

---

# Prototypage de protocoles de communication sans fil

**Fehmi Ben Abdesslem\*** — **Ignacio Solis\*\*** — **Luigi Iannone\*** — **Marcelo Dias de Amorim\*** — **Katia Obraczka\*\*** — **Serge Fdida\***

\* *Université Pierre et Marie Curie – Paris VI*  
*Laboratoire d'Informatique de Paris 6*  
*75015 Paris, France*  
*{fehmi, iannone, amorim, sf}@rp.lip6.fr*

\*\* *University of California, Santa Cruz*  
*Computer Engineering Department*  
*CA 95064, USA*  
*{isolis, katia}@cse.ucsc.edu*

---

*RÉSUMÉ. Malgré le nombre élevé de protocoles de communication destinés aux réseaux sans fil, seule une petite partie a fait l'objet d'une implémentation dans des conditions réelles. En même temps, attendre l'implémentation finale d'une solution afin d'observer son comportement dans une situation réelle peut s'avérer très fastidieux. Dans cet article, nous suggérons d'insérer une phase de prototypage dans le processus de conception de ces protocoles. Ce prototypage s'appuie sur des outils de haut niveau afin d'aboutir à une implémentation entièrement fonctionnelle –bien que non-optimisée– du système. Nous montrons que cette méthode permet d'une part d'accélérer le processus de conception, et d'autre part d'aider à la validation des protocoles de communication sans fil en les confrontant aux conditions réelles. Pour cela, nous proposons Warp, la première plateforme opérationnelle de prototypage destinée aux protocoles de réseaux sans fil.*

*ABSTRACT. Although an impressive number of proposals addressing the multitude of challenges raised by wireless networking exist, few have known real implementation. At the same time, implementation generally requires great effort. In this paper, we advocate including prototyping in the design process of wireless protocols, relying on high-level tools to realize a functional version –although not optimized– of a system. Its goal is to speed up the design process and to help validating novel solutions under real conditions. To this end, we introduce Warp, the first prototyping framework for wireless network protocols.*

*MOTS-CLÉS : Prototypage, Protocoles, Réseaux sans fil, API*

*KEYWORDS: Prototyping, Protocols, Wireless networks, API*

---

## 1. Introduction

La conception de nouveaux protocoles de communication pour les réseaux sans fil auto-organisables est un sujet prolifique abondamment développé par la communauté scientifique depuis une dizaine d'années. Il en résulte une myriade de propositions de protocoles de communication pour les réseaux sans fil, à partir desquelles très peu d'implémentations ont finalement vu le jour. De multiples raisons peuvent expliquer ce fossé entre propositions théoriques et implémentations réelles, parmi lesquelles :

– *Complexité et difficultés d'implémentation.* L'implémentation de protocoles avancés exige des aptitudes spécifiques comme programmer au niveau du noyau, implémenter des fonctionnalités dans le pilote du périphérique de communication et avoir connaissance des particularités des systèmes d'exploitation.

– *La spécificité des solutions.* Les implémentations de protocoles sont souvent spécifiques à une solution particulière. Par conséquent, réutiliser du code devient délicat, augmentant les efforts nécessaires pour développer des nouvelles approches (par exemple, pour les protocoles de routage *cross-layer*)

– *La lenteur du processus d'implémentation.* Implémenter intégralement un nouveau protocole peut demander beaucoup d'effort et de temps. De plus, dans de nombreux cas, l'objectif est seulement d'étudier un paramètre en particulier et non pas le comportement global de la solution. Ainsi, la plupart des protocoles de communication sans fil proposés n'ont pas été testés au delà de la phase de simulation.

Notre objectif est de minimiser ce fossé entre les solutions théoriques et leur implémentation réelle, dans le contexte des réseaux sans fil. Pour cela, nous suggérons d'intégrer une phase de *prototypage assisté* dans le processus de conception de protocoles. Cette phase permet d'arriver plus rapidement à la version finale du protocole, et aide à leur validation sous des conditions réelles. Contrairement au prototypage traditionnel, qui consiste simplement à implémenter partiellement une solution, souvent à des fins démonstratives, la particularité du prototypage assisté réside dans l'utilisation d'outils de haut niveau afin d'aboutir rapidement à une version complète (mais non finale) d'un algorithme de communication.

