Alexandre Boeglin - a.boeglin@free.fr

IUP3

Projet 50h

IP et Mobilité au dessus de Bluetooth

Table des matières

Introduction

Bluetooth est un protocole de communication sans fil, dont le but est d'être un remplacement aux câbles qui relient les périphériques entre eux. Il a été conçu pour être résistant aux parasites, simple à implémenter, sécurisé, consommant peu d'énergie, et peu cher à fabriquer. Il peut être utilisé dans les domaines de la domotique, des télécommunications personnelles et de l'informatique mobile, ainsi que pour accéder à des réseaux informatiques sans fil.

C'est ce dernier point qui rentre dans le cadre de mon projet, dont le but est d'étudier les possibilités de raccordement d'un périphérique Bluetooth à un réseau IP, puis de mobilité entre différents points d'accès tout en gardant la connectivité IP.

Étude du protocole

Cette partie a consisté en l'étude des spécifications du protocole Bluetooth. J'ai tout d'abord étudié la structure du protocole, le rôle des différentes couches, puis quelques cas pratiques, comme la découverte des voisins (inquiry), la connexion de deux périphériques Bluetooth, la connexion d'un périphérique à un piconet existant, et enfin les possibilités d'utiliser IP au dessus de Bluetooth.

Structure en couches

La pile Bluetooth est structurée de la manière suivante :
- Application (utilisant un Profile Bluetooth)
- L2CAP (Logical Link Control and Adaptation Protocol)
- LMP (Link Manager Protocol)
- BaseBand
- Radio

Couche Radio

Bluetooth utilise la bande de fréquence ISM (Industrial, Scientific, Medical) située à 2,4GHz car elle est libre d'utilisation dans la majorité des pays. La norme définit 79 canaux espacés de 1Mhz chacun, à partir de 2402MHz (2402 -> 2480). Malheureusement, en France et en Japon, au moment de la rédaction des spécifications v1.1, seuls les 23 premiers canaux étaient libres. Le matériel fabriqué spécifiquement pour ces deux pays (respectant cette restriction) est donc incompatible avec le reste du parc mondial.

La modulation utilisée est une modulation en fréquence (dérivation positive = '1', dérivation négative = '0').

Sont également définies plusieurs classes d'émetteurs, ayant des restrictions en puissance, le maximum étant la classe 1 avec 100mW.

Couche BaseBand

Les périphériques Bluetooth s'organisent en piconets. Il y a un périphérique maître par piconet. Un piconet peut en plus regrouper jusqu'à 7 périphériques esclaves actifs (qui peuvent émettre sur le lien) et un grand nombre d'esclaves passifs. Un périphérique peut prendre part à plusieurs piconets, en tant que maître ou esclave (mais ne peut être maître que d'un seul piconet) pour former un scatternet entre ces piconets.

Dans Bluetooth, les communications se font uniquement entre maître et esclave. Si deux esclaves d'un piconet veulent communiquer, ils doivent créer un second piconet entre eux, à la manière d'un scatternet. Un périphérique peut choisir d'écouter alternativement les différents piconets auxquels il participe (un périphérique ne peut écouter qu'un seul canal à la fois).

Pour résister aux interférences provoquées par l'utilisation importante de la bande des 2,4GHz (WiFi, fours à micro-ondes...), Bluetooth utilise le Frequency Hopping : chaque paquet est envoyé sur un canal différent, et il y a 1600 sauts de fréquence par seconde. On parle de 1600 slots, chacun pouvant contenir un paquet, et durant 625µs. La séquence des canaux utilisés lors de ces sauts est connue de tous les périphérique, car elle est dérivée de l'adresse MAC du maître. La phase (position courante dans la séquence) est envoyée par le maître aux esclaves lors de leur connexion, garantissant que tous écoutent le même canal à un instant donné.

A ce niveau de la pile, le débit disponible est de 1Mb/s.

Bluetooth définit aussi des paquets plus longs, pouvant utiliser 3 ou 5 slots. Dans ce cas, les 3 ou 5 slots sont transmis sur le même canal, sans saut, et le paquet suivant sera émis sur le canal correspondant à une avancée de 4 (3+1) ou 6 (5+1) dans la séquence des sauts, ce qui évite une désynchronisation avec les périphériques n'ayant pas écouté le réseau à ce moment.

Le canal de communication créé par cette suite de slots est découpé en deux : les slots paires sont réservés aux transmissions du maître, et les slots impaires sont réservés aux réponses des esclaves (la parité est déterminée par la phase de la séquence de saut, qui est un compteur). Un esclave ne peut transmettre sur le canal que lorsque le maître lui a envoyé un paquet dans le slot précédent.

Types de liens

Deux type de liens sont définis dans cette couche :

- **Lien SCO** (Synchronous, Connection Oriented)

Ce type de lien, orienté connexion, est utilisé pour des communications de type voix à 64kb/s. L'établissement de ce lien provoque une réservation de slots, afin de s'assurer que le débit peut être soutenu tout en garantissant une faible gigue. C'est une connexion point à point entre le maître et un esclave. Un périphérique peut supporter jusqu'à trois connexions SCO.

- **Lien ACL** (Asynchronous, Connection Less)

Il y a un seul lien ACL par piconet, il utilise les slots qui ne sont pas réservés. Ce lien est de type non connecté, il fait de la commutation de paquets. Les paquets dont l'adresse vaut zéro sont des paquets broadcast, tous les esclaves le remontent à la couche supérieure, mais il n'est pas acquitté.

Format d'un paquet

Les paquets sont constitués de trois éléments : un code d'accès, une en-tête et la charge utile. Un paquet peut aussi n'être composé que d'un code d'accès, ou d'un code d'accès et d'une en-tête mais sans charge utile.

