SUP'COM

PROJET DE FIN D'ETUDES

Filière : Télécommunication

Spécialité : Ingénierie des réseaux

Implémentation des couches logicielles de Bluetooth sous Linux

Présenté par : M^elle^ Kaouthar SETHOM

Encadré par : M. Hossam AFFIFI

M. Rached HAMZA

Année universitaire 2001/2002

Remerciement

Je tiens à exprimer ma profonde gratitude et mon immense respect à Monsieur HOUSSAM AFIFI, pour la qualité de son encadrement, sa disponibilité, ses hautes qualités morales et scientifiques et pour m'avoir fait découvrir un domaine de recherche si passionnant.

Mes remerciements s'adressent également à Monsieur Rached Hamza, mon co-encadreur à SUPCOM pour son intérêt particulier et son suivi permanent.

Je remercie également l'ensemble de l'équipe RST qui m'ont fait l'honneur de m'accueillir si sympathiquement dans leur groupe.

Finalement, que tous ceux qui m'ont permis de vivre cette expérience enrichissante reçoivent ici le témoignage de ma reconnaissance.

Sommaire

Chapitre III : Bluetooth et Linux

Présentation de l ' INT

L' Institut National de Télécommunications fait partie du Groupe des Ecoles des Télécommunications, établissement public d'enseignement supérieur et de recherche. L'INT rassemble sur son campus, une Grande Ecole de Management, INT MANAGEMENT, une Grande Ecole d'Ingénieurs, **TELECOM INT**, un centre de création d'entreprises, INT ENTREPRENEURIAT, un pôle de formation continue, INT ENTREPRISES, et un centre de recherche, INT-RECHERCHE composant du GET-RECHERCHE.
L'INT représente plus de 1000 étudiants, 160 enseignants - chercheurs, 20 groupes de recherche, plus de 15 plates-formes de recherche, partenaire de 3 Écoles Doctorales, 5 DEA et 3 DESS cohabilités, 2 Master of Science, et plus de 100 séminaires de formation continue.

L'organisation de l'INT :

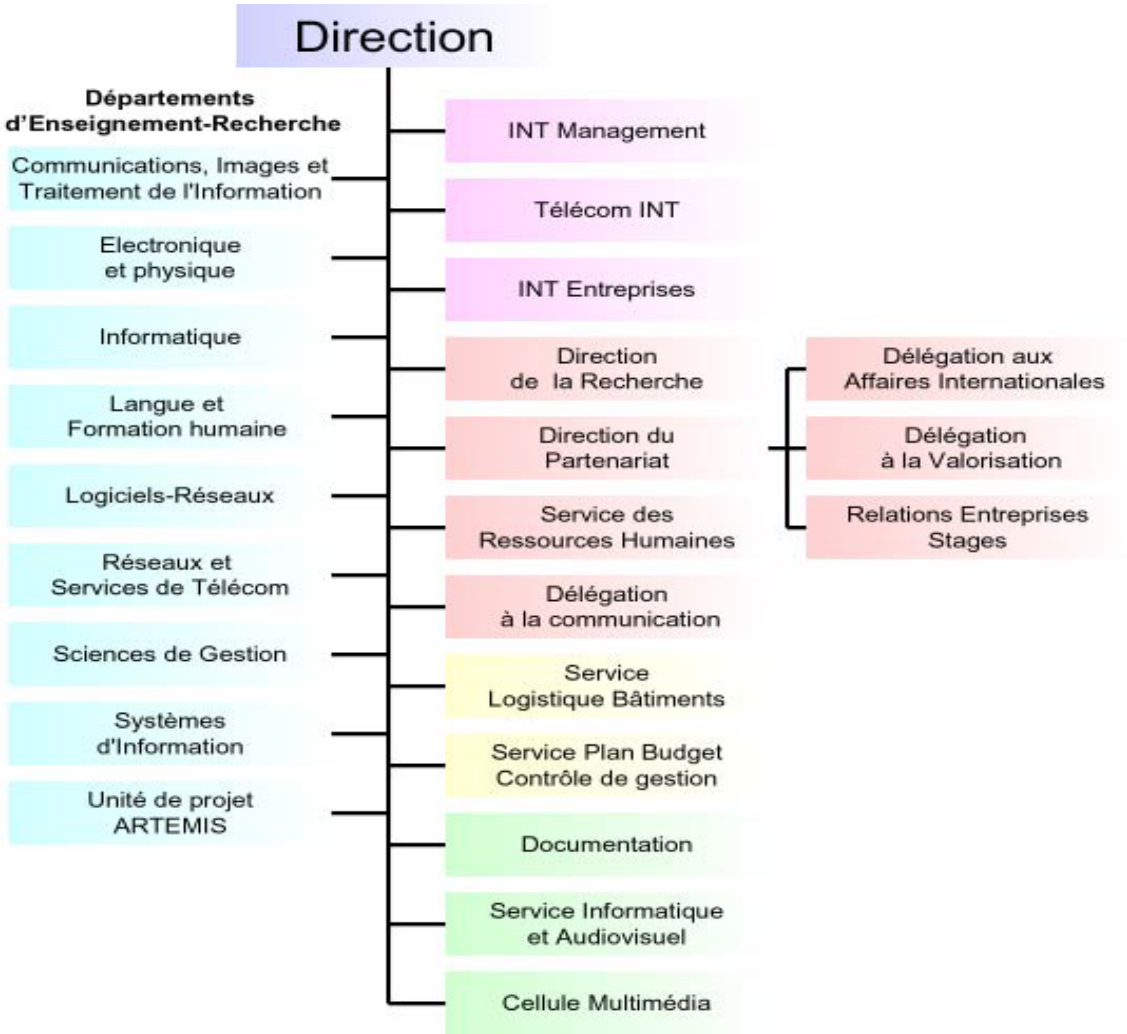


Figure I-1: Organigramme INT

Le Département RST

Créé le 1er Janvier 1994, le département "Réseaux et Services des Télécommunications" (RST) est l'un des huit départements de l'Institut National des Télécommunications.

Son domaine d'activités recouvre tous les aspects de l'enseignement et de la recherche des télécommunications, y compris les communications avec les mobiles, les réseaux large bande et le RNIS.

Le département RST entretient de nombreux contacts avec le monde industriel des télécommunications ainsi qu'avec le CNET et les laboratoires les plus importants dans le domaine; des contrats de recherche avec des partenaires industriels et la Communauté Européenne contribuent également au développement des compétences de ses équipes de recherche et de développement.

Le département est impliqué dans tous les domaines d'enseignement des télécommunications, tant pour le compte de l'Ecole d'Ingénieurs, Telecom INT, que pour celui de l'Ecole de Gestion, INT Management.

Il a en particulier la responsabilité pédagogique de deux options de troisième année:" Communications Mobiles" et "Réseaux de Télécommunications".

Le Département Réseaux et Services de Télécommunications conduit des recherches organisées autour de quatre grands thèmes :

- ***Conception, Planification et Sécurisation***
- ***Gestion des réseaux et des services***
- ***Architecture des réseaux à haut débit***
- ***Communications avec les mobiles***

Introduction

L'ouverture à la concurrence des télécommunications, l'augmentation des débits liée au multimédia et le succès prodigieux du téléphone mobile constituent autant de facteurs d'accélération de la diversification des offres de technologie, avec une prime aux solutions sans fil.

Le monde recherche davantage de mobilité et de nouveaux moyens de communication sans fil: Téléphonie sans fil, ordinateurs sans fil, organiseurs sans fil, le tout sans fil. Relativement récents, les réseaux sans fil sont de plus en plus performants grâce notamment aux avancées de l'électronique et du traitement du signal. Le monde Internet s'intéresse aussi à la mobilité. En témoignent les travaux du groupe de travail MobileIP de l'IETFqui a défini un protocole pour le support de la mobilité sur l'Internet actuel et ceux du nouveau groupe SeaMoby (Seamless Mobility) qui a pour objectif de standardiser des mécanismes optimisés de gestion des handovers avec maintien des contextes de QoS et de sécurité pour les équipements mobiles.

Dans les technologies mobiles il y a :
-Les WPAN (Wireless Personal Area Network) : **Bluetooth**, HomeRF.
-Les WLAN (Wireless Local Area Networks) : IEEE 802.11 (US) et Hiperlan (Europe) .
-Les technologies cellulaires (GSM, GPRS, UMTS) .
-Les technologies Satellite (Vsat qui est bidirectionnel, mais aussi DVB pour la diffusion Vidéo).

Dans cet univers imminent de la télécommunication sans fil, des idées nouvelles - souvent inconcevables avec le fil - semblent aussi infinies que l'air qu'elles traversent. Il s'agit, notamment, de lancer le m-commerce (commerce électronique mobile).

Parmi ces applications, encore très innovantes, le **Software Radio** ou encore la Radio Logicielle (SDR). La SDR a pour objectif la migration du hardware vers le software, en implémentant de plus en plus de fonctionnalités radio dans le logiciel.

C'est dans ce cadre du " tout est possible en logiciel " que vient s'inscrire notre projet de fin d'étude. Lors de ce travail, nous avons commencé par l'étude des spécifications de la norme Bluetooth et la recherche des interfaces qui sont compatibles avec le système Linux.
Dans une deuxième phase, et en tenant compte des caractéristiques de la norme et des contraintes imposées par le matériel , nous nous sommes intéressés, à l'étude à la pile de protocoles Bluetooth développé pour Linux sous le nom de Bluez et aux problèmes d'implémentation et de configuration de IP sur Bluetooth.
Enfin, nous avons réalisé une série de mesures afin d'évaluer les performances de la technologie Bluetooth dans l'environnement Widows et Linux.

Etat de l'art

Avec la montée en puissance d'Internet et du multimédia, les échanges de données entre appareils portables tels que téléphones et organiseurs deviennent de plus en plus fréquents. Une opération, parfois délicate, plonge souvent l'utilisateur dans la plus grande perplexité lors du choix de câbles, fiches et autre adaptateurs d'interconnexion.
L'arrivée de **Bluetooth**, littéralement "dent bleue", sonne le glas de ces accessoires.
Bluetooth se propose de simplifier ces problèmes : cette technologie permet de mettre en liaison un ensemble de périphériques simplement en les rapprochant à moins de 10 mètres les uns des autres.

Mais Bluetooth va bien plus loin : outre la communication entre appareils informatiques, il permet de relier n'importe quel appareil électronique. Il devient donc possible de relier son ordinateur à son imprimante, mais également à son téléphone portable, à son agenda électronique, à sa voiture ...

Ce système se veut *robuste*, *simple*, *économique en énergie* et *peu coûteux* .
Par contre, le principal inconvénient réside dans l'exposition aux parasites atmosphériques. De plus sa portée et son débit binaire sont faible par rapport à d'autres types de réseaux locaux (comme Ethernet par exemple).