Pour mettre en pratique cette approche, nous proposons Warp, la première plateforme de prototypage rapide, à notre connaissance, destinée à la conception de protocoles de communication pour réseaux sans fil.

Les principales contributions de cet article peuvent être résumées en deux parties. Tout d'abord, nous introduisons la notion de prototypage assisté dans le processus de conception des protocoles de communication pour réseaux sans fil, afin de combler le fossé entre théorie et pratique. Ensuite, nous concevons et implémentons une plateforme simple et puissante pour assister le prototypage de protocoles de communication sans fil. À travers un certain nombre de cas d'études, nous montrons la simplicité d'utilisation de Warp et l'utilité fondamentale de la phase de prototypage assisté.

La suite de cet article est organisée comme suit. Dans la Section 2, nous mettons en perspective notre contribution par un rapide tour d'horizon des méthodes existantes

utilisées pour la conception de protocoles de communication sans fil. Nous présentons Warp dans la section suivante 3 en décrivant ses deux principaux composants. Dans la Section 4, nous démontrons à travers divers exemples d'application combien Warp rend facile et rapide le prototypage des protocoles de communication sans fil. Enfin, nous concluons l'article dans la Section 5.

## **2. Le long chemin entre théorie et pratique**

Passer de la première idée conceptuelle d'un nouveau protocole à son implémentation finale et opérationnelle est un long parcours, jonché de nombreux obstacles. Depuis plus d'un quart de siècle, une solide expérience dans les réseaux filaires a cependant été acquise par les chercheurs et développeurs, motivée notamment par la popularité du réseau filaire à grande échelle qu'est Internet. Des méthodologies traditionnelles efficaces, telles que l'analyse mathématique, la simulation, ou encore l'émulation, ont donc été mises en place afin d'améliorer le processus de conception des protocoles. Ces méthodes d'évaluation ont permis d'obtenir des résultats représentatifs du comportement final des protocoles avant même leur implémentation, optimisant et simplifiant ainsi leur conception. Concevoir des protocoles de communication pour les réseaux sans fil est en revanche plus délicat, principalement en raison de l'instabilité du médium radio. Cette difficulté se manifeste en particulier lors de l'évaluation des protocoles.

### **2.1. L'évaluation de protocoles dans les réseaux sans fil**

Parmi les différentes étapes du processus de conception des protocoles, la phase d'évaluation est une étape fondamentale. Elle permet avant l'implémentation d'ajuster le protocole en fonction des résultats obtenus. Trois principales méthodes d'évaluation sont aujourd'hui utilisées lors de la conception des protocoles de réseaux sans fil : *l'analyse mathématique, la simulation, et l'émulation.*

L'analyse mathématique permet au concepteur de protocoles de calculer le comportement moyen d'un protocole, ainsi que le comportement asymptotique (par exemple, pour en étudier sa convergence). L'objectif de cette méthode est de cerner les principales caractéristiques d'une approche afin de parvenir à une compréhension claire et globale du système. Cependant, l'approche analytique entraîne souvent des simplifications grossières et des hypothèses trop fortes et peu fidèles à la réalité. La complexité de l'environnement sans fil, associée à tous les phénomènes imprévisibles (comme par exemple la propagation du signal, les interférences, ou encore l'environnement dynamique), requiert inévitablement de faire abstraction des conditions réelles, rendant souvent biaisés et imprécis les résultats d'évaluation.

La simulation permet quant à elle de parvenir à des observations plus approfondies qui ne peuvent être obtenues à l'aide de modèles analytiques. Par ailleurs, ces observations peuvent découler de scénarios difficilement réalisables sur une plate-forme

d'expérimentation réelle (par exemple des scénarios faisant intervenir des milliers de nœuds sur une très grande surface d'expérimentation), et simulés en faisant varier de nombreux paramètres (portée, densité, etc.). Parmi les simulateurs couramment utilisés pour évaluer les protocoles de communication sans fil, nous pouvons citer NS2 [NS2], OPNET [OPN], ou encore GloMoSim [ZEN 98]. Cependant, comme pour l'analyse mathématique, des simplifications doivent également être faites dans les scénarios de simulation, comme par exemple considérer une couverture circulaire pour chaque nœud, et un environnement à ciel ouvert exempt d'obstacles. Quant aux simulateurs spécialement développés pour évaluer un protocole donné, ils fournissent en général des résultats encore plus spécifiques, s'éloignant d'avantage des conditions réelles. La degré de confiance porté à la simulation est depuis longtemps remis en question par la communauté scientifique, notamment en ce qui concerne les réseaux sans fil. Ainsi, Cavin *et al.* [CAV 02] ont démontré qu'un même scénario de simulation pouvait donner des résultats différents selon le simulateur utilisé. Par ailleurs, dans l'écrasante majorité des simulations, Kurkowski *et al.* [KUR 05] révèle la négligence des chercheurs en ce qui concerne l'initialisation des paramètres de simulation, une légère variation de ceux-ci pouvant donner des résultats très différents.