- **CODE D'ACCÈS** : dans le cas d'une communication dans un piconet, le code d'accès est dérivé de l'adresse MAC du maître du piconet. Dans le cas d'une inquiry (découverte des voisins) c'est un numéro réservé qui est utilisé et dans le cas d'un paging (établissement de connexion), c'est l'adresse MAC de la cible qui est utilisée. Le code est composé de 64 bits, il est précédé de 4 bits qui sont alternés (0101 si le code commence par un 0 et 1010 si il commence par un 1); si le code d'accès est suivi d'une en-tête, il est aussi suivi de 4 bits alternés (en fonction du dernier bit du code d'accès). Ces bits permettent aux périphériques de se synchroniser sur l'émetteur lors de la réception du paquet.
- **EN-TÊTE** : elle est composée de 54 bits, chaque bit étant répété trois fois, dans un but de détection/correction d'erreur, il y a 18 bits utiles :
  - Adresse de membre actif (3 bits) : chaque membre actif du piconet se voit attribué un numéro unique, de 1 à 7; le zéro étant réservé au broadcast.
  - Type (4 bits) : la norme définit 14 types de paquets différents, ce champ sert à interpréter la charge utile du paquet.
  - Flux (1 bit) : quand un membre du piconet envoie un paquet avec ce bit à 1, les communications ACL sont stoppées jusqu'à ce que ce membre renvoie un paquet avec un bit de flux à 0.
  - ARQN (1 bit) : ce champ sert à l'acquittement (1=ACK du dernier paquet; 0=NAK du dernier paquet, qui est donc réémis).
  - SEQN (1 bit) : ce champ est le numéro de séquence. Si il a la même valeur que lors du dernier paquet, cela signifie que le paquet courant est une réémission, sinon, ce bit est alterné à chaque nouveau paquet.
  - HEC (8 bits) : ce champ est un code correcteur polynomial sur l'en-tête.
- **CHARGE UTILE** : Elle peut faire jusqu'à 2745 bits, ce sont les données remontées à la couche supérieure.

Types de paquets

Bluetooth définit plusieurs types de paquets. Je ne me suis pas intéressé ici aux paquets utilisés dans les communications SCO.

- Paquet **ID** : ce paquet est constitué uniquement d'un code d'accès, il est utilisé dans les processus d'inquiry et de paging, afin de contacter un voisin.
- Paquet **NULL** : ce paquet n'a pas de charge utile, il est utilisé lorsqu'un

périphérique doit envoyer un acquittement et n'a pas d'informations à transmettre.
- Paquet **POLL** : ce paquet est similaire au paquet NULL, mais il est utilisé par le maître pour donner la parole à l'esclave désigné dans l'en-tête.
- Paquet **FHS** : c'est un paquet de synchronisation qui contient les informations nécessaires aux esclaves pour se synchroniser au maître. Il est envoyé à l'établissement de la connexion, puis périodiquement.
- Paquet **DM**x (x=1, 3 ou 5) : c'est un paquet de données (Data Medium rate) utilisant un code correcteur à 2/3 (un tiers des bits émis est utilisé par le code correcteur). Il peut avoir une longueur de 1, 3 ou 5 slots.
- Paquet **DHx** (x=1, 3 ou 5) : c'est un paquet de données (Data High rate) sans correction d'erreur, utilisant juste un CRC sur 16 bits. Il peut avoir une longueur de 1, 3 ou 5 slots.

Note : le type de paquet DM1 est utilisé pour les messages de gestion du lien.

Sécurité

Pour l'établissement d'une connexion sécurisée, des deux parties doivent posséder le même secret : le code PIN (de 8 à 128 bits). Un premier challenge est envoyé, utilisant une clé d'initialisation dérivée du code PIN, d'un nombre aléatoire et de l'adresse du périphérique appelant.

Si le challenge est "relevé", cette clé d'initialisation est ensuite utilisée pour échanger la clé de lien (Link Key), qui est dérivée d'une clé privée (Unit Key) de 128 bits contenue dans chaque périphérique (clé générée à la première mise sous tension du périphérique Bluetooth, puis mémorisée en ROM). Cette Link Key est propre à la connexion entre les deux périphériques et elle sert de secret partagé pour la génération d'une clé qui sera utilisée pour chiffrer les données transmises.

Couche LMP

Le premier octet de la charge utile comprend un numero de canal logique sur deux bits qui sert à définit si le paquet est de type données ou de type gestion du lien; et un code d'opération, qui sert à identifier les différentes procédures que définit LMP.

En résumé, LMP (protocole de gestion de lien) définit les différents messages que s'envoient les périphériques lorsqu'ils effectuent les actions relatives à l'utilisation du canal logique.

La majorité de ces actions peuvent être initiées soit par le maître, soit par l'esclave.

Ces actions sont :

- Authentification : authentification par challenge (A envoie un nombre aléatoire à B, B chiffre ce nombre avec sa clé et le renvoie à A, puis A compare ce que lui a envoyé B avec le résultat du chiffrement du nombre avec sa propre clé).
- Pairing : procédure permettant de génerer la Link Key.
- Changement de la Link Key.

- Chiffrement : les messages permettent aux périphériques de négocier le type de chiffrement utilisé, et de commencer et arrêter le chiffrement des données.
- Fonctionnalités supportées : permet aux périphériques de s'informer des fonctionnalités supportées par leur voisins (exemple : types de paquets).
- Role Switch : permet à un maître et un esclave d'échanger leur rôles.
- Requête du Nom : permet d'obtenir le nom d'un périphérique.
- Déconnexion.
- Passage en Hold Mode : Quand le lien ACL passe en Hold Mode, le maître cesse d'émettre. Ce peut être utilisé dans un soucis d'économie d'énergie, ou encore pour permettre à un périphérique de faire autre chose (participer à un second piconet, passer en mode d'attente d'inquiry ou de paging).
- Passage en Sniff Mode : quand un esclave passe en Sniff Mode, il reste actif mais n'écoute plus le lien qu'a intervalles réguliers, pour économiser son énergie.
- Passage en Park Mode : quand un esclave passe en Park Mode, il devient inactif, libérant son adresse de membre du piconet. Il peut être réveillé ou demander à sortir du park mode en utilisant un canal secondaire de balise que le maître et les esclaves parkés vont écouter à intervalles réguliers.
- Négociation de la puissance, du type de paquets à utiliser ou d'une qualité de service.
- Établissement d'un lien SCO.
- Avertissement de la perte du lien (paquet généré localement et remonté à l'application).
- Paging : permet à deux périphériques de se synchroniser en vue d'établir une connexion.
- Établissement de la connexion ACL.