Bluetooh est le nom d'un chef Viking du nord : *Harald Blaatand* ("Harald la dent bleue"). Celui-ci réussit l'exploit d'unifier au sein d'un même royaume le Danemark et la Norvège, à l'époque où l'Europe était divisée par des querelles de religions et de territoires.

La métaphore est belle : aujourd'hui **Bluetooth mène une véritable croisade des temps moderne pour unifier l'ensemble des constructeurs** et leurs appareils électroniques grâce à a cette nouvelle technologie de communication sans fil.

Ericsson, l'initiateur du projet, fut rapidement rejoint en 1998 par IBM, Intel, Nokia et Toshiba. Ils développèrent ensemble cette nouvelle technologie de communication sans fil. Devant le marché prometteur s'ouvrant à cette technologie, de nombreuses sociétés ont rejoint le *Bluetooth Special Interest Group (SIG)* qui est aujourd'hui composé de plus de 2400 constructeurs, dont 3 Com, Motorola et Microsoft.
Leur but est d' imposer cette nouvelle norme comme **le standard des système de communication sans fil**.

Vu que le marché visé est énorme : Ericson estime à plus de 100 millions le nombre de téléphones et autres équipements électroniques équipées de la puce Bluetooth en 2003. L'institut américain Dataquest prévoit de son côté qu'en 2004, 70% des téléphones cellulaires et 40% des assistants personnels communiqueront grâce à Bluetooth pour accéder à Internet et aux réseaux d'entreprise.

I- Principe de fonctionnement

Les appareils compatible Bluetooth communiquent en utilisant les ondes radio sur les fréquences comprises entre 2400 et 2483.5 MHz. Le débit théorique est de **1 Mb/s**.
Tout comme le protocole IP, l'envoie des informations s'effectue par paquets de données entourées de blocs de contrôle. Ces blocs de contrôle permettent la mise en réseau des appareils à distance suffisante, le bon acheminement des données et la correction d'éventuelles erreurs de transmission.

Pour pouvoir communiquer entre eux, les appareils doivent se trouver à une distance maximum de 10 mètres. Il peut y avoir jusqu'à 8 appareils chaînés pour former un petit réseau (nommé picoréseau ou *piconet*). Il est cependant possible de connecter entre eux ces piconets pour former des réseaux plus grands nommés **scatternets**. Certains appareils Bluetooth pourront donc servir de passerelle.

Figure I-2 : Un piconet

1. Les spécifications techniques Bluetooth

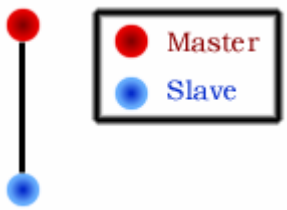
Bluetooth se base sur les ondes hertziennes pour communiquer. Celles-ci sont situées autour de 2.4GHz. Plus exactement, un appareil Bluetooth doit avoir la capacité de balayer toute une plage de fréquences: [2400 - 2483.5 MHz]. Des canaux de communication sont définis sur toute l'étendue de cette plage par tranches de 1 MHz. Ceci fait donc environs 80 canaux distincts. Le changement de fréquence est également appelé saut de fréquence (*frequence hopping*). Ces sauts de fréquence ont lieux 1600 fois par secondes. Entre chaque sauts de fréquence, on trouve des plages de quelques micro secondes durant lesquelles des paquets peuvent êtres envoyés. Ces plages sont appelées " slots ".

2. Bluetooth et la législation internationale

Dans certains pays comme la France les plages de fréquence standard Bluetooth sont occupées par l'armée, et donc interdites. Elles n'ont pas pu être libérées entièrement ou le seront par la suite. En attendent, des restrictions partielles sont en vigueur et ceci entraîne des répercutions sur le nombre de canaux disponibles. 22 seulement en France, au Japon ou en Espagne par exemple.

3. Schémas de connexion

Au sein d'un réseau Bluetooth toutes les unités le composant sont identiques du point de vue hardware ainsi que de l'interface logicielle. Il n'y a que l'adresse Bluetooth sur 48 bits de chacune des unités qui les différencie.
Lors d'une tentative de connexion, l'unité initiant cette connexion devient temporairement l'unité Maître. Ce statut d'unité Maître n'est valable que pour la durée de la connexion.
Le rôle de l'unité Maître est bien sûr d'initier la connexion mais aussi de contrôler le trafic des autres unités appelées Esclaves. Le plus simple des schémas de connexion est établie lors de la communication entre 2 périphériques Bluetooth. Un des deux appareils jouera le rôle de Maître (**M**aster) et l'autre d'esclave (**S**lave). Le maître est chargé de gérer la communication entre les deux périphériques : c'est lui qui a initialisé la connexion.



FigureI-3 : Maître /Esclave

Dans le cas où plus de 2 périphériques se connectent au sein du même piconet, un des appareils devient maître et les autres esclaves. Le maître est alors chargé de gérer les communications entre les différents esclaves : lorsque 2 esclaves souhaitent échanger des informations, cette discussion est orchestrée par le maître.

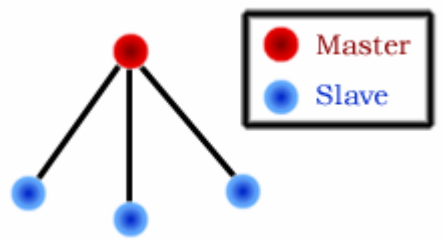


Figure I-4 : Maître/Esclaves

Plusieurs piconets peuvent se réunir pour former un scatternet. Dans ce cas, le maître d'un piconet deviendra l'esclave du maître d'un autre piconet. Un périphérique pourra également devenir l'esclave de plusieurs maîtres de différents piconets, comme le montre le schéma ci-dessus.

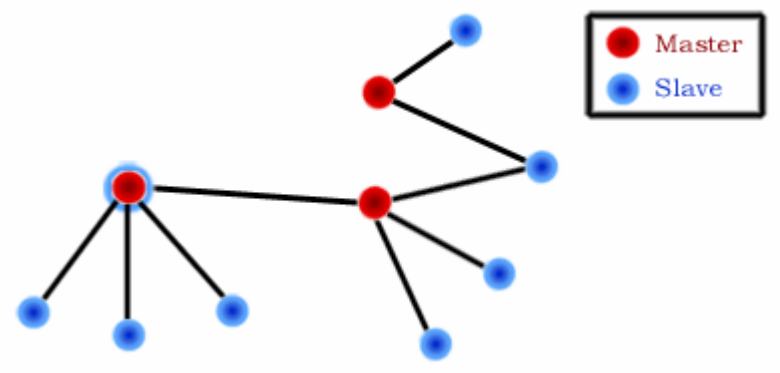


Figure I-5 : Architecture d'un Scatternet

4. Description des liens physiques

Les appareils Bluetooth doivent respecter un standard de communication. Ce standard est basé sur le principe des paquets. Ces derniers constituent la brique de base de la communication entre les appareils Bluetooth.
Il existe deux façons de transporter les paquets dans cette norme ; le transport connecté synchrone orienté (ou SCO) et le transport asynchrone non connecté (ou ACL) :

- **Les liens de connexion SCO :** Ces liens sont des connections symétriques (full-duplex) point à point entre un maître et un seul esclave au sein d'un pico réseau. Le maître maintient ce lien via des "slots " qu'il réserve et qui reviennent à intervalles

régulières . Dans ce cas , on travaille en mode circuit. C'est la régularité du débit de ce mode qui le rend idéal pour le transport des données de type vocales. Un maître peu supporter jusqu'à trois liens de communication SCO simultanément. Un esclave peut lui en supporter trois avec le même maître ou deux avec des maîtres différents. Les paquets de type SCO ne sont jamais retransmis.

- **Les liens de connections ACL :** Ce sont des connections point à multi point entre le maître et tous les esclaves de son pico réseau. Dans les " slots " qui ne sont pas réservés pour les liens SCO, un maître peut établir un lien ACL avec un esclave, même avec ceux déjà impliqués dans une connexion de type SCO. Ce mode permet donc la commutation de paquet. La sous adresse de la partie Header d'un paquet définit l'unité Esclave destinataire de ce paquet; Si la sous adresse n'est constituée que de 0, alors toutes les unités Esclaves sont destinataires; c'est un paquet de type broadcast La retransmission peut être utilisée avec ce type de paquet.

- **Combinaisons de liens :** Ces combinaisons peuvent aller d'un lien SCO full-rate (ou de trois liens SCO third-rate) sans lien ACL jusqu'à un lien ACL full-rate sans lien SCO.

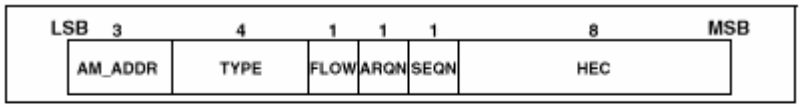
5. Les paquets Bluetooth

Le format standard des paquets est le suivant :

72 bits	54 bits	[0 - 2745 bits]
Code d'accès	Entête	Corps du message

Figure I-6 : Format d'un paquet Bluetooth

Les 72 premiers bits servent à la synchronisation entre les composants Bluetooth.
L'entête est composé de :



- AM_ADDR: 3- bit active member address
- TYPE: 4-bit type code
- FLOW: 1-bit flow control
- ARQN: 1-bit acknowledge indication
- SEQN: 1-bit sequence number
- HEC: 8-bit header error check

Figure I-7 : Format de l'entête d'un paquet Base bande

Elle permet :

- D'identifier le destinataire AM_ADDR. L'adresse est sur 3 bits puisqu'un piconet ne comporte pas plus de 7 esclaves et 1 maître.
- De préciser le type du paquet que l'on est en train de consulter. Les concepteurs de Bluetooth ont identifié 16 types différents de paquets selon leur taille (de 1, 3 ou 5 slots) et leur type de contenu, d'où la nécessité de 4 bits pour les différentier.
- De dire si oui ou non un accusé de récéption est attendu en retour à ce message.
- Un CRC.

- Les autres bits sont typiquement liés à la sécurité de la transmission.

Le corps sert lui à stocker les données à transporter. Il contient généralement un CRC de 8 ou 16 bits. L'envoie d'un paquet nécessite un " slot ".

Lorsqu'un appareil Bluetooth émet un paquet, celui-ci peut appartenir à une des trois grandes familles de paquets existants, son type étant précisé dans le header :

- **Les paquets standards :** Ils sont utilisés dans les opérations "administrative "; c'est à dire dédiés à la gestion des connections entre les appareils.
- **Les paquets SCO :** Ce type de paquet est utilisé pour les communications de données de type SCO.
- **Les paquets ACL :** Ce sont des paquets spécifiques au mode de transmission de données ACL.

