Réalisation d'une API de configuration automatique de réseau WiFi maillé sous Linux

BUGNOT Thibaut

22 février 2018

Table des matières

Introduction **2**

1 Objectifs, definitions, contraintes **3**
1.1 Introduction aux réseaux wifi 3
1.2 La norme 802.11s 4
1.3 Adressage et routage 6

2 Réseaux sans fils sous Linux **7**
2.1 Gestion du réseau sous Linux 7
2.2 Création de réseaux 802.11s 8
2.3 Détection de réseaux existants et sélection de canal 8

3 Adressage et routage **11**
3.1 Adressage dans un réseau maillé 11
3.2 Routage 11

4 Implémentation et architecture logicielle **12**

Introduction

Chapitre 1

Objectifs, definitions, contraintes

1.1 Introduction aux réseaux wifi

Le wifi, abréviation de wireless fidelity, est un ensemble de protocoles permettant la communication sans fil entre deux appareils en utilisant des ondes radios. Ces protocoles se situent au niveau de la couche d'accés du modéle tcp/ip. La standardisation de cette norme à été initié l'IEEE [1] en 1990. Cela à aboutit, en 1997, au standart IEEE 802.11 définissant les réseaux locaux sans fils [1]. La norme d'origine prévoyait l'utilisation d'ondes radios dans la bande de fréquences libre entre 2401 et 2495 MHz[2], courrament appelée bande à 2,4 GHz, ou d'infra rouges. Cependant, pour suivre l'évolution des technologies, le standart IEEE 802.11 s'est enrichi afin d'augmenter le débit et d'utiliser la bande de fréqences libre entre 5170 et 5710 MHz. Les standarts IEEE 802.11a et IEEE 802.11b à donc été définit en 1999, le standart 802.11g et 2003 et le standart 802.11n en 2009.

Depuis sa création, la norme IEEE 802.11 définit 14 cannaux dans la bande 2,4 GHz. Chaque canal à une largeur de 22 MHz et l'écart entre les centres de deux canaux successifs est de 5 MHz [2]. Il en résulte donc un fort recouvrement entre les différents cannaux comme le montre la figure 1.1.

Un réseau wifi est un réseau local découpé en "cellules" appelée BSS [3]. Deux appareils doivent se trouver dans le même BSS pour communiquer entre eux.

1. Institute of Electrical and Electronics Engineers
2. Sauf les centres des cannaux 13 et 14 qui sont espacé de 12 MHz
3. Basic Service Set

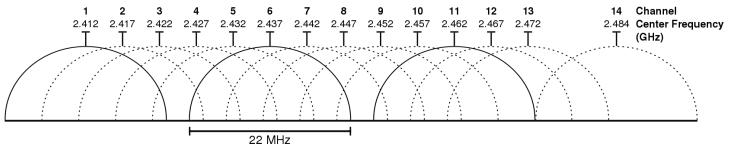

FIGURE 1.1 – Répartition des cannaux dans la bande 2,4 GHz

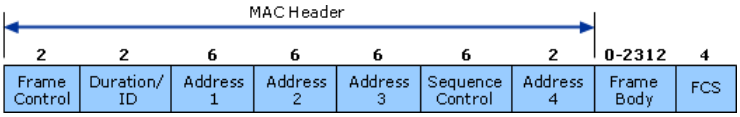

FIGURE 1.2 – Format des trames 802.11

Il existe deux modes de BSS : Le mode Infrastructure et le mode ad-hoc[3]. La pluspart des réseaux wifi de particuliers ou d'entreprises sont des réseaux en mode Infrastructure.

Le mode infrastructure est une topologie centralisé. Il se caractérise par le fait que chaque BSS posséde une station de base, appelé aussi point d'accés, et que toutes les communications passent nécessairement par le point d'accés de la BSS, et ce même si l'émeteur et le récepteur du message se trouvent dans le même BSS. Un point d'accés peut être relié par un réseau cablé à un ou plusieurs autres points d'accés, étendant ainsi le LAN [4] [4],ou à un routeur pour accéder à un réseau WAN [5]. Le mode ad-hoc, au contraire, est un mode "d'égal à égal". Deux entités au sein du même BSS peuvent communiquer directement.