Couche L2CAP

Un paquet L2CAP rajoute un Overhead comprenant la longueur des données en octets ainsi qu'un numéro de canal.

Les fonctions principales de L2CAP sont :

- **Multiplexage** : L2CAP permet à plusieurs application d'utiliser le même lien ACL entre deux périphériques, en utilisant des numéros de canaux L2CAP différents.
- **Segmentation et réassemblage** : L2CAP permet de remonter des paquets ayant jusqu'à 64kb de données alors que la charge utile maximale d'un paquet au niveau BaseBand est de 2745 bits.
- **Qualité de Service** : L2CAP permet aux applications de demander que certains paramètres soient respectés, comme la bande passante ou la gigue.

L2CAP est donc un protocole dont le rôle est de fournir une couche réseau complète aux couches supérieures et aux applications.

Profiles

Les profiles servent à garantir la compatibilité de tous les périphériques Bluetooth en décrivant de manière précise les méthodes à employer pour les différents domaines

auxquels est destiné Bluetooth (établissement d'un lien série, d'une connexion réseau, téléphonie sur bluetooth, envoi de fichiers et synchronisation de périphériques en utilisant OBEX...).

Un profile particulier est le Service Discovery Protocol, qui permet à un périphérique de demander à un voisin si il fournit un service partuiculier.

Applications

Découverte des voisins

La découverte des voisins, ou Inquiry, se déroule sur 16 fréquences désignées. Elle se déroule en deux parties :

- Côté appelant : le périphérique envoie un paquet ID et enregistre les réponses reçues sur chacune des 16 fréquences.
- Coté appelé : chaque périphérique peut choisir de passer en mode Inquiry Scan (attente de requête) à intervalles réguliers. Il choisit une fréquence parmi les 16 disponibles, puis lorsqu'il reçoit un paquet ID, il attend pendant une durée choisie aléatoirement, pour éviter les collisions entre plusieurs réponses, puis envoie un paquet FHS (de synchronisation).

Il existe deux types de paquets ID : un ID général, auquel tous les périphériques répondent, et des ID dédiés, correspondants à différents types de périphériques, auxquels seuls les périphériques du même type répondent.

Connexion de deux périphériques Bluetooth

Après l'Inquiry, le périphérique appelant, qui sera le maître du piconet qui va être crée, envoie un "Page" au périphérique auquel il souhaite se connecter. Le Paging se déroule sur deux plages de 16 fréquences dédiées. On peut déterminer sur laquelle des deux plages écoutera un périphérique à partir du paquet FHS reçu. La séquence est la suivante :

- Le maître envoie un paquet ID, qui contient l'adresse MAC de la cible sur chacune des 16 fréquences de la première plage, et s'il n'a pas de réponse, il réémet ce paquet sur la seconde plage.
- L'esclave passe à intervalles réguliers en mode Page Scan et attend un paquet ID contenant son adresse MAC. À réception de ce paquet, il émet un paquet ID contenant sa propre adresse MAC à la fréquence correspondant au slot suivant.
- Lorsque le maître reçoit ce paquet, il sait à quelle fréquence écoutait l'esclave, et il peut en déduire à quelle fréquence il doit lui envoyer le paquet suivant, qui est un paquet FHS contenant ses informations d'horloge.
- L'esclave utilise ce paquet pour déterminer l'adresse MAC du maître, le code d'accès du piconet qui vient d'être créé, son adresse de membre actif, la séquence de sauts, ainsi que la position courante dans cette séquence. Il envoie ensuite un

paquet ID contenant son adresse MAC pour acquitter le paquet FHS.
- Le paquet suivant utilise enfin le code d'accès et la séquence de sauts du piconet, et c'est un paquet de type POLL envoyé par le maître à l'esclave.
- L'esclave répond au maître en acquittant son paquet POLL.

Le Paging est maintenant terminé, les périphériques sont prêts à établir un lien en utilisant LMP.

Connexion d'un périphérique à un piconet existant

Comme nous l'avons vu avant, lorsqu'un périphérique initie une connexion, il devient le maître du piconet ainsi créé. Imaginons un périphérique 'a' qui souhaite se connecter à 'B', 'B' étant maître d'un piconet. Si 'a' initie une connexion à 'B', on obtient deux piconets : celui de 'a', dont 'B' est esclave, et celui de 'B'. Cette situation occasionne une perte importante : le piconet de 'B' est indisponible lorsque ce dernier discute sur le piconet de 'a'. Il est alors possible à 'B' de demander un Role Switch à 'a' et d'éliminer ainsi le partage de ses slots entre les deux piconets : 'a' et 'B' échangent de rôle, et 'a' rejoint donc le piconet de 'B' en tant qu'esclave. Il n'y a alors plus qu'un seul piconet.

Étant donné que les communications se font entre maître et esclave, et pas entre deux esclaves, on peut imaginer le cas d'un point d'accès à un réseau et de trois clients : il vaut mieux que l'AP soit maître d'un piconet ayant trois esclaves, plutôt que d'avoir trois piconets ayant chacun l'AP pour seul esclave et un des clients pour maître : l'AP n'aura pas à gérer trois liens ACL, ni le fait de devoir partager son temps entre les trois piconets.

