimensionné avec soin.

Par ailleurs, nous avons pu extraire de notre expérience quelques problèmes posés par les plates-formes d'expérimentation actuelles. Tout d'abord, la communauté n'a pas encore développé des scénarios types pour l'étude des performances. Ainsi, il est très complexe de comparer différents protocoles. Ensuite, IEEE 802.11 présente des problèmes majeurs pour les environnements multisaut : l'équité n'est pas respectée entre différents flux ou mêmes terminaux, les débits peuvent chuter de façon drastique en présence de nœuds cachés. Des scénarios problématiques types ont déjà été dégagés (Dhoutaut, 2003). De plus, IEEE 802.11 ne propose actuellement aucune qualité de service. Or, une telle fonction est vitale pour les réseaux multisaut, dans lesquels le trafic de contrôle n'est pas différencié du trafic de données. En effet, il suffit qu'une forte charge réseau perturbe le trafic de contrôle en créant des collisions pour que le fonctionnement global du réseau soit impacté. Les flux peuvent subir ainsi des fluctuations importantes de débits. Par ailleurs, IEEE 802.11 ne présente pas le même débit en unicast et en broadcast. Ainsi, les paquets de contrôle (en broadcast) sont envoyés au débit minimum de IEEE 802.11, et sont donc reçus par des nœuds plus lointains. Cette différence de portée radio unicast/broadcast est problématique pour nombre de protocoles. Il serait peut être judicieux de forcer une même vitesse de transmission pour les deux modes. Par contre, il serait dans un tel cas très complexe de gérer dans un même réseau des terminaux transmettant leurs informations à des débits différents selon la force du lien radio.

Les technologies sans-fil sont actuellement matures pour les scénarios de type cellulaire en IEEE 802.11 : l'équité est bien gérée et de la qualité de services peut être introduite. Bien que certains constructeurs comme Belair Networks (Belair Networks) ou Proxim (Proxim Wireless) commencent à proposer le déploiement de réseaux radio maillés, les applications sont encore limitées aux entreprises chargeant peu le réseau, comme les services de police ou de transport. Par ailleurs, des nœuds multi-interface, intégrant Wimax et IEEE 802.11 semblent une voie prometteuse afin d'étendre la couverture des réseaux maillés tout en réduisant le nombre de sauts. Les réseaux multi-technologie permettent par ailleurs de limiter les interférencs. Les terminaux multi-interface d'un réseau hybride pourraient ainsi relayer tout le trafic venant des terminaux plus *simples*, et donc augmenter le débit global du réseau.

7. Conclusion et perspectives

Cet article a présenté une plate-forme expérimentale pour l'évaluation d'une auto-organisation et d'un protocole de routage pour réseaux hybrides. Le protocole implémenté autorisant une connexion multisaut est disponible pour la communauté sur `sourceforge` (Theoleyre & Valois, 2006). Cet article décrit l'architecture logicielle et matérielle globale d'une plate-forme d'internet multisaut. Sa généricité de conception fait qu'une telle démarche est directement réutilisable pour l'implémentation et l'évaluation de performances d'un protocole quelconque pour réseau hybride. Enfin, nous avons démontré la pertinence et la flexibilité des réseaux hybrides : à terme, cette plate-forme expérimentale servira d'internet sans-fil usuel pour le laboratoire.

Nous avons mesuré les performances de ce réseau tant à travers des métriques globales qu'atomiques, et avons mesuré l'impact de la mobilité sur l'accès multisaut.

Nous avons par ailleurs pointé quelques difficultés posées par les plates-formes de recherche expérimentale pour réseaux sans-fil. En particulier, IEEE 802.11 présente des problèmes spécifiques dans les réseaux multisaut, notamment d'équité, de débit faible et d'absence de qualité de service. Une nouvelle couche MAC adaptée aux réseaux multisaut doit donc absolument être proposée. Cette plate-forme ne constitue que le point de départ pour la conception d'un réseau hybride intégré. Dans le futur, nous souhaitons notamment introduire des terminaux multi-interface combinant par exemple Wimax, IEEE 802.11 ou Ethernet. Cette multi-technologie doit être incluse dans le réseau hybride de façon transparente pour l'utilisateur. Nous souhaitons également auto-configurer les terminaux (e.g. acquisition automatique d'une adresse IP). Enfin, il serait intéressant de créer un pendant de PlanetLab pour le sans-fil, reliant des réseaux hybrides entre eux pour les expérimentations.