Comme le montre la figure 1.2[5], le premier champ de l'en tête wifi est le FCF [6], permettant d'identifier les trames en fonction de leur rôle. Ainsi, les trames peuvent être de trois types, identifiées par les deux bits en position 3 et 4 du FCF : Management, Contrôle ou Donées.[6]. Les 4 bits suivants identifient le sous type, et les 8 derniers bits sont des flags. Les trames de données sont utilisé pour transporter des données de plus haut niveau. Les trames de contrôles sont utiliser pour les acquittements et les réservations, et les trames de management servent à organiser et maintenir le réseau[7].

Les Beacons frames sont des trames de management particuliéres qui permettent à un point d'accés de déclarer sa présence aux appareils à proximité. Ils transportent différentes informations comme le SSID [7] du réseau, qui est une chaîne de 2 à 32 charactéres, un timestamp permettant de se synchroniser, le canal sur lequel il émet, et d'autres informations.[3].

1.2 La norme 802.11s

Comme dit précédamment, le mode infrastructure est actuellement le plus utilisé. Cependant, il posséde des limites du au fait que, dans certaines situations, il n'est pas toujours possible de connecter un point d'accés à un switch[8]. En effet, la longueur des cables ethernet est limité, ce qui rend difficile le déploiement de points d'accés dans des environnement ouverts.

C'est ce qui fait la force du mode ad-hoc. Chaque appareil peut communiquer avec tous les autres appareils qui sont à portée. De plus, chaque appareil peut relayer le message si le destinataire final n'est pas à portée. Ainsi, si l'on prends l'exemple la figure 1.3, chaque noeud peut communiquer avec nimporte quel autre, à la condition qu'un alogrithme de routage s'execute sur le réseau et que chaque noeud sache quel est le suivant pour atteindre la destination. Ce genre

4. Local Area Network, ou Réseau local
5. Wide Area Network, ou Réseau étendu
6. Frame Control Field, ou Champ de contrôle de trame
7. Service Set IDentifier

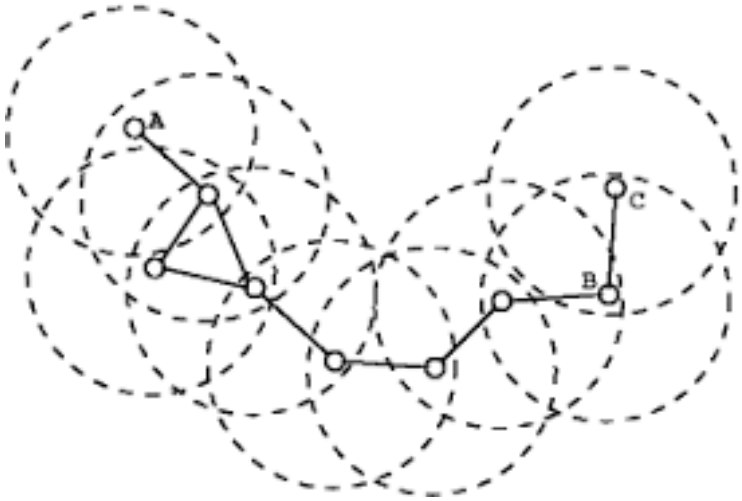

FIGURE 1.3 – Exemple de réseau ad-hoc

de réseau est appelé réseau maillé [8]. Le gros avantage de ces réseaux est qu'ils sont trés flexible. On peut les étendre sans avoir à tirer de nouveaux cables ou a ajouter de nouveaux equipements intermédiaires[8]. A l'inverse, leretrais d'un petit nombre de noeuds ne doit pas empécher le réseau de fonctionner si il est possible de trouver des routes alternatives pour les trames.

Le standart 802.11s est un amendement de la norme 802.11, définissant la maniére dont les appareils disposant d'une carte réseau sans fils peuvent s'interconnecter pour former un réseau sans fil maillé. L'IEEE a commencé a travailler sur ce standart en 2003 et celui-ci à été adopté en 2006. Pour faciliter l'interopérabilité, un réseau 802.11s est vu de l'extérieur comme un unique segment ethernet. Pour permettre la retransmission des informations d'un noeud à l'autre, la norme 802.11s étends l'en tête 802.11 classique avec un en tête mesh comme montré dans la figure 1.4[7].