Pour chacun de ces types, plusieurs sous catégories existent. Les différents types de paquets qui en découlent se sont vus attribuer une nomenclature. On distingue dans cette nomenclature les paquets suivants :

- **Les paquets DV** : " Data Voice packet ". Ces paquets permettent de transporter à la fois des données et de la voie.
- **Les paquets DM x** : " Medium Data rate packet". Ce type de paquet n'est disponible qu'en mode ACL. Cette dénomination est due au fait que le corps de ce type de paquet est toujours encodé afin d'obtenir de la redondance (prévention d'erreur).
- **Les paquets DH x** : " High Data rate packet". Ce type de paquet n'est également disponible qu'en mode ACL. Son nom vient du fait qu'aucun encodage de prévention d'erreur n'est employé, d'où un meilleur taux de transfère.
- **Les paquets HV y** : " High quality Voice packet ". Ce type de paquet n'est disponible qu'en mode SCO. Ces paquets n'utilisent pas de CRC dans leur corps.

Le x dans les notations " DM **x** " et " DH **x** " remplace un des chiffres suivants : 1,3 ou 5. Ce chiffre représente le nombre de slots sur lesquels ce paquet s'étend. Par exemple les paquets de type DM3 s'étalent sur 3 slots.

6. Les protocoles de connexion Bluetooth

Lorsque aucune connexion n'est établie dans le réseau, tous les périphériques sont en mode STANDBY. Dans ce mode, une unité non connectée " écoute " les messages périodiquement toutes les 1.28 secondes. La procédure de connexion est initiée par n'importe quelle unité du réseau, celle-ci devenant alors Maître.
Afin d'expliquer le fonctionnement du protocole Bluetooth lors d'une connexion, nous allons prendre un exemple simple. Soit deux terminaux Bluetooth notés **M** et **E**. Ici **M** jouera le rôle du maître et **E** celui de l'esclave.

Le maître et l'esclave sont au début de notre exemple dans un état dit passif. La puce **M** va initialiser un lien avec la puce **E**. L'état passif pour une puce Bluetooth signifie

principalement une consommation d'énergie réduite. Dans un état passif il existe plusieurs sous états. Ceux-ci servent durant l'établissement d'une connexion (Figure **I-9**).
Le schéma suivant permet de mieux comprendre **le mécanisme de connexion** :

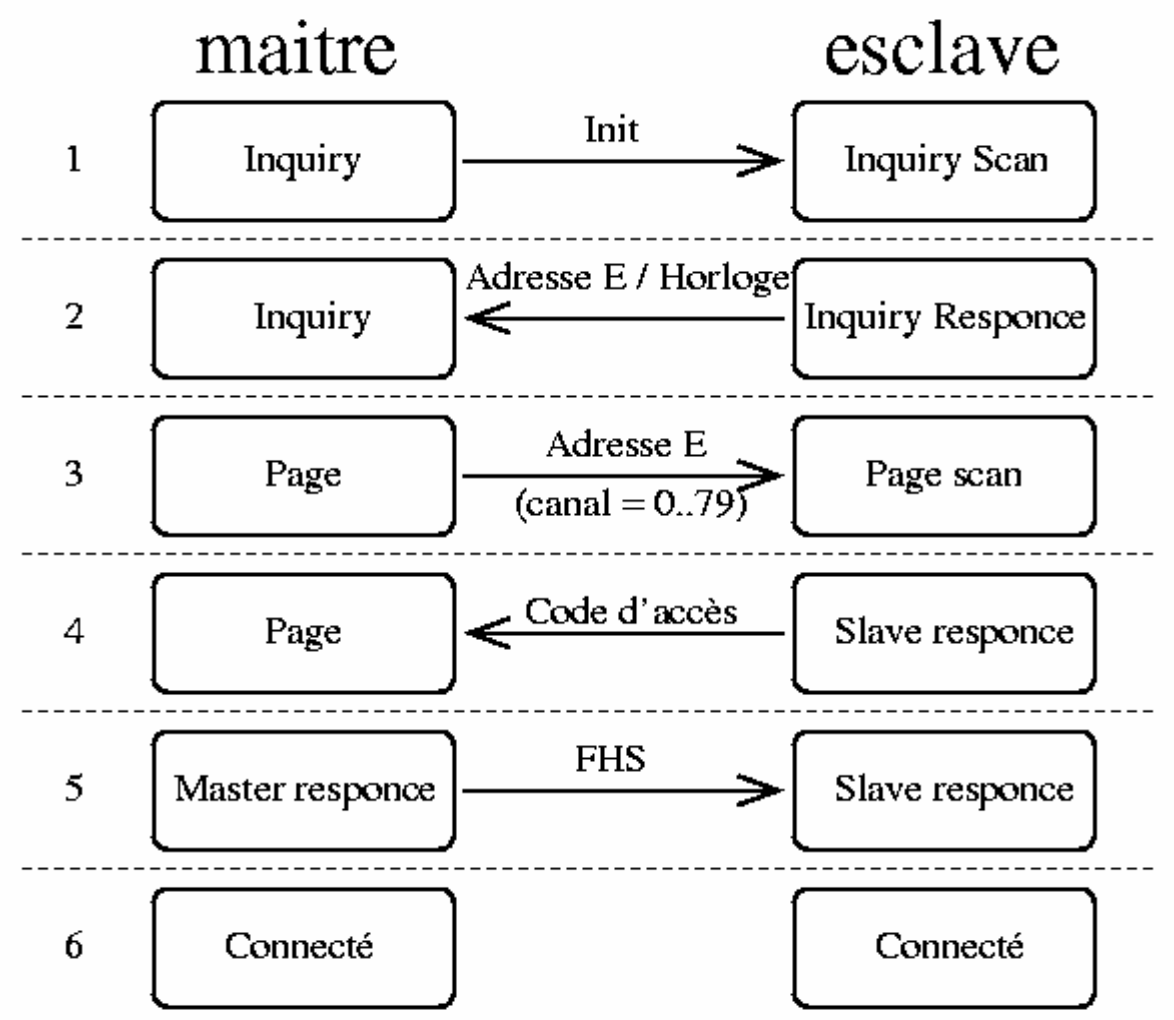


Figure I-8: Mécanisme de connexion

Une connexion est établie par un message de type PAGE si l'adresse de l'unité à connecter (unité Esclave) est connue ou alors un message de type INQUIRY (demandant à toutes les unités de répondre) suivi d'un PAGE si l'adresse n'est pas connue.
Au début du processus, le maître **M** doit se trouver dans le sous état " Inquiry " et l'esclave **E** dans l'état " Inquiry scan " :

1. Etant dans l'état " Inquiry ", **M** envoies un signal pour prévenir **E** qu'il souhaite initialiser une connexion. **E** se trouve alors dans l'état "inquiry scan".

2. Si **E** se trouve à portée et qu'il est dans l'état " Inquiry scan ", il passe alors dans le sous état " Inquiry responce " puis répond effectivement au maître. La réponse de **E** comporte entre autre son adresse (adresse MAC sur 48 bits) ainsi que des informations sur son horloge.

3. Une fois que **E** a envoyé sa réponse, il passe dans l'état " Page Scan ". Il se met ensuite en attente d'un message comportant sa propre adresse sur un des 80 canaux existants. Lorsque **M** reçoit le message réponse de **E**, celui-ci passe

dans l'état " page ". C'est à dire que **M** stocke les informations reçus (pagination). Ces informations permettent à **M** d'avoir conscience de la présence de **E**. Lorsque **M** souhaite poursuivre le processus de connexion, celui-ci renvois un message réponse en y plaçant l'adresse de **E**. Ce message est renvoyé plusieurs fois sur tous les canaux.

4. Lorsque **E** voit une réponse à son nom arriver, il se place dans le sous état " Slave responce " puis renvoie un message réponse à **M** en y joignant son code d'accès.

5. De son coté, **M** une fois ce code d'accès récupéré, se place alors dans un état " Master responce " et renvoie un paquet de type FHS à **E**. Ce paquet de type FHS (Frequency Hopping Synchronisation) permet à **E** de se synchroniser avec **M**.

6. Une fois ce dernier message envoyé, **M** passe dans l'état " connecté ". De même, lorsque **E** reçoit ce message il passe aussi dans l'état "connecté".

Ici l'état "connecté" n'est pas un sous état. Pour vérifier que la connexion s'est bien passée, le maître envoie un paquet de type POLL et attend en retour n'importe quel type de paquet. Si une connexion s'est effectivement bien passée, l'esclave est synchronisé avec son maître et se trouve sur le bon canal de communication.

Le temps de connexion est typiquement de 0.64 s (ceci s'applique lorsque l'adresse de l'unité en question est connue et que 5 heures ne se sont pas écoulées avant la dernière connexion). Moins d'une seconde étant nécessaire pour débuter une transmission, il n'est donc pas utile qu'une unité soit connectée en permanence. C'est pourquoi, lorsqu'une unité n'est pas active, celle-ci est placée dans un mode de repos (mode STANDBY) où seul un Low Power Oscillator (LPO) fonctionne. Ceci est bien entendu bénéfique pour la durée de vie des batteries.

Ainsi, une fois qu'un esclave est dans un état "connecté", celui-ci peut à nouveau se trouver dans plusieurs sous états :

- **Sous état actif :** en mode actif, le maître comme l'esclave participent activement à la communication sur le canal (écoute, envois de paquets, réception).

- **Sous état suspendu (hold) :** dans un état suspendu, un esclave ne peut plus recevoir que des messages de type SCO. Cet état dure pendant N slots. Ce dernier coefficient est fixé par le maître. Les messages de type SCO arrivants à intervalles réguliers, l'esclave peut s'endormir lorsqu'il n'est pas susceptible d'en recevoir. Ceci permet d'économiser de l'énergie.

- **Sous état parqué (park) :** Un esclave dans cet état est très peu actif (économie d'énergie). Lorsqu'un esclave est dans l'état parqué celui-ci ne reçoit plus du tout de messages ni n'en envois. Sa seule activité est de se réveiller de temps en temps pour se synchroniser avec le maître grâce à des "balises" que celui-ci envoie régulièrement. Cet état passif permet de libérer une des 7 places disponibles dans un pico réseau. Plus de 7 périphériques Bluetooth peuvent ainsi cohabiter au sein d'un même pico réseau en jouant avec cette possibilité.

- **Sous état tâtonnage (sniff) :** Dans le sous état tâtonnage, un esclave peut alterner N slots d'état endormis (économie d'énergie), et K slots d'états actifs.