8. Bibliographie

Barron P., Weber S., Clarke S. and Cahill, V., « Experiences Deploying an Ad-hoc Network in an Urban Environment », In *Workshop on Multi-hop Ad hoc Networks : from theory to reality (REALMAN)*, 2005, Santorini, Greece, IEEE.

Belair Networks. `www.belairnetworks.com`.

Bicket J., Aguayo D., Biswas S. and Morris R, « Architecture and Evaluation of an Unplanned 802.11b Mesh Network », In *Conference on Mobile Computing and Networking (MOBICOM)*, 2005, Cologne, Germany, ACM.

Chakeres I. D. and Belding-Royer E. M., « AODV Routing Protocol Implementation Design », In *Workshop on Wireless Ad Hoc Networking (WWAN)*, 2004, p. 698–703, Tokyo, Japan.

Raychaudhuri D. *et al.*, 2009. « Orbit Lab », Available on : `http://www.orbit-lab.org`.

De P., Raniwala A., Sharma S. and Chiueh T.-c.. « Design considerations for a multihop wireless network testbed », *IEEE Communications Magazine* vol. 43, n° 10, p. 102–109, 2005.

De Couto D. S. J., Aguayo D., Chambers B. A. and Morris R., « Performance of multihop wireless networks : shortest path is not enough », *ACM SIGCOMM Computer Communication Review* vol. 33, n° 1, p. 3–88, 2003.

Dhoutaut D., *Etude du standard IEEE 802.11 dans le cadre des réseaux ad hoc : de la simulation à l'expérimentation*. Ph.D. thesis, INSA Lyon, 2003.

Gray D., Kotz R. S., Newport C., Dubrovsky N., Fiske A., Liu J., Masone C., McGrath S. and Yuan Y., « Outdoor experimental comparison of four ad hoc routing algorithms », In *Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, 2004, Venice, Italy, ACM.

Heusse M., Rousseau F., Berger-Sabbatel G. and Duda A., « Performance anomaly of 802.11b », In *INFOCOM*, 2003, San Francisco, USA, IEEE.

Huang E., Hu W., Crowcroft J. and Wassell I., « Towards Commercial Mobile Ad Hoc Network Applications : A Radio Dispatch System », In *MobiHoc*, 2005, p. 355–365, Urbana-Champaign, USA, ACM.

Kaba J. T. and Raichle, D. R., « Testbed on a desktop : strategies and techniques to support multi-hop MANET routing protocol development », In *Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2001, Long Beach, USA, ACM.

Kawadia V., Zhang Y. and Gupta B., « System Services for Implementing Ad-Hoc Routing : Architecture, Implementation and Experiences », In *Conference on Mobile Systems, Applications, and Services (MOBISYS)*, 2003, San Francisco, USA, ACM.

Kiess, W. and Mauve M., « A survey on real-world implementations of mobile ad-hoc networksnext term », *Ad Hoc Networks* vol. 5, n° 3, p. 324–339, 2007.

Klein-Berndt L. et.al, 2004. « Kernel AODV Implementation », Available on : `http://w3.antd.nist.gov/wctg/aodv\_kernel/`.

Maltz D. A., Broch J. and Johnson D. B., « Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed », Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon University, 1999.

Proxim Wireless. `http://www.proxim.com`.

Raychaudhuri D., Seskar I., Ott M., Ganu S., Ramachandran K., Kremo H., Siracusa R., Liu H. and Singh M., « Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols », In *Wireless Communications and Networking Conference (WCNC)*, 2005, New Orleans, USA, IEEE.

Rekhter Y. and Li T., « An Architecture for IP Address Allocation with CIDR », RFC 1518, IETF, 1993.