Les 4 champs d'adresses de l'en tête 802.11 sont utilisées, puisqu'il faut à chaque transmission du message donner l'adresse du noeud qui à effectué la transmission, du prochain noeud, du destinataire final et de l'expéditeur originel. Dans certains cas plus complexes, par exemple si l'émeteur ou le destinataire, ou les deux, ne se trouvent pas dans le réseau mesh, mais que la trame va traverser un réseau mesh, il faut ajouter des adresses supplémentaires, d'ou le fait que l'en tête mesh comporte un champ optionel d'extention d'adresse. Parmis les autres valeurs ajoutées, le TTL [9] et le Mesh sequence number [10] permettent d'éviter les boucles infinies qui risqueraient de saturer le réseau.

8. Mesh Network en anglais
9. Nombre de fois maximal que peut être relayé une trame avant d'être abandonnée, cette valeur est décrémenté à chaque saut
10. nombre identifiant de maniére unique un paquet

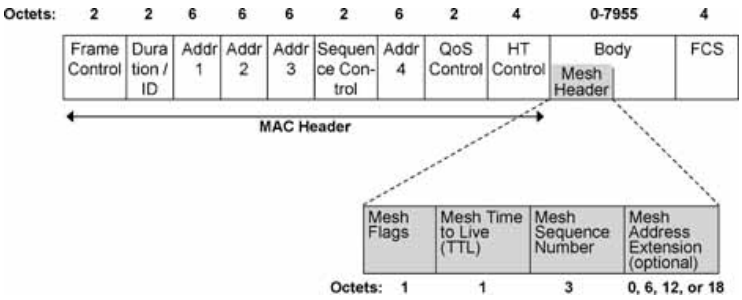

FIGURE 1.4 – Format des trames 802.11s

1.3 Adressage et routage

Dans un réseau TCP/IP, chaque noeud doit disposer de deux adresses. Chacune permet de l'identifier, en théorie, de maniére unique. La premiére est l'adresse MAC, une adresse sur 48 bits, utilisé pour identifier les noeuds dans les protocoles de la couche d'accés du modéle TCP/IP. Cette adresse permet à une trame de voyager sur un LAN jusqu'à sa destination, mais sera changé à chaque fois que le paquet passe par un routeur. La deuxiéme est l'adresse IP, une adresse sur 32 bits qui est utlisé par le protocole IP, qui est un protocole de la couche réseau du modéle TCP/IP. Cette adresse est inchangé d'un bout de la transmission à l'autre [11].

L'adresse MAC est atribué à une carte réseau par le constructeur. Ainsi, nous avons la garantie que chaque appareil posséde une adresse MAC unique. L'adresse IP doit également être unique mais, contrairement à l'adresse MAC, elle n'est pas enregistré dans la carte réseau par le constructeur car toutes les adresse IP identifiant les appareils d'un même LAN doivent avoir le même préfixe. Il existe des protocoles permettant d'affecter automatiquement des adresses IP à des appareils sans avoir besoin de recourir à une intervention humaine. Le protocole majoritairement utilisé est DHCP [12]. Ce protocole nécessite qu'un serveur dispose d'une liste d'adresses IP disponible qu'il va affecter à chaque noeuds du réseau sur demande de ces derniers[9]. Néamoins, le recours à un serveur central d'adresse IP amoindrit les avantages à l'utilisation d'une infrastructure décentralisé tel qu'un un réseau maillé.

Dans un réseau maillée, il est aussi nécessaire de prévoire le routage des trames. La norme 802.11s définit également le protocole HWMP [13] comme protocole de routage pour les réseaux wifi maillées. Contrairement à la majorité des protocoles de routages, HWMP ne se base pas sur les adresses IP, mais sur les adresses MAC, puisque le but est d'aiguiller les trames au sein d'un même LAN. Il s'agit d'un protocole de routage à vecteur distance puisque les noeuds n'ont pas conaissance de l'intégralité de la topologie du réseau mais uniquement des noueds qui le constituent et de la "distance" de chacun d'eux [10].

11. en l'absence de mécanismes de traduction d'adresse (NAT)
12. Dynamic Host Configuration Protocol
13. Hybrid Wireless Mesh Protocol

Chapitre 2

Réseaux sans fils sous Linux

2.1 Gestion du réseau sous Linux

Une grosse partie du projet consiste à pouvoir communiquer avec les interfaces sans fils dont dispose les appareils. En effet, il est nécessaire d'une part, de récupérer des informations à propos de ces derniéres et d'autre part, de les configurer pour arriver à les utiliser de la maniére que nous le souhaitons. Pour gérer les interfaces sans fils, il est nécessaire de bien distinguer deux niveaux : Les cartes réseaux [1] , qui existent physiquement, et les interfaces que l'on pourait qualifier de "logiques" qui utilisent ces cartes réseaux et sont utilisées pour envoyer et recevoir des données. C'est aux interfaces que l'on associe un type (Point d'acces, client, noeud mesh, monitor, ...) determinant son mode de fonctionnement. Une seule carte réseau peut être utilisé par plusieures interfaces.