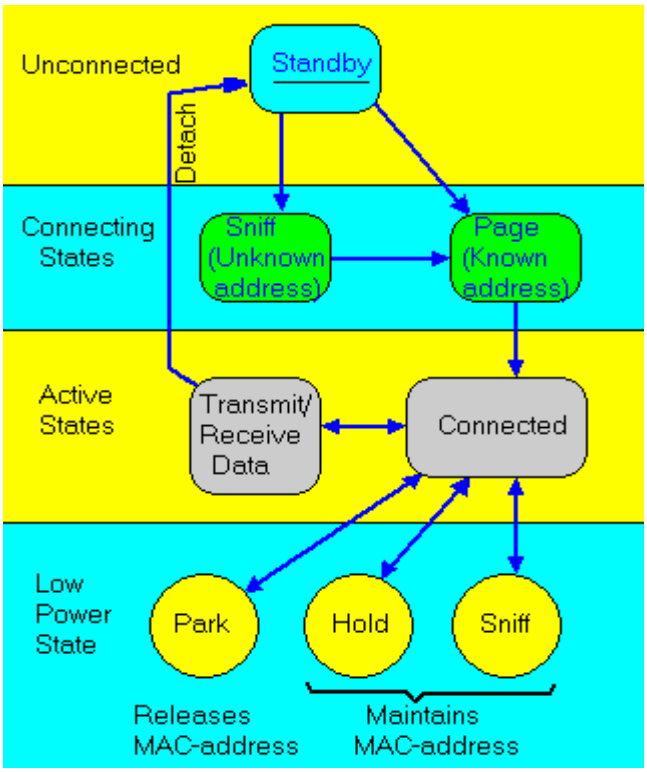


Figure I-9 : Les différents états d'un esclave

II- Architecture Bluetooth

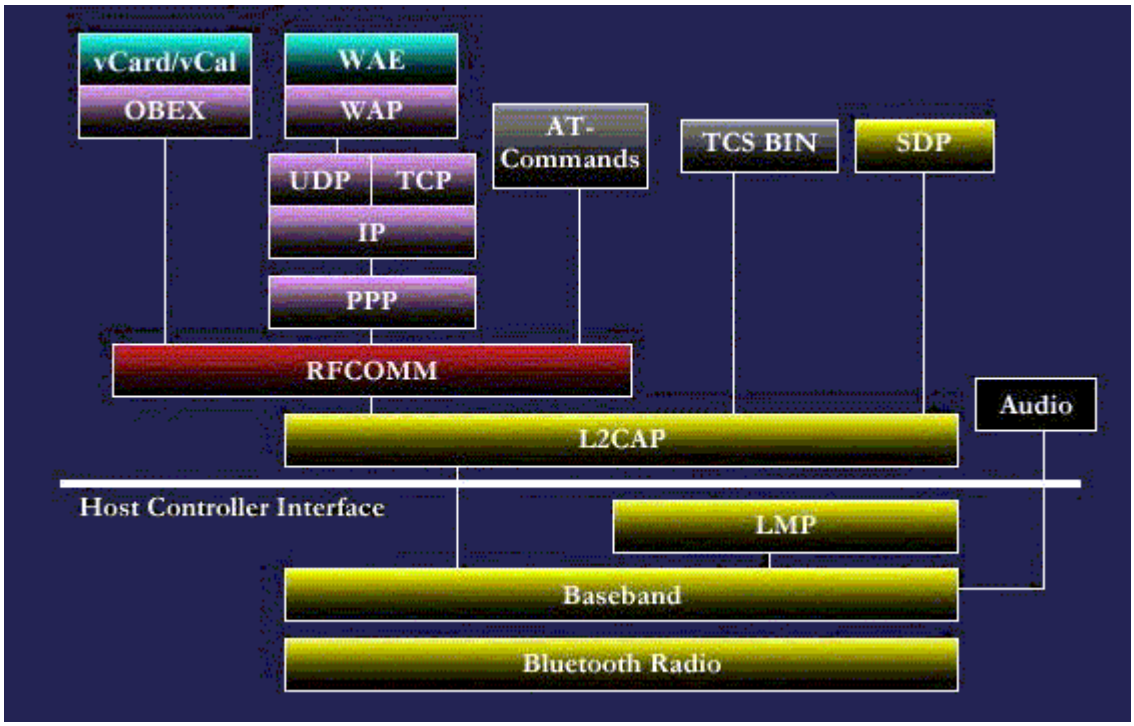


Figure I-10 : Pile de protocole Bluetooth

- Système de communication Bluetooth (seul) : Bluetooth Radio, Baseband, LMP, L2CAP et SDP.
- Protocoles de communication avec la téléphonie (Se sert de la partie *Audio*) :AT-commands, TCS BIN.
- Protocole simple de transmission de données. RFCOMM, sert entre autre à l'émulation de ports série.
- Protocoles adoptés par le système Bluetooth : PPP, IP, UDP/TCP, WAP et OBEX.
- Applications de haut niveau :vCARD/vCal et WAE.

1. Composants des systèmes Bluetooth

On peut dénombrer plusieurs composants indispensables dans un système Bluetooth :

- L'antenne de réception/émission des fréquences radio :

Ce composant n'a pas de spécifications générales pour tous les systèmes. L'antenne dépend du type d'appareil sur lequel Bluetooth est implémenté. C'est pour cela qu'elle n'est pas représentée sur la **figure I-10** .

- Système radio et Baseband :

Il s'agit ici, de la partie liaison physique avec la couche de plus bas niveau (Bluetooth Radio) et la deuxième couche (Baseband).

- La couche radio est destinée à émettre et recevoir les ondes radio des systèmes environnant. Elle fait la conversion des ondes radio (dont la fréquence de base est 2.4 GHz) en signaux électriques utilisables par la couche supérieure (Baseband).

- La couche Baseband inclue des routines de liaison bas niveau servant à établir les communications entre les systèmes Bluetooth. Elle permet, entre autre, le contrôle des liaisons entre une ou plusieurs unités Bluetooth, la définition des paquets de données, la détection et la correction d'erreurs de bas niveau, la gestion des canaux logiques et la gestion des intervalles de temps de transmission.

- Couches de protocole logiciel (bas niveau) :

Ces différentes couches logicielles peuvent être perçues comme des *drivers*. Elles permettent de faire la liaison entre les couches supérieures et la partie matérielle (Radio et Baseband). Les couches à prendre en compte, ici, sont les suivantes :

- Gestion des liaisons (Link Manager et le protocole LMP).

- Contrôle des liens logiques et adaptation des protocoles (Logical Link Control and Adaptation Protocol : L2CAP).

- Interface de contrôle de l'hôte (Host Control Interface : HCI).

- Protocole de découverte de service (Service Discovery Protocol : SDP).

- Autres couches de protocole logiciel (moyen niveau) :

- Pour la couche TCS BIN (Telephony Control protocol Specification BINary), il s'agit d'un protocole permettant de communiquer avec des outils de téléphonie, téléphone portable par exemple, possédant la technologie Bluetooth. Il permet de transmettre la voix (via la passerelle *Audio*) et des données.

- La couche AT-Commands sert à communiquer avec des appareils de type modem, téléphones mobile acceptant les commandes AT et les FAX (compatibilité avec plusieurs classes). Cette couche communique avec la couche inférieure (RFCOMM) décrite ci-dessous.

- Le protocole RFCOMM sert à transmettre des données aux couches de haut niveau. De plus, ce protocole permet l'émulation de liaison série du type RS232 pour tout types d'applications. Notamment, pour la communication avec des commandes AT.

- Protocoles et Applications de haut niveau :

Certaines couches protocolaires ont été ajoutés par le système Bluetooth afin de permettre aux applications traditionnelles ,exemple transfert de fichier, de fonctionner indépendamment des systèmes Bluetooth. (Voir section suivante :les profils).

2. Les profiles

Un profile définit un ensemble de composantes protocolaires (SDP,RFCOMM….) nécessaires à la mise en œuvre d'applications Bluetooth. A l'heure actuelle, plusieurs profiles ont été spécifiés par le Bluetooth SIG (Special Interest Group). De façon générale, un profile doit être compris comme étant la définition de l'application.

a. Transfère de fichiers :File transfer

Ce profile offre la possibilité d'un transfert de données d'une unité Bluetooth vers une autre.

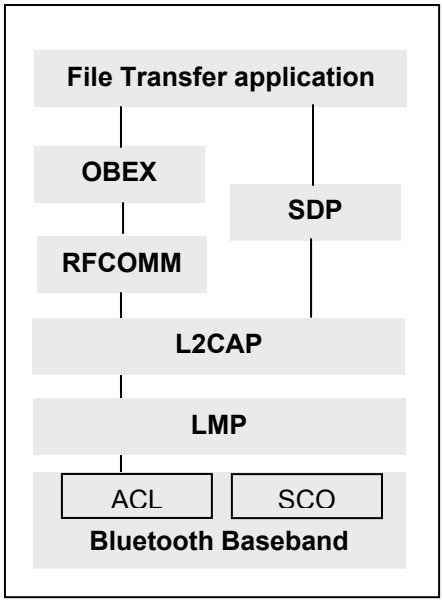


Figure I-11: Transfer de fichiers

III- TCP/IP over Bluetooth

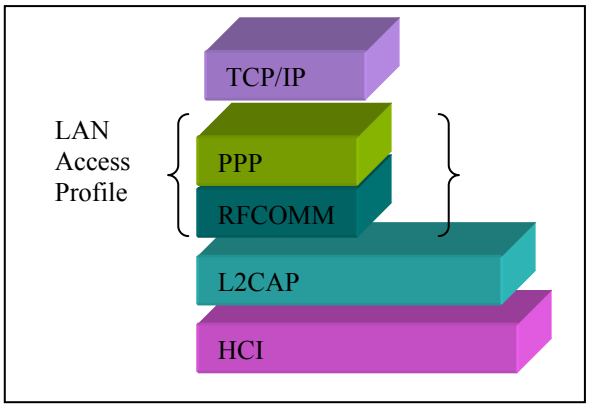


Figure I-12: TCP/IP over Bluetooth

Le protocole point à point (PPP) est un protocole de liaison de données assurant l'échange de données de manière fiable sur une liaison point à point (par exemple, une liaison RTC). Sa principale caractéristique est, une fois la liaison établie et configurée, de permettre à plusieurs protocoles de transférer des données simultanément. De ce fait, ce protocole est très utilisé dans l'environnement de l'Internet. L'utilisation de PPP au dessus de RFCOMM est le moyen le plus simple pour supporter TCP/IP ou tout autre protocole réseau au dessus de la pile de protocole Bluetooth.

En effet :

- PPP est déjà mis en œuvre dans la plupart des dispositifs portables.
- PPP permet l'authentification des utilisateurs (le contrôle d'accès de RÉSEAU LOCAL).
- PPP porte IPv4, IPv6 et d'autres protocoles.
- PPP résout (à un degré raisonnable) le problème de nomination d'adresse IP dans des contextes ad hoc.