Sanghani S., Brown T. X., Bhandare S. and Doshi S., « EWANT : The Emulated Wireless Ad Hoc Network Testbed », In *Wireless Communications and Networking Conference (WCNC)*, 2003, New Orleans, USA, IEEE.

Stojmenovic I., Nayak A., Kuruvila J., Ovalle-Martinez F. and Villanueva-Pena E., « Physical layer impact on the design and performance of routing and broadcasting protocols in ad hoc and sensor networks », *Computer Communications* vol. Ê28, n° 10, p. 1138–1151, 2005.

Theoleyre F and Valois F., « Mobility Management in Multihops Wireless Access Networks », In *Personal Wireless Communications (PWC)*, 2005, Colmar, France, IFIP.

Theoleyre F and Valois F., 2006. « somomd », Available on : `http://sourceforge.net/projects/somom`.

Theoleyre F and Valois F., « A self-organization structure for hybrid networks », *Ad Hoc Networks* vol. 6, n° 3, p. 393–407, 2008.

Toh C. K., Chen R., Delwar M. and Allen D., « Experimenting with an Ad Hoc wireless network on campus : insights and experiences », *ACM SIGMETRICS Performance Evaluation Review* vol. 28, n° 3, p. 21–29, 2000.

Welte H. et al., 2005. « The Netfilter.org project », Available on : `http://www.netfilter.org/`.

Zhang Y. and Li W., « An integrated environment for testing mobile ad-hoc networks », In *Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2002, Lausanne, Switzerland. ACM.

Article reçu le 23 novembre 2006
Accepté après révisions le 19 février 2009

A. Lexique

ABR	Associativity-Based Routing
AODV	Ad Hoc On-Demand Vector Routing
AP	Point d'Accès
APRL	Any Path Routing without Loops
DSR	Dynamic Source Routing
MAC	Medium Access Control
MANET	Mobile Ad Hoc Networks
ODMRP	On-Demand Multicast Routing Protocol
SOMOM	Self-Organized Mobility Management
STARA	System- and Traffic-dependent Adaptive Routing Algorithm

Fabrice Theoleyre *a reçu un doctorat en Informatique de l'INSA de Lyon en 2006. Il est depuis 2007 chargé de recherche CNRS au Laboratoire Informatique de Grenoble. Ses recherches s'intéressent aux réseaux radio multisaut en général, et en particulier aux aspects algorithmiques et protocolaires dans les couches MAC et réseau. Il a participé à de nombreux comités de programmes de conférences internationales (e.g. PIMRC 2008, IWCMC 2007-2009) et est éditeur associé de IEEE Communications Letters.*

Fabrice Valois *est professeur des Universités à l'INSA Lyon depuis 2008. Il a passé son doctorat d'Informatique à l'Université de Versailles en 2000 sur la modélisation et l'évaluation de performances de réseaux cellulaires. Devenu MCF en 2000 à l'INSA Lyon, il a participé à la création du laboratoire CITI et a lancé les activités de recherche sur l'auto-organisation (auto-* plus généralement). Il a passé son habilitation à diriger des recherches sur l'auto-organisation de réseaux radio multi-saut.*

Annexe pour le service de fabrication

Article pour la revue :
RSTI - TSI

Auteurs :
Fabrice Théoleyre† *— Fabrice Valois*§

Titre de l'article :
Conception d'une plate-forme d'expérimentations pour réseaux ad hoc et hybrides

Titre abrégé :
Une plate-forme d'expérimentations

Traduction du titre :
Design of an experimental testbed for ad hoc *and hybrid networks*

Date de cette version :
3 juin 2009

Coordonnées des auteurs :

– téléphone : 04 76 82 72 19
– télécopie : 04 76 82 72 87
– Email : Fabrice.Theoleyre@imag.fr

Logiciel utilisé pour la préparation de cet article :
LaTeX, avec le fichier de style `article-hermes.cls`,
version 1.10 du 17/09/2001.

Formulaire de copyright :
Joindre le formulaire de copyright signé, récupéré sur le web à l'adresse
`http://www.hermes-science.com`