En ce qui concerne l'émulation, l'objectif est de modifier les caractéristiques de bas niveau afin d'émuler le comportement du medium radio et des mécanismes d'accès au medium. Ainsi, des environnements d'émulation tels que EMPOWER [ZHE 03] et Seawind [KOJ 01] émulent un medium radio à partir d'interfaces filaires en changeant des propriétés de celles-ci, telles que les taux de pertes ou les délais de transmission. D'autres émulateurs comme m-ORBIT [RAM 05] émulent même la mobilité des nœuds par sauts virtuels sur une plate-forme de nœuds fixes. Un avantage spécifique de l'émulation dans le contexte des réseaux sans fil est d'éviter les contraintes d'espace lors du déploiement expérimental. Cependant, l'émulation en elle-même requiert des concepteurs une implémentation avancée de leur protocole, et éventuellement un passage de main à des techniciens, compliquant le processus de conception. Par conséquent, l'émulation ne permet ni d'obtenir des observations sous des conditions réelles, ni de simplifier le processus de conception de protocoles.

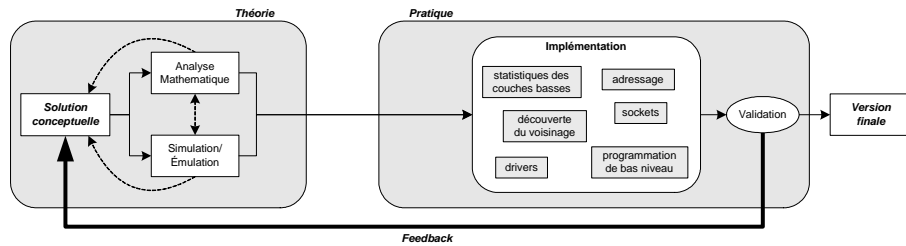
## **2.2. Le prototypage, une étape manquante**

Les inconvénients des méthodes d'évaluation mentionnés dans la section précédente peuvent être divisés en deux catégories :

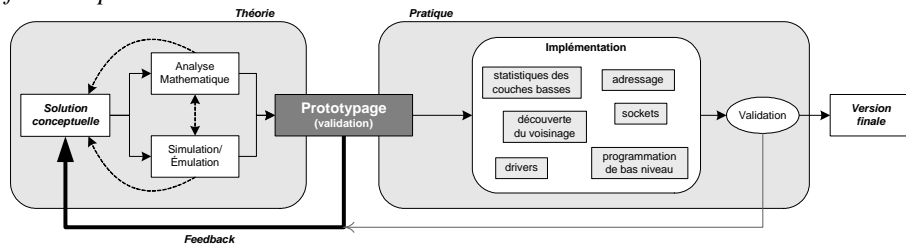
– *Trop simplistes* : Les fortes hypothèses et simplifications effectuées dans le processus d'évaluation débouchent sur des résultats ne concordant pas avec la réalité (analyse mathématique, simulation).

– *Trop complexes* : Les efforts requis pour coder et développer les scénarios d'évaluation du protocole sont comparables aux efforts à fournir pour obtenir une réelle implémentation (émulation).

Afin de mettre en place un nouveau protocole de communication, les concepteurs ont généralement recours au préalable à une approche simpliste pour l'évaluer, puis