Cependant, la spécification est ouverte et il est aussi possible de configurer IP directement sur L2CAP. Mais actuellement, il n'y a aucun profile prédéterminé qui décrit cette fonctionnalité; alors que des profiles tel que le Dial-up networking et le LAN Access décrivent comment on peut établir une connexion TCP/IP au dessus de RFCOMM / PPP.

Nous allons maintenant voir plus en détails les spécifications de chaque couches :

1. Host Control Interface (HCI)

L'HCI fournit une interface qui permet une méthode d'accès uniforme aux fonctionnalités de la couche Base bande indépendamment de l'interface réelle du matériel Bluetooth : USB, UART, RS232 .

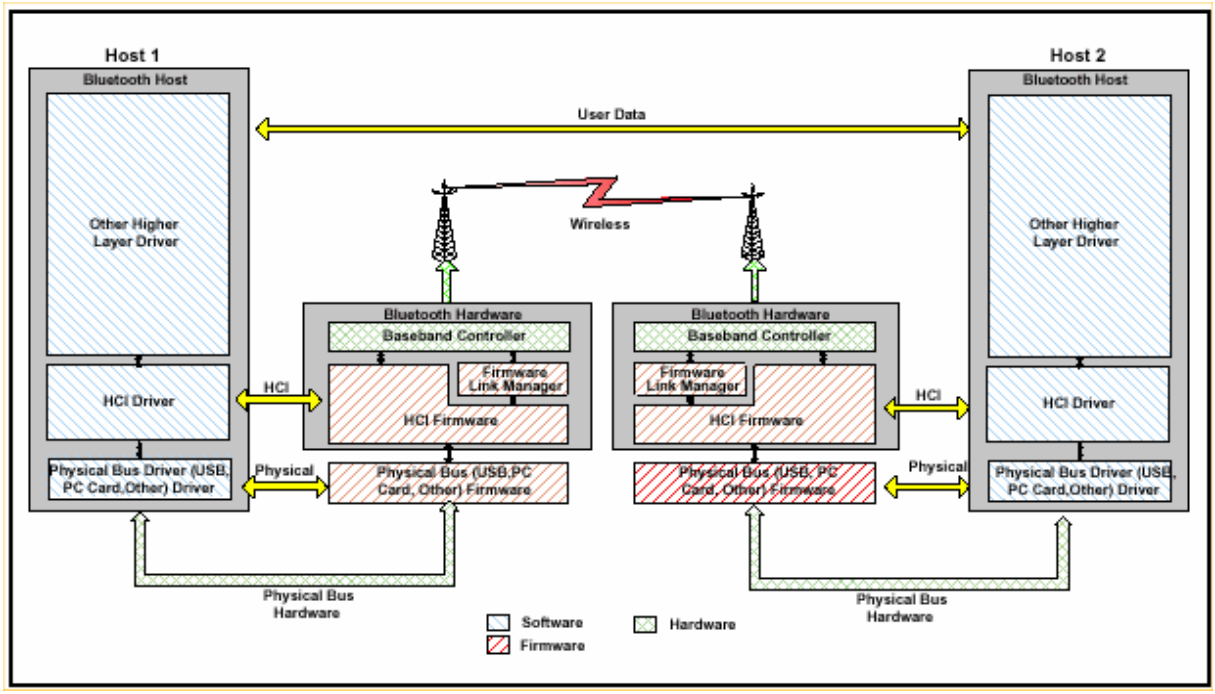


Figure I-13 : Entités Fonctionnelles HCI

Pour beaucoup de dispositifs, par exemple, sur un PC ou un ordinateur portable, le matériel Bluetooth peut être ajouté comme une carte PCI ou un adaptateur USB.
Ces modules mettent d'habitude en œuvre la radio des couches inférieures : Base bande et LMP. Un Driver est exigé sur "l'hôte", qui est le PC ,et "une interface de contrôleur d'hôte" est exigée sur la carte Bluetooth pour accepter des données sur le bus physique.
Ainsi, si L2CAP et les couches qui lui sont supérieures sont dans le logiciel et celles qui lui sont inférieurs dans le hardware, alors les couches supplémentaires suivantes sont exigées :

a. *HCI Firmware*

Le HCI Firmware, est placé sur **le Contrôleur de l'Hôte**, (par exemple la carte Bluetooth). C'est un microprogramme qui exécute les Commandes HCI sur le matériel Bluetooth en ayant accès aux commandes Base bande et LM, aux registres d'état, …

b. *Driver HCI*

Le Driver HCI est placé sur **l'Hôte** (exemple le software). L'Hôte recevra des avis asynchrones d'événements HCI pour lui notifier que quelque chose vient d'arriver. Quand l'Hôte découvre qu'un événement vient de se dérouler, il effectue alors l'analyse syntaxique du paquet reçu pour déterminer de quel événement s'agit il.

c. *Couche de Transport*

Le driver HCI et le Microprogramme communiquent via **la Couche de Transport**, c'est-à-dire une définition de plusieurs couches qui peuvent exister entre le driver HCI sur le système d'hôte et le microprogramme HCI dans le matériel Bluetooth. Ces couches intermédiaires doivent fournir la capacité de transférer des données sans connaissance intime des données

étant transférées. Plusieurs Couches de Contrôleur d'Hôte différentes peuvent être employées, 3 ont été définies au début du projet Bluetooth : **USB**, **UART** et **RS232.**

2. Contrôle de Liaison Logique et Protocole d'Adaptation L2CAP

Le protocole de contrôle des liaisons logiques et d'adaptation se situe au-dessus du protocole bande base et réside au niveau de la couche liaison de données.
L2CAP permet aux protocoles de niveau supérieur de transmettre et de recevoir des paquets de données L2CAP de longueur jusqu'à 64 kilo-octets. L2CAP assure un service de multiplexage, segmentation et ré assemblage des paquets . Il permet de rendre "**compréhensible**" Bluetooth par les protocoles ou applications externes.
Deux types de liaisons sont supportées par la couche Base bande : Synchrone Orienté Connexion (SCO) et Asynchrone Sans connexion (ACL). SCO sont des liaisons pour le trafic de voix en temps réel. ACL est du best effort. La Spécification L2CAP est définie seulement pour des liaisons ACL.

L2CAP est à base de paquets, mais suit un modèle de communication basé sur *des canaux*. Un canal représente un flux de données entre des entités L2CAP dans des dispositifs éloignés ; il est identifié par un identificateur de canal logique CID (*logical channel identifier*).
Trois types de canaux L2CAP existent :
- les canaux de signalisation bidirectionnels qui véhiculent des commandes ;
- les canaux orientés connexion bidirectionnels pour des connexions point a point;
- et les canaux sans connexion unidirectionnels pour des connexions point-multipoint, permettant à une entité L2CAP locale d'être connecté à un groupe de dispositifs éloignés.

L2CAP doit être capable de déterminer l'adresse Bluetooth (BD_ADDR) du dispositif qui a envoyé les commandes.

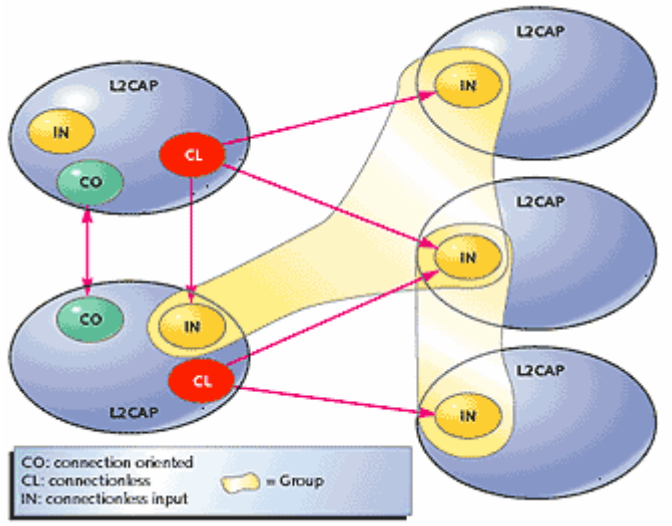


Figure I-14 : Les canaux L2CAP

a. Les étapes d'une connexion L2CAP

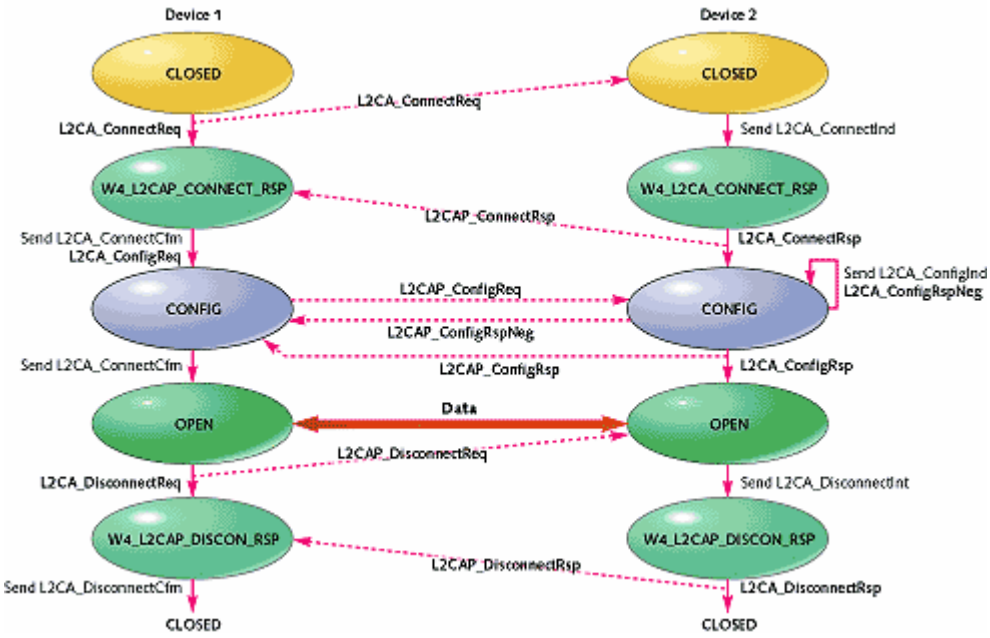


Figure I-15: Etat d'un canal L2CAP

Le transfert de données n'est possible que si les deux extrémités du canal L2CAP (orienté connexion) sont dans l'état OPEN. Ainsi, pour initialisé une connexion avec un dispositif éloigné, il faut d'abord configuré et connecté le point extrémité. Une requête de connexion est envoyé par le dispositif local ; Il entre ensuite dans un état d'attente de la réponse du dispositif éloigné W4_L2CAP_Connect_RSP . Une fois la réponse positive reçue, il s'agit maintenant de négocier les paramètres et les options de la transmission (maximum transmission unit (MTU), flush timeout, la qualité de service (QoS),…) c'est l'étape de configuration. Une fois tous ces paramètres négociés les deux entités sont alors à l'état OPEN et le transfert de données peut avoir lieu. La déconnexion a lieu à la demande de l'une des deux entités.

b. Fonctionnalités de la couche L2CAP

1) Multiplexage de Protocole

L2CAP doit soutenir le multiplexage de protocoles vu que le Protocole Base bande ne possède pas de champ ' type' pour identifier le protocole des couches hautes. L2CAP doit être capable de distinguer entre les protocoles des couche supérieurs comme le Protocole de Découverte de Service (SDP), RFCOMM et le Contrôle de Téléphonie.
En effet, pour le mode connecté c'est au niveau du paquet de signalisation " Connection Request " qu'un champ PSM Protocol/Service Multiplexor est réservé pour cet usage.
Tant dis que pour le mode sans connexion c'est au niveau du paquet de donnée qu'existe le champ PSM.