Utilisation d'IP au dessus de Bluetooth

Il existe actuellement deux Profiles Bluetooth permettant un accès à un réseau IP :

- RFCOMM est une émulation de port série au dessus de Bluetooth, il est possible d'établir une liaison ppp au dessus d'un port RFCOMM. Cette solution n'est pas la plus élégante, du fait des contraintes qu'impose ppp, mais c'est la solution historique : la seule disponible sur la plupart des systèmes anciens, et des systèmes propriétaires.
- BNEP est une solution plus récente, qui offre un moyen de faire passer plusieurs protocoles réseaux, dont IPv4 et IPv6 directement sur L2CAP. Malheureusement, du fait de son âge plus jeune, elle n'est pas présente sur tous les systèmes équipés de Bluetooth.

Étude pratique

Cette étude a consisté en plusieurs points :
- La configuration d'un périphérique Bluetooth (iPAQ tournant sous GNU/Linux).
- La configuration d'un point d'accès utilisant RFCOMM et la connexion à celui-ci.
- L'installation d'un point d'accès utilisant BNEP et la connexion à celui-ci.
- Le test de la capacité de ces deux méthodes à tenir un certain débit.
- L'étude des possibilités de handover entre plusieurs points d'accès Bluetooth.

Configuration de l'iPAQ

Pour cette étape, le site handhelds.org et plus particulièrement le wiki sont des aides précieuses.

Installation de la distribution

- Prérequis : Tout d'abord, il m'a fallu installer la dernière version de la distribution Familiar GNU/Linux. Pour ce faire, j'ai connecté l'iPAQ dans son cradle (son socle) à un PC via le port série. J'ai donc installé les packages debian minicom (émulateur de terminal) et lrzsz (qui contient les commandes d'envoi et de réception de fichiers en utilisant le protocole xmodem). J'ai ensuite lancé 'minicom -o' (l'option -o servant à éviter qu'il n'envoie un ATZ à l'iPAQ). J'ai configuré minicom (touches Ctrl+A puis O) pour qu'il communique avec le port série /dev/ttyS0 à 115200 bauds, 8N1 et contrôles de flux désactivés. Après avoir redémarré minicom une fois la config enregistrée, j'ai rebooté l'iPAQ et je l'ai forcé à rester en mode 'serial console'.

- Bootloader (2.18.54) : J'ai changé le bootloader qui était trop vieux, et incompatible avec les dernières modifications apportées à la Familiar. J'ai donc tapé, dans minicom, sur le prompt du bootloader : 'load bootldr'. Il demande ensuite un fichier, qu'il faut lui envoyer en xmodem (Ctrl+A puis S). Une fois le fichier reçu, il est copié dans la ROM. Si tout s'est bien passé, et _seulement_ dans ce cas, on peut rebooter l'iPAQ afin de charger le nouveau bootloader, et repasser en mode 'serial console'.

- Familiar v0.7-pre6 (fichier image : bootgpe2-v0.7-pre6-h3600.jffs2) : Pour l'installation de la Familiar, il faut ensuite préparer la ROM à accueillir le système en tapant 'partition reset'; on peut enfin faire un 'load root' puis envoyer le fichier .jffs2 en xmodem. Trente minutes après, le transfert se termine et le fichier est copié en ROM. On peut enfin taper 'boot' pour amorcer le système.

Configuration post-installation

Pour pouvoir mettre le système à jour, il faudra tout d'abord établir une connexion réseau, soit en utilisant une carte réseau PCMCIA/CompactFlash, soit en utilisant le

module USB Networking. Le port série peut aussi être utilisé comme un terminal. Un point à retenir est que lors de la connexion Bluetooth, il faudra au préalable détruire la route par défaut créée lors de ce premier établissement de connexion réseau.

- USB Networking : Cette démarche est complètement détaillée sur le wiki de handhelds.org, il y a un script à lancer sur l'iPAQ et un autre à copier et lancer sur la station, qui active le forwarding et met en place un NAT. Le seul problème est que le noyau de la station ne peut être un 2.4.18, ce dernier ayant un bug au niveau du module usbnet. Pour que la connexion s'établisse normalement, il faut : lancer le script sur l'iPAQ, débrancher et rebrancher l'iPAQ de son cradle, puis monter l'interface (ou relancer le script) sur la station.

Installation des paquets

Avant tout, il faut vérifier que le fichier /etc/ipkg.conf contient bien ces deux lignes :
src base http://familiar.handhelds.org/releases/v0.7/base/armv4l
src x http://familiar.handhelds.org/releases/v0.7/x/armv4l

Il faut aussi commenter (par un #) la ligne suivante si elle est présente :
src unstable http://familiar.handhelds.org/feeds/unstable

On peut ensuite mettre le système à jour en faisant un 'ipkg update' suivi d'un 'ipkg upgrade'.

Note : je donne ici les noms de packages complets, tels qu'ils existent lors de l'écriture de ce rapport. Pour savoir avec quel nom de package invoquer 'ipkg install', il faut utiliser la fonction 'ipkg list'.

- Paquets à installer en v0.7 :
  - Task-bluez : c'est un meta-package qui contient la majorité de ce qui est nécessaire à Bluez.
  - Bluez-pin2 : c'est une application qui permet d'entrer un code PIN lorsque c'est nécessaire. Cette version est écrite en C , la commande par défaut, étant en python, m'a posé des problèmes d'exécution. (il faut supprimer ou déplacer /bin/bluepin avant pour que l'installation se déroule bien)
  - Bluetooth-uart-modules_2.4.19-rmk6-pxa1-hh9_ipaqsa : ce package contient le driver du chip Bluetooth intégré à l'iPAQ.
  - Bluetooth-modules_2.4.19-rmk6-pxa1-hh9_ipaqsa : ce package contient les modules nécessaires à Bluez.
  - Ssh : nécessaire pour repasser par la suite à un noyau 2.4.18, et permet le scp.
  - Ipv6-modules_2.4.19-rmk6-pxa1-hh9_ipaqsa : ce package contient le module ipv6. Attention : la commande ifconfig installée sur l'iPAQ n'est pas compatible ipv6 et n'affichera pas les adresses des interfaces. Il faut recompiler ifconfig ou utiliser celui venant du package debian pour ARM.