Historiquement, sous linux, la communication avec les interfaces se faisait avec des appels systémes ioctl [2]. Des outils permettant de manipuler les interfaces en utilisant cette méthode sont depuis longtemps fournis avec les distribution linux. C'est le cas par exemple du package net-tools, incluant le programme ifconfig, et permettant de manipuler les interfaces (état, informations d'adressages) ou de wireless-tools, incluant le programme iwconfig, permettant de mainpuler plus précisément les interfaces sans fils.

Cependant, depuis 2007, il existe un autre moyen de manipuler les interfaces. En effet, se développe **netlink**, une famille de socket ayant pour but de fare communiquer les processus entre eux. Cela permet, entre autre, de faire communiquer un processus utilisateur avec un processus du noyeau linux. La librairie libnl implémente les pré-requis fondamentaux pour utiliser le protocole netlink. Cependant, celle ci se veux minimaliste. C'est pourquoi elle est complété par 3 API : libnl-route, libnl-genl et libnl-nf[11]. Des outils de configuration d'interfaces utilisent ces nouveaux moyens. C'est la cas de la suite d'outil iproute pour controles les interfaces et de iw pour controler plus précisément les interfaces sans fils. Nous utiliserons netlink dans notre code.

L'API que nous utiliserons principalement pour gérer les interfaces est libnl-genl. Celle ci permet, grace à nl80211, un en-tête netlink, d'envoyer des mes-

1. généralement abrégé **wiphy**, pour WIreless PHYsical device
2. Abréviation de Input Output ConTroL, il s'agit d'une fonction permettant de manipuler des fichiers spéciaux

sages permettant de contrôler les interfaces wifi. Les commandes qu'il est possible d'envoyer sont définies dans l'énumération nl80211_commands dans le fichier nl80211.h. En créant des messages netlink avec ces commandes, il nous est possible, entre autre, de récupérer les informations dur les cartes réseaux connectés (NL80211_CMD_GET_WIPHY), celles sur les différentes interfaces (NL80211_CMD_GET_INTERFACE), de choisir la fréquence d'émission de de réception de la carte réseau (NL80211_CMD_SET_WIPHY), de créer une interface sur une carte réseau (NL80211_CMD_NEW_INTERFACE) ou de changer le type d'une interface (NL80211_CMD_SET_INTERFACE)

Les commandes transmisent dans un message netlink peuvent avoir besoin d'un ou plusieurs attributs. La liste de tous les attributs existants dans le cadre de la gestion des interfaces sans fils est définit dans l'énumération nl80211_attrs dans le fichier nl80211.h.

L'API libnl-genl dispose déja de fonction permettant de créer des socket netlink, des messages netlink, d'y ajouter des attributs et de l'envoyer sur le socket. Selon la commande donnée, il se peut que le noyeau réponde en renvoyant un ou plusieurs messages netlink contenant des informations. C'est le cas lorsque l'on utilise par exemple la commande NL80211_CMD_GET_INTERFACE. Le message renvoyé contiendra alors les informations sur les interfaces. Il est alors nécessaire d'analyser le massage pour en extraire les informations souhaitée. Dans tous les cas, Le dernier message renvoyé par le noyeau sera un message d'acquitement ou, en cas de probléme, un message d'erreur. Aprés avoir envoyer un message netlink, il est donc nécessaire d'écouter sur le socket jusqu'à la réception d'un de ces deux messages.

2.2 Création de réseaux 802.11s

2.3 Détection de réseaux existants et sélection de canal

Un des objectif du projet consiste à détecter les réseaux wifi mesh déja existant et, si il n'en existe aucun, à choisir le meilleur canal, c'est à dire le moins encombré, pour en créer un nouveau. Pour cela, il est nécessaire d'être en mesure de scanner les réseaux déja existants. A cette fin, les beacons frames décrites chapitre 1, partie 1 sont particuliérement utiles. En effet, elles sont envoyées à intervales réguliers par les points d'accés, mais aussi par tous les noeuds d'un réseau mesh. Ces trames contiennent en effet de nombreuses informations à propos du réseau. Ces informations n'étant pas toujours les mêmes en fonction du contexte, elle se présentent, pour la pluspart, sous la forme tag-longueur-valeurs. C'est à dire qu'une fois les différents en-têtes enlevées et les informations fixes (dont la longueur est connue) écartées, le premier octet est un tag, c'est à dire un nombre identifant l'information qui suit, le deuxiéme octet nous donne la longueur de l'information, nottons la n, et les n octets suivants sont l'information. L'octet suivant est à nouveau un tag, suivi d'une longueur, ect.