❖ *Mode connecté :*

Code=0x02	Identifier	Length
PSM		Source CID

FigureI-16 : Paquet de signalisation Connection Request

❖ *Mode sans connexion :*

Length	Channel ID (0x002)
PSM	Information
Information	

Figure I-17: Paquet de donnée L2CAP

PSM value	Description
0x0001	Service Discovery Protocol
0x0003	RFCOMM
0x0005	Telephony Control Protocol
<0x1000	RESERVED

Tableau I-1 : Description du paramétre PSM

2) Segmentation et Re assemblage

Comparé à d'autres médias physiques, les paquets de données définis selon le Protocole Base bande sont limités en taille. Une unité de transmission maximale (341 octets pour des paquets DH5) limite l'utilisation efficace de la largeur de la bande pour les protocoles de couche supérieurs qui sont conçus pour employer de plus grands paquets.

De grands paquets L2CAP doivent être segmentés dans des paquets Base bande plus petits avant leur transmission.

Plusieurs paquets Base bande reçus de la même façon peuvent être rassemblés dans un seul paquet L2CAP après un contrôle d'intégrité simple. La Segmentation et le Re assemblage (SAR) sont des fonctionnalités absolument nécessaires pour soutenir des protocoles employant des paquets plus grand que ceux soutenus par la couche Base bande.

3) Qualité de Service

Une implémentation L2CAP doit contrôler les ressources employées et s'assurer que les contrats de QoS sont honorés. Le processus d'établissement d'une connexion L2CAP permet l'échange d'informations quant à la qualité de service (QoS) attendu entre les deux unités Bluetooth. En effet, c'est au niveau du paquet " configuration request " et du " configuration responce " qu'un champ 'Options' est réservé à cet usage.

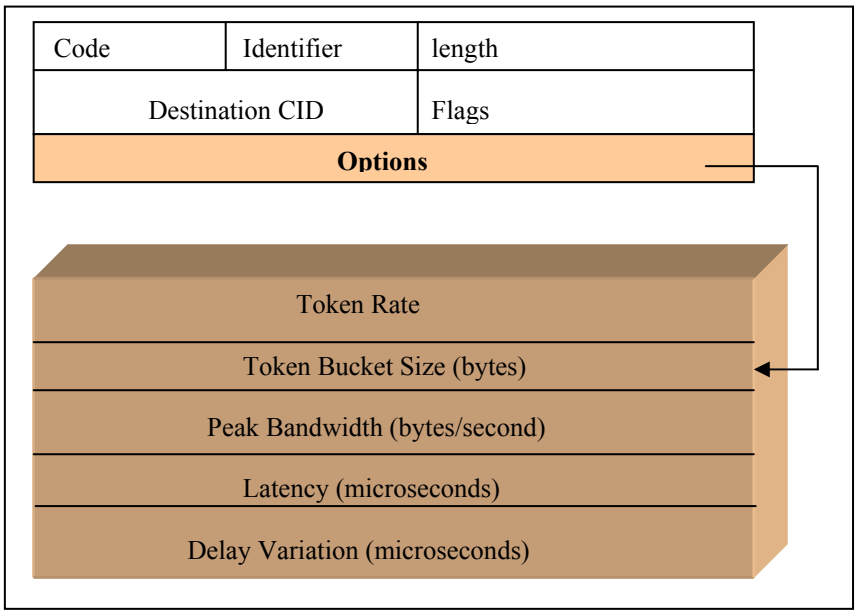


Figure I-18: Spécification de la QoS

3. Protocole RFCOMM

Le protocole RFCOMM est la base du remplacements des fil par Bluetooth. C'est un simple protocole de transport avec des fonctionnalités additionnelles comme l'émulation des 9 circuits des ports série RS-232 sur la partie L2CAP de la pile de protocole Bluetooth. Ceci est basé sur les standards TS 07.10 de l'ETSI et supporte une multitudes d'applications qui utilisent la communication par port série. RFCOMM fournit un transfert de donnée fiable, plusieurs connexions simultanées, le contrôle de flux et les caractéristiques d'une ligne série.
Deux dispositifs BT employant RFCOMM dans leur communication peuvent ouvrir plusieurs ports série émulées . RFCOMM soutient jusqu'à 60 ports émulés ouverts. **Un Data Link Connection Identifier** (DLCI) identifie un lien entre un client et une application serveur.
Le DLCI est représenté par 6 bits, mais les valeurs utilisable sont 2... 61. Le DLCI est unique dans une session RFCOMM entre deux dispositifs. À tout moment, il doit y avoir tout au plus une session RFCOMM entre n'importe quelle paires de dispositifs. En établissant un nouveau DLC, l'entité d'initiation doit vérifier si il existe déjà une session RFCOMM avec le dispositif distant, et si oui, établir le nouveau DLC sur celle ci. Une session est identifiée par l'adresse Bluetooth BD_addr des deux points extrémités.

Etude expérimentale sous Windows

Dans le chapitre précédent nous nous sommes intéressés a l'étude de la norme Bluetooth, nous allons maintenant entamer la partie expérimentale sous Windows. Pour cela nous avons besoin d'installer deux cartes Bluetooth chacune sur un PC, ainsi que des logiciels approprier pour permettre de gérer leurs communications.

I. la carte Xircom Bluetooth

Notre choix s'est porté sur l'utilisation de la carte Bluetooth xircom CBT également et plus simplement appelée *Bluetooth PC Card*, car elle figure dans la liste (limitée) des cartes PCMCIA compatible avec la pile de protocole Bluetooth sous Linux : " Bluez " ; et donc elle allait nous permettre dans un deuxième temps de réaliser l'implémentation de cette pile sur un PC Linux.

1. Caractéristiques techniques de la carte Xircom

La plage de fréquence de la carte s'étend entre **2,402 GHz** et **2,480 GHz.** Mais compte tenu des réglementations locales en vigueur, cette bande passante est réduite au Japon, en France et en Espagne. Cette particularité est gérée par un commutateur logiciel interne. Le nombre maximum de sauts de fréquence est de 1600 par seconde. La plage nominale de connexion va de 10 cm à 10 m.

Figure II-1: Carte Bluetooth Xircom CBT

2. Services disponibles

La carte Bluetooth Xircom prend en charge les profils Bluetooth suivants ainsi que les services et applications correspondants :

- ***Generic Access Profile*** (Profil d'Accès générique) - permet aux périphériques Bluetooth de se reconnaître et d'accéder les uns aux autres même lorsqu'ils ne partagent pas d'applications communes.
- ***Service Discovery Application Profile*** (Profil Application de détection de service) - permet aux applications de détecter les services et les fonctionnalités correspondantes disponibles sur les périphériques connectés
- ***Serial Port Profile*** (Profil Port série) - permet le remplacement du câble série pour des applications telles que les imprimantes et les assistants numériques communiquant via le port série.
- ***Dial-Up Networking Profile*** (Profil Accès réseau à distance) - applications de téléphonie sans fil et modem

- ***Fax Profile*** (Profil Fax) - applications de fax .
- ***LAN Access Profile*** (Profil d'accès LAN) - accès réseau par le biais d'un point d'accès Bluetooth/LAN prenant en charge le profil d'accès LAN
- ***File Transfer Profile*** (Profil Transfert de fichier) - à l'aide d'Intellisync for Notebooks File Transfer
- ***Synchronization Profile*** (Profil Synchronisation) - à l'aide d'Intellisync for Notebooks Synchronize

3. L'installation

Installer la carte Bluetooth de xircom n'est possible que sous Windows 98 & ME uniquement (ne fonctionne pas sous Windows NT4 et 2000).Cette carte par sa nature PCMCIA ne peut être utiliser directement sur un PC .En effet, elle est dédiée pour les PCs portables , et donc pour adapter son utilisation à un PC ordinaire nous avions besoin d'un adaptateur ISA (Industry Standard Architecture. Ancien format de bus de données et de connecteur sur la carte mère).Pour cela, Nous avions utilisé l'adaptateur ISA ORINOCO de Lucent qui avait servi dans des tests ssimilaires sur la norme 802.11.

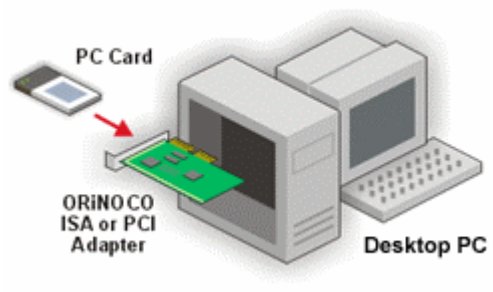


Figure II-2: Adaptateur ISA

Il a aussi fallut télécharger d'Internet les pilotes correspondants à notre adaptateur ISA pour que l'ordinateur puisse le reconnaître, connu respectivement sous le nom de ORINOCO PCI Adapter Software v5.02b et Firmware Update for ORiNOCO PC Cards v8.10.

Figure II-3 : Interface du driver de l'adaptateur ORINOCO

II. Blueview

L'utilitaire Xircom BlueView permet de configurer et de gérer les connexions et les relations entre les périphériques Bluetooth au moyen d'une interface d'exploration familière.
Avant de pouvoir communiquer, les deux périphériques Bluetooth doivent :

- se reconnaître mutuellement ;
- s'accorder pour communiquer ;
- s'accorder sur les conditions de communication ;
- déterminer quels services ou applications chacun d'eux a à offrir ;
- établir une connexion permettant l'utilisation des services.

L'interface BlueView permet de suivre toutes les communications Bluetooth transitant par la carte Xircom Bluetooth et les périphériques Bluetooth distants. Au moins un périphérique Bluetooth distant doit être installé et configuré avant de communiquer au moyen de la carte Xircom Bluetooth et du logiciel BlueView.
Pour cela il faut que le périphérique distant soit :

- sous tension ;
- dans la zone de couverture de la carte Xircom Bluetooth (moins de 10 mètres) ;
- configuré en mode "détectable" ;
- en mode "couplage" si la liaison entre ce périphérique et la carte Xircom Bluetooth doit être sécurisée.