Il faut ensuite passer en unstable : décommenter ou ajouter la ligne 'src unstable http://familiar.handhelds.org/feeds/unstable' à /etc/ipkg.conf, puis relancer un 'ipkg update'.

Note : le problème semble réglé depuis, mais à l'époque de mes tests, les modules

bluez du 2.4.19 étaient défectueux. Si tel est le cas, il faut faire un 'ipkg install task-2.4.18 bluetooth-modules-2.4.18' suivi d'un 'ipkg remove kernel-image-2.4.19' puis rebooter.

- Paquets à installer en unstable :
  - Bluez-rfcomm : permet de se connecter à un point d'accès utilisant RFCOMM.
  - Il peut être aussi nécessaire ou simplement intéressant de faire un 'ipkg upgrade' afin de passer tout le système en unstable. Malheureusement, cela pourra créer des problèmes de dépendance entre des packages, si certains sont en retard par rapport aux autres.

Configuration de Bluetooth

Pour pouvoir utiliser Bluetooth de manière transparente sur l'iPAQ, il faut encore faire quelques réglages, résumés sur la page 'Bluetooth Howto' du wiki de handhelds.org :
- Dans /etc/modules.conf, vérifier que des alias ont été créés pour bluez, hci_uart et l2cap.
- Dans /etc/modules, vérifier que 'bluez', 'l2cap', 'hci_uart', 'bnep' et 'ipv6' apparaissent, chacun sur une ligne.
- Dans /etc/init.d, il faut deux scripts : 'bluez' et 'hcid' avec des liens dans /etc/rc2.d :
  - Le premier est chargé d'attacher le chip Bluetooth, avec la commande 'hciattach /dev/ttySB0 bcsp 230400' (le paramètre 230400 est à ajouter); ce script est disponible sur la page 'Bluetooth Howto'.
  - Le second est chargé de lancer hcid, le démon qui gère les interactions entre la partie matérielle et la partie logicielle de la pile Bluetooth; ce script a dû être installé lors de l'installation du package task-bluez.

Une fois ces scripts lancés, il est possible d'utiliser des commandes telles 'hciconfig' pour vérifier l'état du chip Bluetooth et 'hcitool scan' pour découvrir les périphériques Bluetooth se trouvant à proximité.

Configuration de l'AP AXIS et connexion en RFCOMM

La première chose à faire, après avoir effectué le reset de l'AP, est de le connecter directement à une station avec un câble croisé, et d'ajouter une entrée ARP statique à la station, avec l'adresse IP que l'on souhaite attribuer à l'AP. Après avoir lancé un navigateur web, on peut accéder au site d'administration disponible sur le port 80 pour fixer les paramètres de configuration de l'AP, qui pourra ensuite être branché n'importe où dans le réseau.

Ensuite, du coté du client, il y a juste deux commandes à entrer :
- 'rfcomm conn 0 <addresse de l'AP>' qui permet de se connecter à l'AP par Bluetooth (le code PIN est demandé à ce moment) et d'attacher le port série /dev/bluetoth/rfcomm/0 au lien Bluetooth qui vient d'être créé.
- 'pppd /dev/bluetooth/rfcomm/0 noauth' qui permet d'établir une connexion IP point à point entre les deux périphériques.

Installation d'un AP Bluetooth utilisant BNEP

Concernant cette installation, plusieurs documentations sont disponibles : http://bluez.sourceforge.net/contrib/HOWTO-PAN et dans le cas d'une carte 3com : http://www.holtmann.org/linux/bluetooth/bt3c.html .

La première étape est de compiler ou d'installer les paquets des applications suivantes :

- Un noyau avec support de Bluetooth, ipv6, 802.1d et PCMCIA.
- Le service pcmcia-cs.
- L'ensemble du projet Bluez (libs, utils, sdp, pan et blueFW)

Une fois que tout est en place, il faut que les démons pcmcia-cs, hcid, sdpd et pand soient lancés au démarrage, et que les modules bluez, l2cap, bnep et ipv6 soient chargés.

Pour que la carte 3com soit reconnue, il faut que le firmware soit extrait des drivers windows comme indiqué dans le second lien ci-dessus.

Le code pin que l'AP utilisera est à spécifier dans /etc/bluetooth/pin. La configuration Bluetooth se fait dans /etc/bluetooth/hcid.conf : il faut changer la ligne 'lm accept' en 'lm accept, master' pour que l'AP switch en master, et qu'il n'y ait pas un piconet par client connecté. Il faut aussi décommenter la ligne 'auth on' pour activer l'authentification par code PIN.

Sdpd doit être lancé avant Pand, pour que ce dernier puisse s'enregistrer auprès du démon SDP. Pand peut ensuite être appelé avec les options suivantes 'pand - -listen - -role NAP' pour qu'il se lance en mode Access Point et qu'il s'enregistre corrrectement.