Prenons comme exemple la figure 2.1, qui est un exemple de becon frame obtenu avec wireshark. Les octets surlignées et ceux qui les précédent constituent l'en-tête et ne nous interessent pas. Les octets encadré en jaune sont des informations "fixes", dans le sens ou elles sont toujours présentes dans une bea-

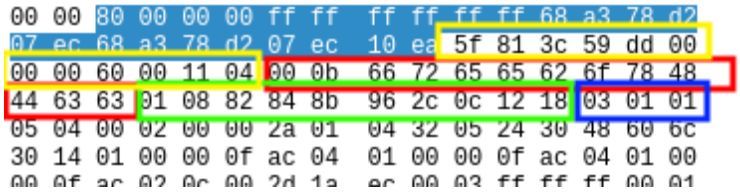

FIGURE 2.1 – Un exemple de beacon frame

con frame. Il s'agit d'un time stamp, du beacon interval [3] et quelques flags. Le reste du message illustre le fonctionnement décris plus haut. Ainsi, le premier octet dans le cadre rouge est 00, ce qui indique que l'information qui va suivre est le SSID [4]. L'octet suivant est 0b [5] indiquant que le SSID est donné sur 11 octets. Les 11 octets suivants donnent le SSID au format ASCII. Le tag dans le cadre vert est 01, indiquant que l'information à suivre est les débits supportées, et la longueur de ce champ est de 8 octets. Les 8 octets suivants indiquent donc chacun un débit. Dernier exemple, dans le cadre bleu, le tag est 03, indiquant que l'information à venir est le canal utilisé, la longueur de cet information est de 01 octet et l'information est 01, indiquant que ce point d'accés fonctionne sur le canal 1, à la fréquence 1412 [6].

Encore une fois, l'API libnl-genl nous fournis les outils pour demander au noyeau linux de faire un scan, via les commandes NL80211_CMD_GET_SCAN et NL80211_CMD_TRIGGER_SCAN à envoyer dans un message netlink. Cela nous permet de récupérer toutes les informations sur les différents réseaux contenues dans les beacon frames. Cependant, aprés une phase de tests, la conclusion empirique à été que ce scan ne détectait pas toujours tous les réseaux wifi mesh. Nous avons donc pris la décision, dans le but d'avoir les résultats les plus complets possibles, d'implémenter notre propre scanner pour exploiter nous même les beacon frames.

TCPdump est un programme généralement inclus dans les distribution linux permettant de "sniffer" des packets sur une interface réseau. Ce programme se base sur libpcap[12], une librarire permettant de capturer des paquets. C'est cette librarire que nous utiliserons. Elle permet de récupérer les messages tel qu'ils ont été capturées par l'interface. Il est ensuite possible d'effectuer toutes oppérations souhaités sur eux.

Pour être en mesure de capter les beacon frames, il est nécessaire régler une interface en mode monitor. De plus, puisqu'il existe 14 cannaux, il est nécessaire de passer de l'un à l'autre pour tous les scanner. Ces deux oppérations sont possibles avec la librairie libnl décrite chapitre 2, partie 1. La méthode que nous utiliserons est donc la suivante : nous paramétrons la carte réseau associé à une interface en mode monitor sur une fréquence, puis nous utilisons les fonctions de libpcap pour écouter sur l'interface pendant 3 secondes. Pendant ce temps, à chaque reception d'une trame, nous vérifions qu'il s'agisse d'une beacon frame en analysant son type. Si c'est le cas, on analyse chaque information qu'elle contient en recharchant les tags 0, 3, 113 et 114 indiquant respectivement le

3. L'intervale de temps entre l'envoie de deux beacon frames
4. nom du réseau wifi
5. Il s'agit d'une valeur en hexadecimal valant 11 en décimal
6. voir figure 1.1

SSID, le canal de fonctionnement, la configuration mesh et le MeshId. Bien sur, seuls les beacon frames indiquant un réseau mesh possédent les deux derniers tags. En outre, le tag 00 est toujours présent mais dans le cas d'une beacon frame indiquant un réseau mesh, sa longueur est de 0 bits, due au fait qu'un réseau mesh n'as pas de SSID.