1. Etablissement d'une connexion Bluetooth

L'établissement de la connexion entre deux équipements Bluetooth se fait en lançant manuellement cette détection : Pour cela , il faut sélectionner **New Devices in Range** au niveau de l'équipement maître (l'équipement maître est celui qui demande en premier l'établissement de cette connexion) et cliquer sur **Refresh** dans le menu Edit ou dans la barre d'outils correspondante. BlueView détecte et affiche tous les nouveaux périphériques Bluetooth (inconnus) situés à moins de 10 mètres et "détectables" (celles qui ont été configurés pour répondre aux interrogations d'autres périphériques ; les périphériques configurés comme "non détectables" ne sont pas affichés). Le nom, la classe du périphérique, l'adresse et le ou les services disponibles sont affichés. Le périphérique distant reste détectable pendant 2 minutes (ou jusqu'à ce que vous ayez cliqué sur Cancel dans la fenêtre Discoverable).

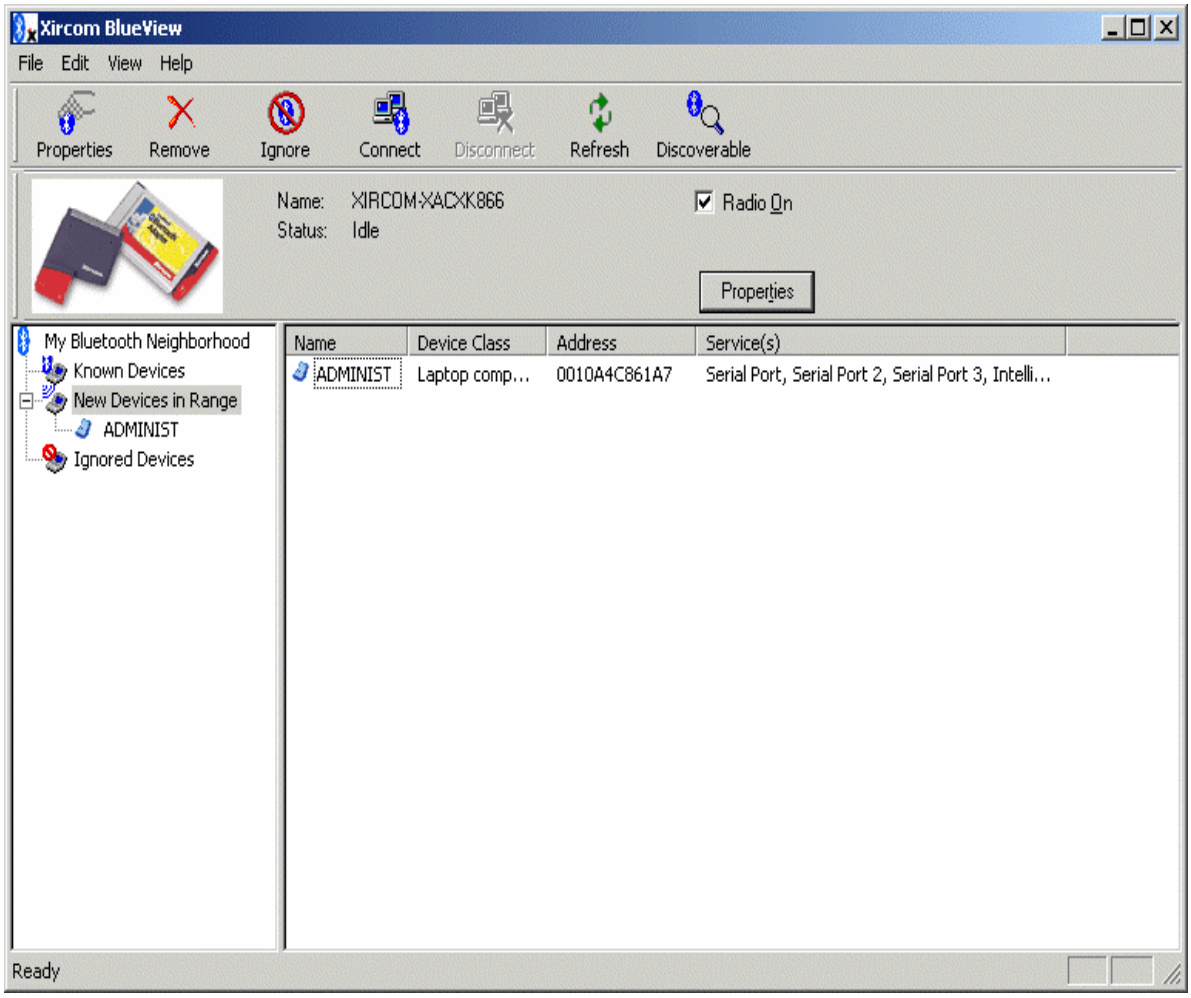


Figure II-4 : Equipement Bluetooth découvert

Une fois l'équipement distant détecté , l'initialisation de la connexion passe par une étape de configuration afin de s'accorder sur les conditions de communication et déterminer quels services ou applications chacun d'eux a à offrir : La fenêtre "Connect to remote Bluetooth device" (Se connecter à un périphérique Bluetooth distant) apparaît. Elle permet de décider si la connexion doit être permanente (cette connexion reste active tant qu'elle n'est pas modifiée ; si l'option Autoconnect est activée, la connexion est rétablie automatiquement à chaque fois que les deux périphériques sont dans la zone de couverture l'un de l'autre) ou temporaire One-time (cette connexion reste active jusqu'à ce que l'un des périphériques soit hors de portée ou jusqu'à ce que la connexion soit interrompue au moyen de la commande Disconnect) ou encore de déterminer si le périphérique à connecter nécessite une connexion sécurisée (cocher l'option Pair) on est alors invité à entrer un mot de passe sur le périphérique local. Un mot de passe correspondant doit être entré sur le périphérique distant pour que les deux périphériques soient "couplés". Les futures connexions seront authentifiées par les périphériques au moyen de ce mot de passe, sans intervention de l'utilisateur.

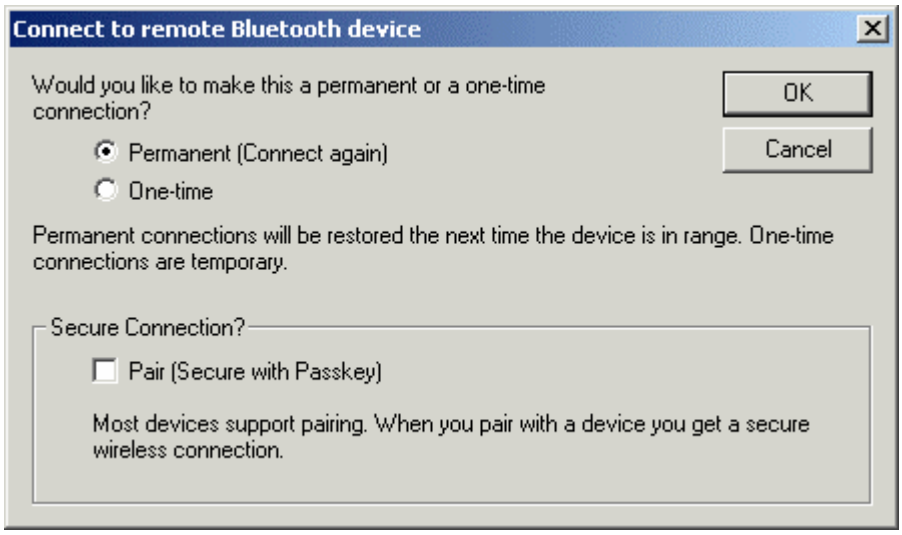


Figure II-5 : Propriétés de la connexion

Enfin, il faut sélectionner un service du périphérique distant. Le périphérique distant passe de "New Devices" à "Known Devices" et le périphérique local figure parmi les périphériques connus (Known Devices) sur le périphérique distant.
Une fois la connexion établie, on peut alors utiliser le service sélectionné sur le périphérique distant.

2. Propriétés du périphérique Bluetooth

Une fois notre connexion est établit on peut visualiser les informations relatives à notre connexion et ceux de l'équipements local et distant.

a. Propriétés du périphérique Bluetooth local

- ***Name* (nom)** : nom attribué par l'utilisateur au périphérique Bluetooth local.
- ***Address*** - adresse hexadécimale : adresse du périphérique Bluetooth. Cette option peut être utile lors de la détection de périphériques.
- ***Device Class* :** type du périphérique Bluetooth.
- ***Service Type(s)*** : services pris en charge par ce périphérique.
- ***Connection Profile*** (profil de connexion) :

  Discoverable Mode - Discoverable or Non-Discoverable (mode détection - détectable ou non détectable).

  Pairable Mode - Bondable or Non-Bondable (mode couplage activé ou non). Permet de coupler votre périphérique à un autre. Cette liaison offre un surcroît de sécurité.

  Security Mode - Encryption On Or Encryption Off (Mode sécurisé, chiffrement activé ou désactivé). Le chiffrement renforce la sécurité des données transmises.

Connectable Mode - Connectable or Non-Connectable (mode 'connectable' activé ou non). Permet de connecter votre périphérique à tout périphérique Bluetooth dans la zone de couverture.

Authentication - Passkey Required or Passkey Not Required (Authentification, mot de passe requis ou non). Mesure de sécurité supplémentaire exigeant la saisie d'un mot de passe identique sur les deux périphériques avant de permettre la transmission de données.

Device Polling - Off or On (interrogation des périphériques, désactivée ou activée). Indique au périphérique s'il doit rechercher d'autres périphériques Bluetooth dans la zone de couverture, et avec quelle fréquence.

Autoconnect : lorsque cette option est activée, tout périphérique Bluetooth pour lequel une connexion permanente a été établie avec votre carte Xircom Bluetooth se reconnecte automatiquement lorsqu'il arrive dans la zone de couverture.