Pour que l'AP agisse en tant que bridge, il faut créer un bridge entre toutes les interfaces, comme ceci :

```
brctl add pan0
brctl stp pan0 off                  #désactive le Spanning Tree
brctl setfd pan0 0                  #délai de forwarding nul
brctl addif eth0
ifconfig eth0 0.0.0.0               #passe eth0 en promiscuous
ifconfig pan0 130.79.X.Y            #attribue une IP à l'AP
route add default gw 130.79.X.Y     #définit la route par défaut
```

Pour ajouter les interfaces virtuelles crées par Bluetooth lors de la connexion de clients, il faut ajouter un script : /etc/bluetooth/pan/dev-up ; ce script est appelé par le démon pand avec le nom de l'interface en premier argument :

```
#!/bin/bash
brctl add pan0 $1
ifconfig $1 0.0.0.0
```

Le point d'accès est maintenant prêt à accepter des connexions Bluetooth. Si aucun serveur DHCP n'est présent sur le réseau local, il faudra aussi attribuer une adresse IP au client et lui indiquer la route par défaut (la même que l'AP).

Connexion à l'AP BNEP

Pour se connecter de manière statique (une seule connexion au périphérique spécifié, comme en RFCOMM) à l'AP, il suffit de taper 'pand -c <adresse de l'AP>' sur le client, d'entrer le code PIN, et de configurer l'interface bnep0. La configuration de l'interface peut être automatisée via la fonction hotplug du kernel. Il suffit de rajouter une ligne 'HOTPLUG_NET_DEFAULT_HARDWARE=net' dans /etc/sysconfig/hotplug , et de rajouter une entrée pour bnep0 dans /etc/network/interfaces. Cela permettra à l'interface d'être montée automatiquement à chaque détection d'établissement de lien.

Mesure de qualité des liens RFCOMM et BNEP

Ces mesures ont été effectuées à l'aide de mgen, un générateur de trafic, employé ici pour envoyer un certain nombre de paquets par seconde, ces paquets ayant une taille définie. Trpr est ensuite utilisé pour en extraire un graphe exploitable par gnuplot.

Du côté de l'émetteur, il faut créer un script pour mgen, afin de déterminer le trafic généré :

```
0.0 ON 1 UDP SRC 5000 DST 130.79.X.Y/5001 PERIODIC[100 1000]
0.1 OFF 1
```

Ce script enverra 100 paquets UDP de 1000 octets par seconde pendant 60 secondes depuis le port 5000 vers le port 5001 de 130.79.X.Y.

il faut ensuite appeler mgen sur l'émetteur : 'mgen ipv4 input <fichier script>' pour effectivement envoyer ce trafic transporté dans ipv4 par exemple.

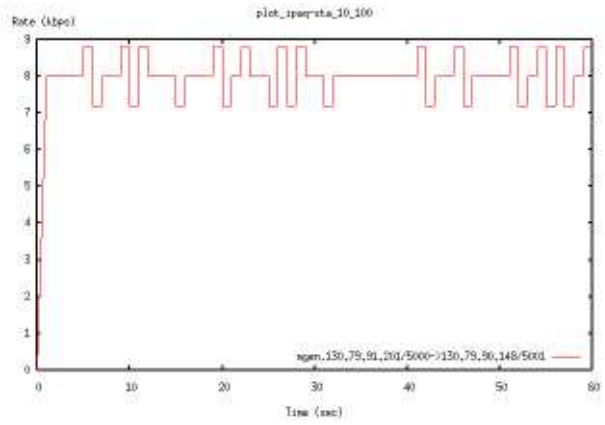
Avant de lancer le générateur de trafic, il faut appeler mgen sur le destinataire pour qu'il génère un log : 'mgen ipv4 port 5001 output <fichier log>'. On peut arrêter cet absorbeur de trafic à la fin de l'envoi par un Ctrl+C.

On peut ensuite tirer un graphe par un 'trpr mgen auto X input <fichier log> output <fichier plot>'.

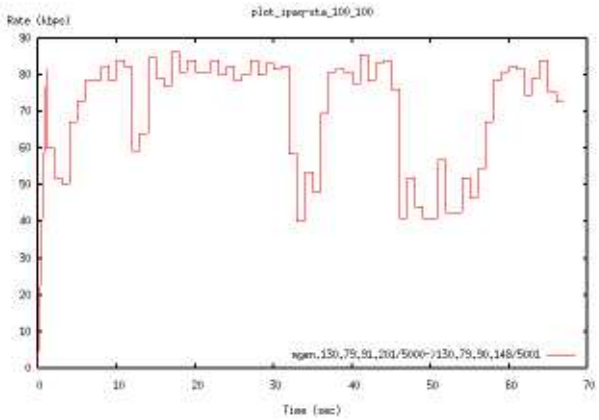
Puis l'afficher grâce à un 'gnuplot -persist <fichier plot>'.

Avec RFCOMM

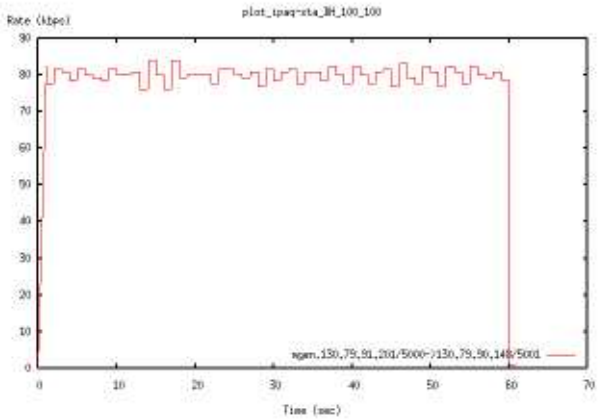
En envoyant d'abord 10 paquets de 100 octets par seconde, on observe que la liaison tient le coup, et on atteint les 8kb/s (soit 1ko/s).