Une fois le scan répété sur chacun des 14 cannaux, nous avons donc deux listes : Une liste des réseaux "normaux" et une liste des réseaux mesh. De plus, nous pouvons calculer simplement l'occcupation de chaque fréquence, et si besoin trouver la moins encombré. Cependant, comme évoqué précédemment, et comme le montre bien la figure 1.1, les cannaux se superposent. Il n'est pas rare qu'en écoutant sur un canal, on reçoive un message envoyé sur un autre canal voisin. Ainsi, lors de la selection de canal, il peut être judicieux d'exclure non seulement le dit canal, mais aussi ses voisins.

Chapitre 3

Adressage et routage

3.1 Adressage dans un réseau maillé

3.2 Routage

Plus précisément, le protocole HWMP dispose de deux modes de fonctionnement : “on demand” et “proactive tree building”. Le second nécessite qu’un noued soit désigné comme noued racine[10].

Avec le mode “on demand”, cahque fois qu’un noeud à besoin de conaitre le chemin vers un autre noeud, il envoie une paquet “route request (RREQ)” en broadcast, en identifiant le noeud de destination. Le paquet RREQ contient aussi un champ métrique initialisé à 0. Chaque noeud intermédiaire va recevoir le paquet RREQ, eventuellementen plusieurs exemplaire. Si le paquet RREQ à une métrique plus faible que celle déja connue, le noeud intermédiaire met a jour sa table de routage et le retransmet aprés avoir augmenté la metrique. Lorsque le paquet attein le noued de destination, ce dernier répond avec un paquet “Route Reply (RREP)” en unicast vers la source. Ainsi, tous les noeuds entre la destination et la source conaissent une route vers ces deux points [10].

Chapitre 4

Implémentation et architecture logicielle

Bibliographie

[1] Michel Terré. Wifi, mars 2007.
http://easytp.cnam.fr/terre/images/WiFi.pdf.

[2] Canaux et fréquences wifi 2.4 ghz et 5 ghz, janvier 2015.
https://infos-reseau.com/canaux-et-frequences-wifi-2-4-ghz-et-5-ghz/.

[3] F.Nolot. Les réseaux sans-fil : Ieee 802.11.
http://www.nolot.eu/Download/Cours/reseaux/m1info/ProtoAv-Cours7-Wifi.pdf.

[4] James F. Kurose ; Keith W. Ross. Ieee 802.11 lans, 1999.
https://www.net.t-labs.tu-berlin.de/teaching/computer_networking/05.07.htm.

[5] How 802.11 wireless works, Mars 2003.
https://technet.microsoft.com/en-us/library/cc757419(v=ws.10).aspx.

[6] Nicolas Darchis. 802.11 frames : A starter guide to learn wireless sniffer traces, octobre 2010.
http://www.nolot.eu/Download/Cours/reseaux/m1info/ProtoAv-Cours7-Wifi.pdf.

[7] Guido Hiertz ; Dee Denteneer ; Sebastian Max ; Rakesh Taori ; Javier Cardona ; Lars Berlemann ; Bernhard Walke. Ieee 802.11s : The wlan mesh standard, fevrier 2010.
http://ieeexplore.ieee.org/document/5416357/.

[8] Jerome Henry. 802.11s mesh networking, novembre 2011.
https://www.cwnp.com/uploads/802-11s_mesh_networking_v1-0.pdf.

[9] R. Droms. Dynamic host configuration protocol, mars 1997.
https://tools.ietf.org/html/rfc2131.

[10] Avinash Joshi ; Hrishikesh Gossain ; Jorjeta Jetcheva ; Malik Audeh ; Michael Bahr ; Jan Kruys ; Azman-Osman Lim ; Shah Rahman ; Joseph Kim ; Steve Conner ; Guenael Strutt ; Hang Liu ; Susan Hares. Hwmp protocol specification, novembre 2006.
doc. : IEEE 802.11-06/1778r1.

[11] Netlink protocol library suite (libnl).
http://www.infradead.org/~tgr/libnl/.

[12] Tcpdump et libpcap.
http://www.tcpdump.org/.