**Figure 1.** Boucle traditionnelle pour passer d'un protocole théorique à sa version finale implémentée.



**Figure 2.** La boucle de contrôle réduite par l'insertion d'une phase de prototypage.

passent ensuite directement à la phase d'implémentation. Cette seconde et dernière phase, reléguée en général à des techniciens spécialisés, requiert du temps, des aptitudes en programmation, et des connaissances spécifiques de bas-niveau. Cependant, pour les protocoles de communication sans fil, ce simple processus de conception recèle un inconvénient majeur. Le fait est que, une fois implémenté, le comportement final d'un protocole sans fil dans un environnement réel ne correspond pas toujours au comportement escompté par les concepteurs, et observé lors de l'évaluation. Ce décalage conduit à une modification du protocole initial par le concepteur afin de prendre en compte les phénomènes observés dans l'environnement réel. La nouvelle version du protocole est alors évaluée une nouvelle fois, puis passée aux techniciens pour procéder à la phase d'implémentation et être confrontée à nouveau aux conditions réelles. Ce processus de conception récursif dessine une longue boucle (illustrée sur la figure 1), qui ne convergera qu'une fois que le comportement réel du protocole aura répondu aux attentes initiales du concepteur.

Afin de contourner ce long processus, nous proposons d'optimiser le processus de conception en y ajoutant une phase de *prototypage*, qui prendrait en compte une grande partie des conditions réelles de l'environnement sans fil, afin d'évaluer plus efficacement les protocoles avant leur implémentation et de faciliter ainsi sa conception. La figure 2 illustre notre proposition ; la boucle de retour d'information (*feedback*) devient alors plus courte, réduisant ainsi le temps de conception et la difficulté d'implémentation.

Ce concept de prototypage est déjà largement utilisé dans de nombreux domaines de l'industrie : automobile, aéronautique, architecture, etc. Dans les réseaux, des fabricants ont également recours aux prototypage lors du processus de développement de leurs routeurs [KOH 00]. Pour développer des protocoles de communication, une forme basique de prototypage est traditionnellement utilisée. Quelques rares travaux proposant des protocoles sans fil présentent en effet un prototype de leur proposition, souvent uniquement à des fins démonstratives. Mais, en général, ce prototype n'est qu'une implémentation partielle de leur proposition. De notre côté, nous proposons une plate-forme de prototypage, pour faciliter l'implémentation de prototypes complètement fonctionnels. Le prototype ainsi généré à l'aide de la plate-forme de prototypage est une version complète et opérationnelle du protocole, implémentée sous des conditions réelles.

### 3. Le prototypage avec Warp

Warp est une plate-forme de prototypage pour protocoles de communication sans fil destinée aux chercheurs et concepteurs de protocoles. La principale utilité de cet outil est de compléter les méthodes d'évaluation existantes comme la simulation ou l'analyse mathématique, en apportant au concepteur un nouveau regard sur le comportement final de son protocole dans des conditions réelles.

L'apport de cette méthode d'évaluation est multiple. D'abord, les mesures et observations obtenues à partir du prototype prennent en considération l'environnement réel, jusqu'alors difficile à modéliser avec les simulations. Le prototype du protocole étant implémenté réellement sur des interfaces sans fil, les phénomènes radio tels que la réflexion du signal, les interférences, ou encore la propagation à chemins multiples (*multi-path*), sont pris en compte dans les résultats.

De plus, la réelle mobilité des nœuds et la nature dynamique de l'espace environnant sont également inhérents aux expérimentations réalisées à l'aide du prototypage. Il devient alors inutile d'élaborer des modèles de mobilité ou d'établir un scénario virtuel réaliste.

Enfin, les expérimentations réalisées à l'aide de Warp reposent sur du matériel de communication sans fil disponible sur le marché. Ainsi, le comportement du protocole est observé en prenant compte des réelles capacités des interfaces de communication, en termes de portée par exemple, ou encore de contrôle de puissance.

Warp a été conçu pour être utilisé dans diverses situations, parmi lesquelles :

- L'implémentation de nouveaux protocoles de routage.
- L'expérimentation d'approches en recouvrement (*overlay*) dans les réseaux sans fil multi-sauts.
- L'évaluation d'algorithmes de sécurité distribués.
- La mesure de connectivité sans fil dans des scénarios d'intérieur ou d'extérieur.
- L'estimation de l'impact de l'environnement sur des protocoles.

- Le développement rapide de versions de démonstration.

Pour se faire, Warp se divise en deux principaux modules : la librairie Warp (*Warp Library*) et le moteur Warp (*Warp Engine*). La librairie Warp met à disposition du concepteur de protocoles un set de fonctions de haut niveau décliné en plusieurs langages de programmation. Le *Warp Engine* est un processus actif s'exécutant en arrière plan. L'objectif de ce module est de traduire les primitives de haut niveau en primitives dépendantes du noyau (et *vice-versa*), tout en implémentant les principales fonctionnalités requises par les protocoles de communication, telles que la découverte du voisinage, l'évaluation de la qualité des liens, etc. Dans sa version actuelle, Warp s'exécute sur les architectures PC (par exemple des mini-PCs, des ordinateurs portables, ou des ordinateurs de bureau) utilisant un système d'exploitation Linux.