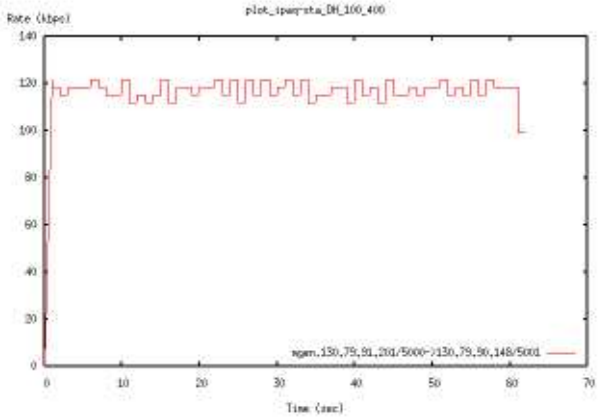
Lorsque l'on envoie 100 paquets de 100 octets par seconde, le lien a du mal à assurer le débit.



Après recherches, il s'agit du type de paquets employés qui pose problème : le type employé par défaut est DM1, qui permet un débit théorique maximum de 108kb/s. Forçons le type de paquet à DH5 (qui permet un débit théorique de 723kb/s en liaison asymétrique) avec la commande 'hcitool cpt <adresse de l'AP> DH5' (ce paramètre est aussi réglable dans le fichier hcid.conf). On observe maintenant un débit correct de 80kb/s.

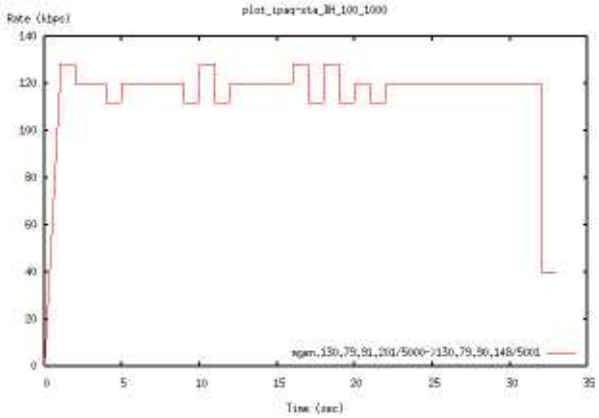


Lorsqu'on monte encore le débit à 100 paquets de 400 octets par seconde, on s'aperçoit que le lien sature à environ 120kb/s; cette vitesse correspond en fait à la vitesse d'horloge de la liaison avec le chip Bluetooth dans l'iPAQ.



Lorsque l'on augmente encore (100 paquets de 1000 octets) on observe que la connexion ppp se désynchronise et la connexion IP est donc perdu après environ 30 secondes. La liaison RFCOMM quant à elle est toujours en place, il faut uniquement

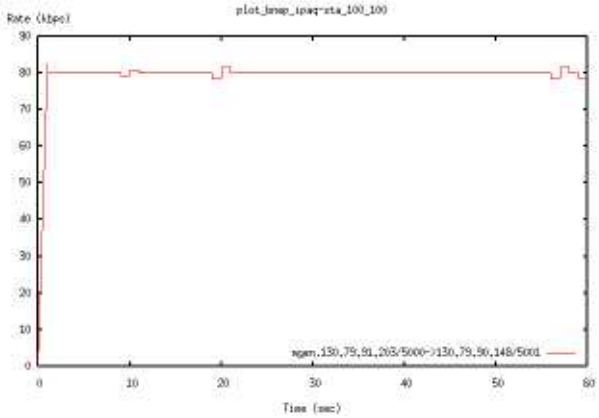
relancer pppd.



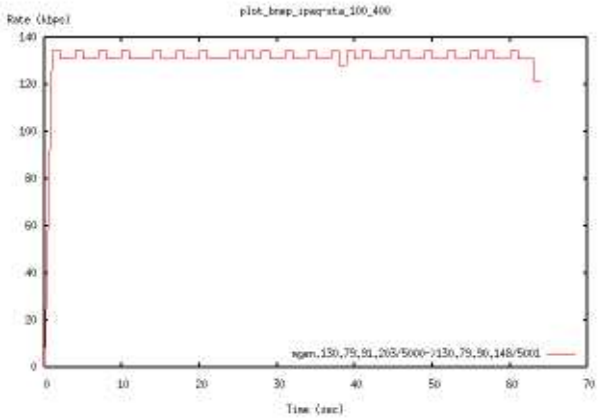
Avec BNEP

J'ai ici refait les mêmes tests, en n'utilisant plus que des paquets de type DH5.

Tout d'abord, pour 100 paquets de 100 octets par seconde, on observe que le débit est correct; la courbe est aussi plus 'lisse' qu'avec RFCOMM pour le même débit.

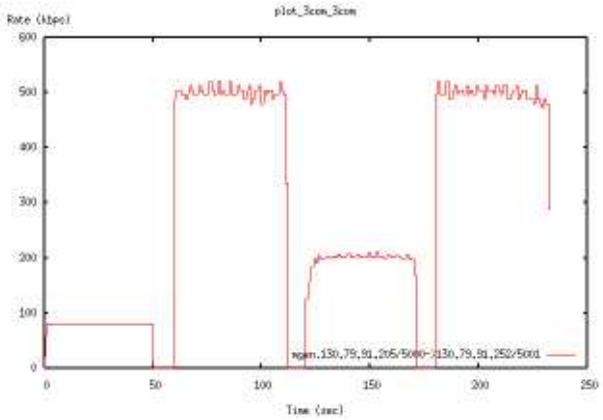


En montant à 100 paquets de 400 octets par seconde et plus, on observe à nouveau que la liaison sature à 130kb/s, mais la connexion n'est jamais désynchronisée ici et la connectivité IP reste disponible.



Pour voir quelles étaient les réelles performances de Bluetooth, j'ai ensuite effectué un test entre deux machines disposant d'une carte 3com. J'ai envoyé plusieurs salves de

paquets à différents débits, et la liaison sature ici à 500kb/s, ce qui est acceptable, étant donné le débit théorique d'une liaison DH5, soit 723kb/s au niveau de la couche 2.