### 3.1. La librairie Warp

La librairie Warp fournit un jeu de primitives de haut-niveau orientées particulièrement vers le routage et implémentées en différents langages de programmation. Il est possible à l'aide de ces primitives de faire abstraction des détails de bas niveau, comme l'établissement de la communication, les sockets, les adresses, etc. Pour garantir une simplicité d'utilisation, le nombre de primitives a volontairement été réduit. Ce choix est motivé par notre volonté de rendre l'implémentation du prototype plus simple que l'implémentation directe du protocole. L'utilisateur doit donc pouvoir maîtriser la plate-forme de prototypage Warp intuitivement, en s'appuyant sur une documentation aussi claire et réduite que possible. En revanche, la simplicité des primitives ne doit en aucun cas nuire à l'universalité de la plate-forme, qui doit pouvoir générer les prototypes de la plupart des protocoles et algorithmes de communication pour les réseaux sans fil, dans toute leur diversité. A titre d'exemple, citons les primitives suivantes :

- **Warp\_Neighbors()** : Retourne la liste de voisins directs et à 2 sauts, ainsi que les statistiques de qualité de lien correspondantes.
- **Warp\_Send(Data, ID, Pwr)** : Envoie un paquet de données contenant **Data** au voisin **ID**. L'argument optionnel **Pwr** permet de sélectionner la puissance d'émission.
- **Warp\_Send\_Broadcast(Data, Pwr)** : Envoie un message en broadcast contenant **Data**. L'argument optionnel **Pwr** permet de sélectionner la puissance d'émission.
- **Warp\_Receive()** : Vérifie si un message a été reçu par le processus actif de Warp ; retourne le message le cas échéant.

La librairie communique avec le processus actif de Warp par l'interface locale (*loop-back*), à l'aide d'un simple mécanisme de requête/réponse. En pratique, la librairie se présente sous la forme d'un fichier (ou deux, selon les langages) à inclure en en-tête de l'implémentation du prototype.

### 3.2. Le moteur Warp

Les principaux objectifs du processus actif de Warp sont : (i) d'assurer une mise à jour en temps réel du voisinage (à 2 sauts) ainsi que des informations relatives à la qualité des liens ; (ii) de fournir, à la demande de la librairie, une mise à jour de ces informations ; (iii) d'encapsuler et de décapsuler les messages envoyés et reçus, respectivement. Nous détaillons brièvement par la suite les mécanismes mis en œuvre pour obtenir les informations sur le voisinage.

**Le mécanisme d'envoi de balises.** Afin d'établir et de garder à jour la liste de voisins, chaque nœud implémentant Warp envoie périodiquement des balises en *broadcast* en augmentant la puissance de transmission de manière cyclique. Ces balises contiennent des informations sur le transmetteur ainsi que des informations propres au mécanisme (puissance d'émission de la balise, numéro de séquence, etc). Les valeurs des puissances de transmission sont configurables par l'utilisateur et dépendent des capacités de l'interface radio. La réception de ces balises permet de générer une série de statistiques utile aux protocoles de communication. La valeur de la période d'envoi (qui est par ailleurs configurable selon la densité et la stabilité du réseau) est en effet incluse dans les balises. Ainsi, chacun des voisins connaît cette période et peut détecter la perte d'une balise.

**Le mécanisme de retour d'information (*feedback*).** Les statistiques obtenues à l'aide du mécanisme précédemment décrit sont collectées uniquement en recevant les balises. Un mécanisme de *feedback* est utilisé en parallèle afin d'obtenir des statistiques bilatérales. Les paquets de *feedback* sont régulièrement envoyés aux voisins, incluant la valeur minimale des puissances de balises reçues, afin de caractériser une fois encore la qualité du lien. Ainsi, un nœud recevant seulement les balises d'un voisin émises à au moins 20 mW, lui envoie cette valeur dans le paquet de *feedback*.

D'autre part, ce mécanisme permet d'obtenir des informations à 2 sauts, puisque les paquets de *feedback* sont envoyés en *broadcast*.

**Les informations retournées par le processus actif.** La figure 3 montre une capture d'écran des informations fournies par la console du processus actif de Warp, pour le nœud dont l'identifiant est **Bob**. **Bob** a deux voisins directs **John** et **Alice**. Néanmoins, pour les faibles puissances d'émission, les balises d'**Alice** sont mieux reçues que celles de **John**. À la réception d'une balise, Warp collecte la puissance de réception du paquet (RSSI, *Received Signal Strength Indicator*), permettant ainsi de caractériser d'une autre manière encore la qualité du lien. Connaissant la puissance utilisée pour émettre la balise, le nœud récepteur peut donc également en déduire la perte du signal subie par la transmission. De nombreux protocoles, notamment de routage, peuvent exploiter ces nombreuses informations pour évaluer la qualité des liens.

```

===== Neighbor List for node BOB =====
=====
2F6D Active 00:40:96:A9:2F:6D Beacon period : 10000 JOHN
Weakest beacon received by this neighbor : 1 mW
-----
1mW Active @9860 R 0 S 9 B 9 [0.015848932 nW (-78 dBm)] 4/5
12mW Active @9861 R 0 S 9 B 9 [0.079432823 nW (-71 dBm)] 4/5
21mW Active @9863 R 0 S 9 B 10 [0.199526231 nW (-67 dBm)] 5/5
29mW Active @9865 R 0 S 9 B 10 [0.501187234 nW (-63 dBm)] 5/5
60mW Active @9856 R 0 S 8 B 10 [1.995262315 nW (-57 dBm)] 5/5

2hop-Neighbors : ALICE (1 mW, -55 dBm) JACK (1 mW, -62 dBm)
=====

=====
30A7 Active 00:40:96:A9:30:A7 Beacon period : 10000 ALICE
Weakest beacon received by this neighbor : 1 mW
-----
1mW Active @9857 R 0 S 9 B 9 [0.158489319 nW (-68 dBm)] 5/5
12mW Active @9859 R 0 S 9 B 10 [0.794328235 nW (-61 dBm)] 5/5
21mW Active @9860 R 0 S 9 B 10 [1.995262315 nW (-57 dBm)] 5/5
29mW Active @9862 R 0 S 9 B 10 [3.981071706 nW (-54 dBm)] 5/5
60mW Active @9863 R 0 S 9 B 10 [12.58925412 nW (-49 dBm)] 5/5

2hop-Neighbors : JACK (1 mW, -49 dBm) JOHN (1 mW, -56 dBm)
=====

```

**Figure 3.** Exemple d'informations obtenues à l'aide du processus actif de Warp, concernant le nœud dont l'identifiant est "**BOB**."

#### 4. Exemples simples d'utilisation de Warp

Dans cette section, nous illustrons l'universalité de la plate-forme et sa simplicité d'utilisation à travers des exemples basiques et variés de protocoles et d'algorithmes de communication. Des prototypes de protocoles plus complexes peuvent bien sûr également être implémentés avec Warp (comme les protocoles de routage AODV ou DSR), mais notre choix s'est plutôt porté sur des algorithmes de communication plus simples pour présenter plus clairement l'utilisation de Warp.

**Inondation.** L'algorithme d'inondation consiste à diffuser les paquets à travers l'ensemble d'un réseau. Chaque nœud recevant un paquet le retransmet donc à ses voisins, une seule fois. Utiliser la plate-forme Warp pour implémenter cet algorithme permet de faire abstraction des sockets, des ports à utiliser, de l'adressage, des interfaces, etc. Seules les primitives **Warp\_Receive** et **Warp\_Send\_Broadcast** sont utilisées par le concepteur.

Les 12 lignes de code Perl suivantes s'exécutent avec succès sur notre réseau sans fil d'expérimentation implémentant Warp.