Il apparaît clairement lors de ces tests que BNEP est meilleur en termes de qualité, tout en ne 'décrochant' pas lorsque le lien est saturé

Mobilité avec BNEP

Pour pouvoir mettre en place la mobilité, qui ici consiste en la recherche d'un point d'accès auquel se reconnecter en cas de déconnexion, il faut avant tout que les points d'accès utilisent sdpd, le démon chargé de répondre aux requêtes SDP. Sur le client, il est possible de lancer une recherche de service avec sdptool, par exemple la commande 'sdptool search NAP' permet de connaître les périphériques Bluetooth proches qui proposent un service de point d'accès BNEP.

Il est ensuite très recommandé d'utiliser les capacités hotplug sur le client, pour que la reconnexion soit vraiment 'seamless'.

On peut ensuite lancer pand sur le client en mode persistant : 'pand - -search - -role PANU - -service NAP - -persist'. Cette commande fait un inquiry, envoie une requête SDP à chaque périphérique découvert et tente de se connecter à ceux qui proposent le service PAN à chaque fois qu'une connexion n'est pas déjà active, en mémorisant les résultats obtenus afin d'éviter de refaire plusieurs les mêmes requêtes.

Mesure les performances lors d'un Handover

J'ai fait ici deux tests :

- Handover : pour ce test, l'AP auquel est accroché le périphérique est débranché.
- Déconnexion manuelle : ici, c'est uniquement le lien qui est arrêté avec la commande 'pand -K'.

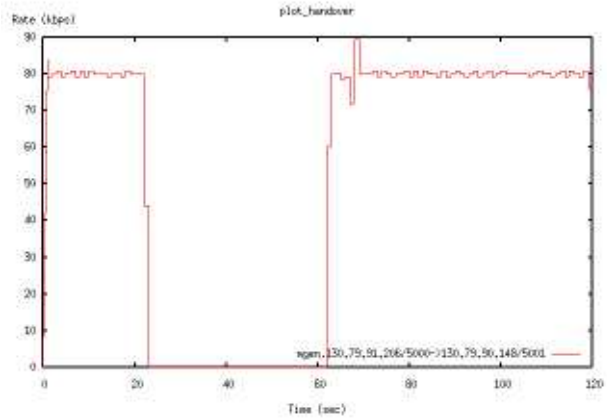
J'ai en outre observé qu'à chaque reconnexion, l'interface bnep0 est détruite et recréée, les communication TCP perdent donc leur socket, ce qui pose problème.

Handover

Ici, lorsque l'AP auquel le client est accroché est déconecté, on observe ceci :
- 20 secondes de timeout après le dernier paquet reçu, après ces 20 secondes, la liaison est déclarée coupée par le chip Bluetooth.
- 13 secondes qui correspondent à un inquiry permettant de découvrir les périphériques proches.
- 2 secondes par requête SDP pour savoir si le périphérique concerné propose le bon service.

Si le périphérique propose le service NAP, on tente une connexion à ce dernier, et l'interface est remontée; sinon, on passe au périphérique suivant découvert.

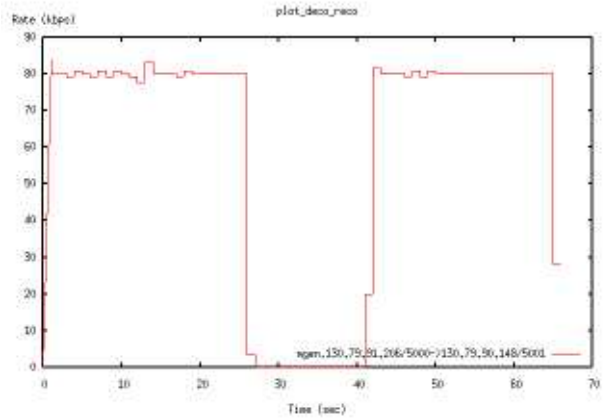
On peut observer cette durée sur le graphique suivant :



La durée de 'blackout' est un peu plus longue, car le bridge sur l'AP met environ 6 secondes à se 'stabiliser' avant de commencer à forwarder les trames.

Déconnexion Manuelle

Ici, lorsque l'on force la déconnexion, on peut s'affranchir des 20 secondes de timeout et on passe directement à l'inquiry ou à la requête SDP. On observe un temps de reconnexion de 9 secondes auxquelles s'ajoutent les 6 secondes nécessaires au bridge pour rétablir la liaison.



Liens

Debian GNU/Linux : http://www.debian.org

La communauté des utilisateurs d'iPAQ sous Linux : http://www.handhelds.org
La distribution Familiar : http://familiar.handhelds.org

Les spécifications du protocole Bluetooth : http://www.bluetooth.org

Le résumé du protocole par Aman Kansal : http://www.ee.iitb.ernet.in/uma/~aman/

La pile Bluetooth pour linux Bluez : http://bluez.sourceforge.net
Les modules Bluez pour le noyau : http://www.holtmann.org/linux/kernel/
Les drivers pour les cartes 3com : http://www.holtmann.org/linux/bluetooth/bt3c.html

Mgen : http://manimac.itd.nrl.navy.mil/MGEN/
Trpr : http://proteantools.pf.itd.nrl.navy.mil/trpr.html

Conclusion

La solution présentée ici n'est pas complète, dans le sens où une reconnexion coupe l'interface réseau et détruit en même temps les sockets. Une solution qui permettrait de remédier à ce problème serait de mettre en place un bridge au niveau du client, qui agirait comme une interface virtuelle placée au dessus de l'interface crée par BNEP, et qui ne subirait donc pas ces déconnexions.

Une solution plus propre existera à l'avenir, nommée APR (PAN Access Point Roaming), dont le rôle sera de gérer le handover entre plusieurs Acces Points; mais à l'heure actuelle, le groupe de travail chargé de ce profile Bluetooth n'a pas encore publié de spécifications.