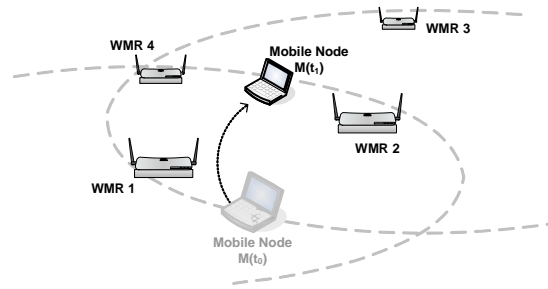
Prototype en Perl.	Pseudo-code.
<pre>require "warp.pl"; while(1){   while(!@Message){     \$Message = Warp_Receive();   }   \$msgID = unpack("N",\$Message[4]);   if (!grep(/\$msgID/,@ID_list)){     push(@ID_list,\$msgID);     Warp_Send_Broadcast(\$Message[4]);   }   @Message = (); }</pre>	<pre>while 1 do   while rien reçu do     Packet ← Warp_Receive()   end while   PacketID ← Packet.ID   if ListID ne contient pas PacketID   then     Ajouter PacketID à ListID     Warp_Send_Broadcast(Packet)   end if   Effacer Packet end while</pre>

**Contrôle de topologie.** Le contrôle de topologie requiert des informations à jour concernant le voisinage. Sélectionner de bons voisins est souvent primordial dans un réseau auto-organisable. La plate-forme Warp permet le prototypage d'algorithmes de contrôle de topologie basés sur des critères très variés de sélection de voisins : par exemple en choisissant les voisins avec la plus faible puissance de transmission requise (pour économiser de l'énergie et réduire les interférences), ou bien en sélectionnant les voisins avec la plus forte puissance de réception de signal.

Prototype en Perl.	Pseudo-code.
<pre>require "warp.pl"; \$Neighbor = Warp_Neighbors(); for (\$i=1; \$i&lt;=\$Neighbor[0]; \$i++){   push(@rx, [\$i,\$Neighbor[\$i]{MAX_POW}}); } @s_list = sort {{\$b}-&gt;[1]&lt;=&gt;(\$a)-&gt;[1]} @rx; @Selected_Neighbors=@sorted_list[0..1];</pre>	<pre>Voisinage ← Warp_Neighbors() for i=1 to Nbre_de_voisins do   Ajouter (MAX_RSSI de i) à List end for Liste_triée ← trier List Voisins ← 2 premiers de Liste_triée</pre>

Le code présenté ici, qui s'exécute avec succès sur notre réseau expérimental, montre comment obtenir en 7 lignes une liste de voisins, sélectionnés selon la puissance du signal reçue.

**Localisation de nœuds.** Dans les réseaux maillés sans fil, les nœuds fixes formant l'infrastructure sont également connectés à des nœuds mobiles. Comme illustré dans la figure 4, lorsque les nœuds **WMR1**, **WMR2** et **M** sont tous voisins (à portée), un nœud fixe **WMR1** peut approximativement déterminer la direction du déplacement de son voisin **M**. En effet, considérons que **WMR1** observe une perte de qualité sur son lien avec **M**. **WMR1** peut alors utiliser les messages de *feedback* envoyés par le nœud fixe **WMR2** vers **M**. En prenant en compte ce *feedback*, **WMR1** constate que le lien **WMR2-M** ne subit pas de perte de qualité lorsque **M** s'éloigne. Par conséquent, le nœud **WMR1** peut en déduire que **M** se déplace approximativement vers **WMR2**. Une telle utilisation des informations fournies par le processus actif de Warp peut par exemple permettre le prototypage simple d'approches de routage *cross-layer*.



**Figure 4.** Scénario d'évaluation du déplacement d'un nœud dans un réseau maillé.

---

#### Prototype en Perl.

---

```

foreach $i (@Selected_Neighbors){
  for (my $j=1 ; $j<=$Neighbor[$i->[0]]{Neighbors}[0] ; $j++){
    if ($Neighbor[$i->[0]]{Neighbors}[$j]{NAME} eq "M"){
      push(@list_M,[$Neighbor[$i->[0]]{NAME},$Neighbor[$i]{Neighbors}[$j]{RSSI}]);
    }
  }
}

```

---

#### Pseudo-code.

---

```

for chaque voisin i de Voisins do
  for j=1 à (Nombre de voisins à 2 sauts pour i) do
    if (nom du voisin à 2 sauts j) = "M" then
      Ajouter (identifiant de i, RSSI avec M) à List_M
    end if
  end for
end for

```

---

Le code Perl présenté ici reprend la liste de voisins obtenue précédemment, et crée la liste (**list\_M**) des puissances de signal avec le nœud **M**. En exécutant ce code, **WMR1** peut évaluer la qualité des liens **M-WMR4** et **M-WMR2** afin de déterminer la direction de déplacement de **M**.

Ces différentes utilisations des informations fournies par Warp illustrent son universalité. L'objectif d'une telle plate-forme est de pouvoir fournir un maximum de fonctionnalités pour couvrir les besoins de la plupart des protocoles de communication pour réseaux sans fil. Naturellement, il serait illusoire d'espérer prototyper l'intégralité des protocoles sans fil. Certains algorithmes de communication, notamment ceux basés sur des fonctionnalités de bas niveau non proposées par Warp, ne pourront être prototypées. D'autre part, tout comme la simulation, Warp ne permet pas de prévoir avec précision les performances de l'implémentation finale de l'algorithme de communication. Il apporte cependant une première idée du comportement final de l'algorithme, en prenant en compte des paramètres réels ignorés par la simulation, comme la connectivité des nœuds ou la propagation du signal.

## 5. Conclusion

L'implémentation complète d'un protocole de communication est une tâche longue et ardue. De plus, l'évaluation traditionnelle des protocoles pour réseaux sans fil à l'aide des simulations ou des analyses mathématiques ne parvient pas à prévoir avec exactitude le comportement final après implémentation. Pour ces raisons, nous avons proposé dans cet article l'inclusion d'une phase de prototypage dans le processus de conception. Cette approche permet d'une part d'accélérer la conception des protocoles de communication pour réseaux sans fil en s'appuyant sur des outils de haut niveau, et constitue d'autre part une méthode d'évaluation complémentaire aux méthodes traditionnelles, prenant en compte les phénomènes physiques de propagation du signal sous des conditions réelles.

Pour mettre en pratique cette notion de prototypage assisté, nous proposons Warp, la première plateforme de prototypage destinée aux protocoles de communication pour réseaux sans fil. Cette plate-forme est opérationnelle et fonctionne actuellement sur du matériel disponible sur le marché. Warp est implémenté sur un réseau expérimental composé de miniPCs (sous Linux) équipés d'interfaces sans fil 802.11.

L'évaluation de Warp passe par des mesures de performance, pour estimer la charge (*overhead*) induite par la plate-forme. Outre ces mesures, nos travaux futurs s'orientent vers le portage de Warp sur d'autres systèmes d'exploitation, et la traduction de la librairie en d'autres langages de programmation. Nous espérons également ajouter de nouvelles fonctionnalités et tester le concept de prototypage assisté sur d'autres technologies radio (autres que IEEE 802.11) et d'autres architectures (assistants personnels, téléphones mobiles, etc).

## 6. Bibliographie

- [CAV 02] CAVIN D., SASSON Y., SCHIPER A., « On the Accuracy of MANET Simulators », *In proceedings of POMC'02*, Toulouse, France, octobre 2002.
- [KOH 00] KOHLER E., MORRIS R., CHEN B., JANNOTTI J., KAASHOEK M. F., « The click modular router. », *ACM Trans. Comput. Syst.*, vol. 18, n° 3, 2000, p. 263-297.
- [KOJ 01] KOJO M., GURTOV A., MANNER J., SAROLAHTI P., ALANKO T. O., RAATIKAINEN K. E. E., « Seawind : a Wireless Network Emulator », *MMB*, Aachen, Germany, septembre 2001, p. 151-166.
- [KUR 05] KURKOWSKI S., CAMP T., COLAGROSSO M., « MANET simulation studies : the incredibles », *Mobile Computing and Communications Review*, vol. 9, 2005, p. 50-61.
- [NS2 ] NS2, THE NETWORK SIMULATOR, <http://www.isi.edu/nsnam/ns/>.
- [OPN ] OPNET MODELER, <http://www.opnet.com/products/modeler/>.
- [RAM 05] RAMACHANDRAN K., KAUL S., MATHUR S., GRUTESER M., SESKAR I., « Towards Large-Scale Mobile Network Emulation Through Spatial Switching on a Wireless Grid », *Proceedings of ACM Sigcomm'05*, Philadelphia, PA, août 2005.
- [ZEN 98] ZENG X., BAGRODIA R., GERLA M., « GloMoSim : a Library for Parallel Simulation of Large-Scale Wireless Networks », *Proc. of PADS98*, Banff, Canada, mai 1998.
- [ZHE 03] ZHENG P., NI L., « EMPOWER : A Network Emulator for Wireless and Wireline Networks », *Proceedings of IEEE Infocom'03*, San Francisco, CA, avril 2003.