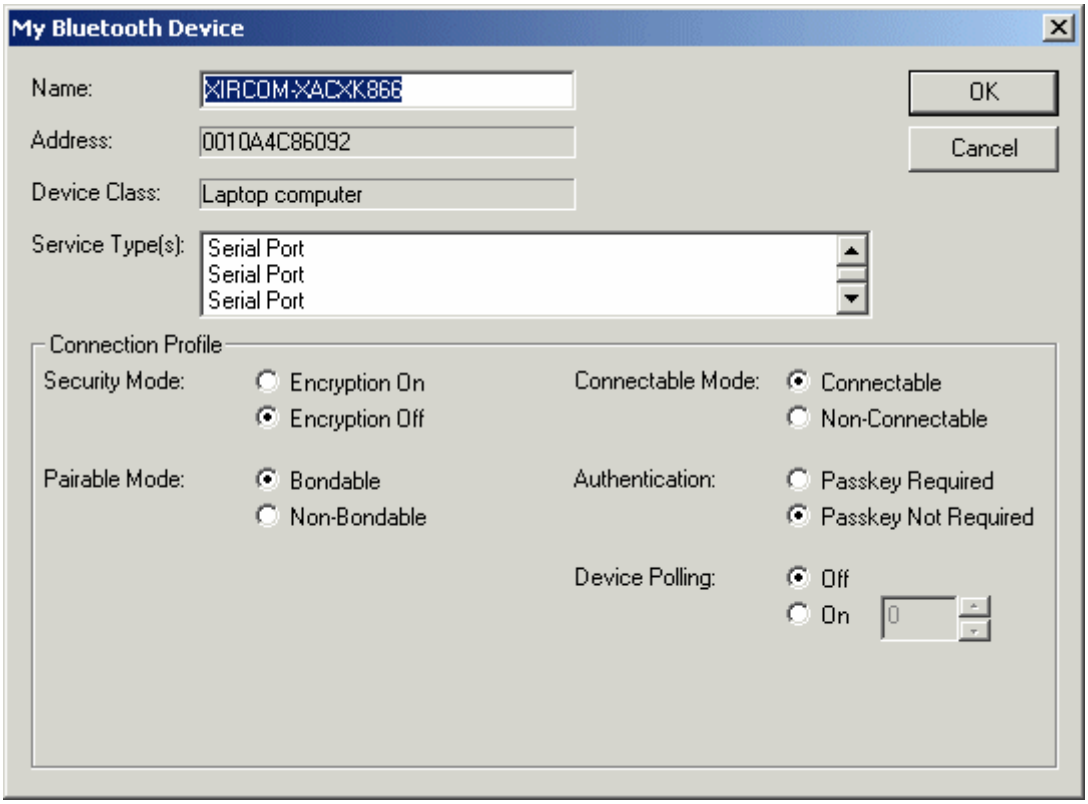


Figure II-6 : Propriétés du périphérique local

b. Propriétés du périphérique distant

Dans la fenêtre principale de BlueView, on sélectionne un périphérique Bluetooth dans la liste Known Devices et on clique sur Properties dans le menu Edit pour afficher les informations relatives à ce périphérique.

- ***Name*** **(nom)** : nom attribué par l'utilisateur au périphérique Bluetooth .
- ***Address*** **-** adresse hexadécimale : adresse du périphérique Bluetooth.
- ***Device Class*** **:** type du périphérique Bluetooth.

- ***Service Type(s)*** : services pris en charge par ce périphérique.
- ***Status*** - Active or Inactive (État, actif ou inactif).

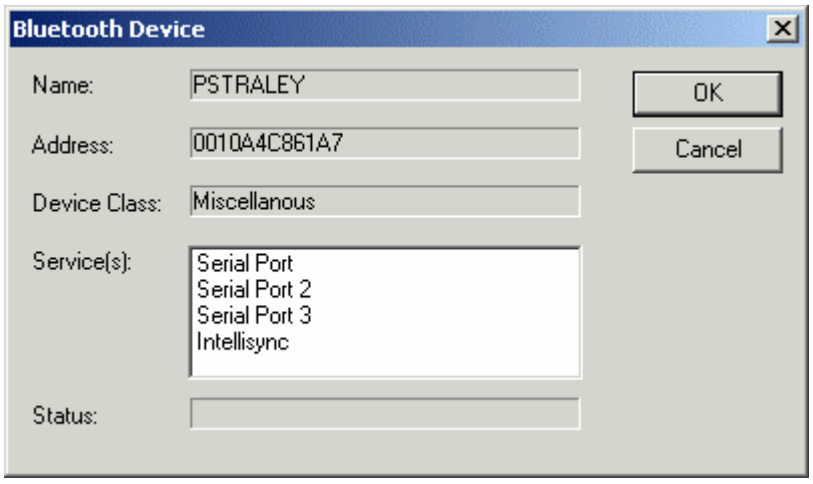


Figure II-7 : Propriétés du périphérique distant

III. Mesures et tests

Le but essentiel de ces tests est une approche pratique d'un réseau Bluetooth et plus précisément une première évaluation des performances de la carte xircom dans l'environnement Windows.
Ces tests sont effectués dans le laboratoire de l'INT au 4éme étage du département RST (Réseaux et Systèmes de Télécommunication). On a utilisé pour cet effet un PC et un ordinateur portable tout deux Windows 98 équipés de cartes Xircom. Les paramétres de tests sont essentiellement la direction du transfert, la distance entre les équipements Bluetooth et la mesure du débit.
Afin de garantir l'exactitude de nos mesure, notre premier soucis fût d'éliminer tout équipement pouvant être source d'interférence. En effet, Ce laboratoire contient des équipements 802.11 , or un réseau sans fil 802.11 est basé sur une architecture cellulaire et où chaque cellule BSS est contrôlée par une station de base appelée Access Point ou AP qui émettent dans la même bande de fréquence que Bluetooth (2.4GHz). Mais, la réglementation de l'utilisation de cette bande de fréquence en France fait que l'émission des équipements 802.11 est limité à la plage **2.399**GHz –**2.452**GHz et celle de la carte Bluetooth à la plage du **2.46**GHz à **2.483**GHz . Toute fois, il est à noter que Bluetooth possède un mécanisme intéressant de lutte contre les interférence appelé saut de fréquence ou encore Frequency hopping au cours du quel la porteuse change de fréquence à un rythme de 1600 sauts par secondes.

1. Distribution spectrale de puissance

Notre première expérience fût l'analyse du signal émis par la carte Bluetooth . Ceci fût possible grâce à l'analyseur de spectre le HP 89440A disponible dans le laboratoire d'électronique de l'INT. et d'une petite antenne branché au connecteur INPUT de ce dernier.

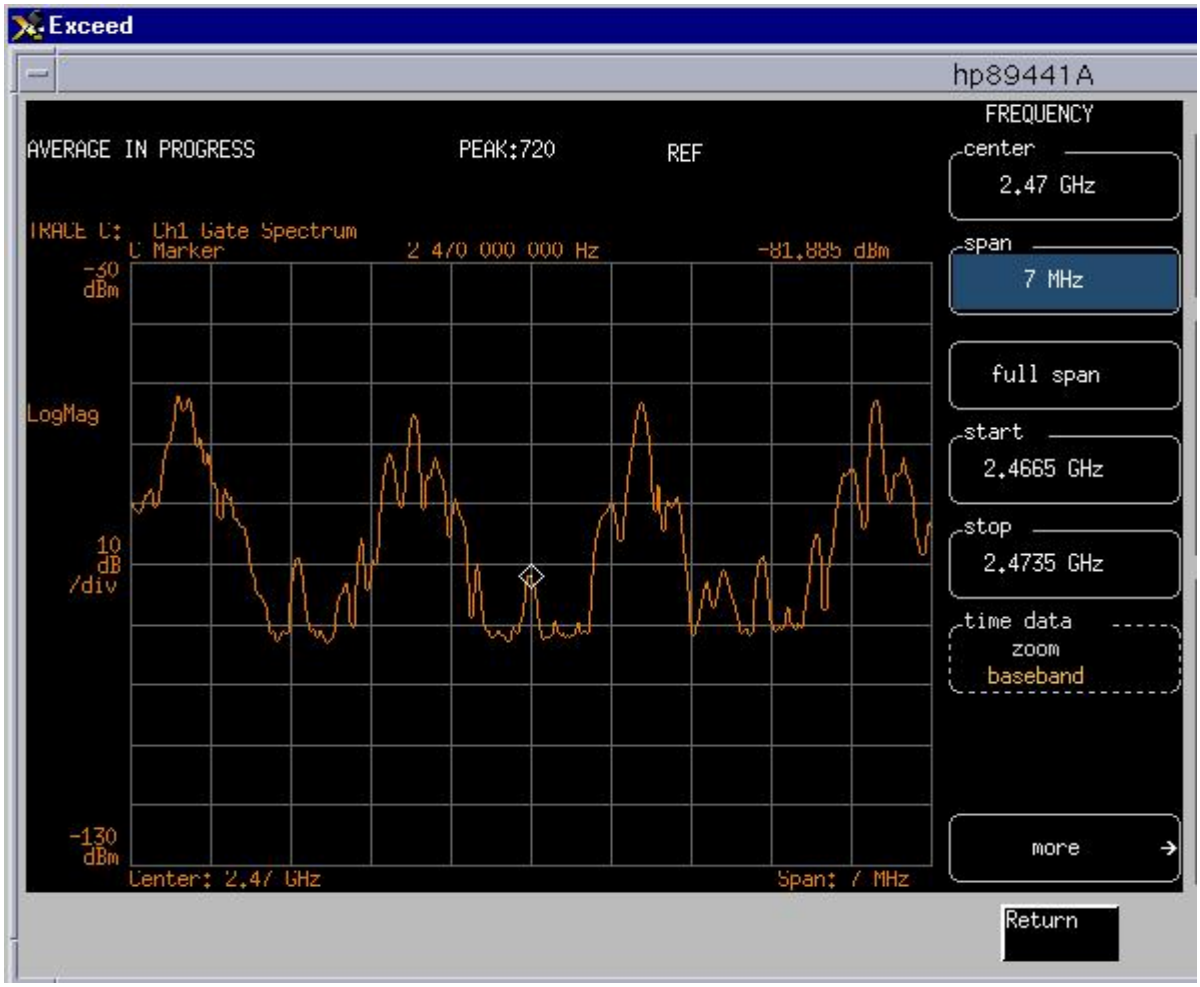


Figure II-8: Distribution spectrale d'un signal Inquiry

- ***Interprétation :***

Après une étape de réglage et de configuration de l'analyseur, on a pu observer la distribution spectrale d'un signal Inquiry émis par l'unité Maître à la recherche d'esclaves .Des pics de puissance défilaient sur l'écran dans la bande de fréquence comprise entre le 2.46 et le 2.48GHz.

Dans la figure ci dessus on présente un Zoom de ce que l'on a observé au cours de notre manipulation. La fréquence centrale de l'analyseur étant réglée sur la valeur de 2.47GHz, il effectue un balayage de la plage de fréquence comprise entre 2.4665GHz et 2.4735GHz.

Ces pics représentent les sauts de fréquences de l' unité Maître au cours du mécanisme du Frequency Hopping.

Fréquence en

	GHz
Pic1	2,46685
Pic2	2,46895
Pic3	2,47105
Pic4	2,47315

Tableau II-1: Pics de fréquence

2. Temps de connexion

Dans un deuxième temps, on s'est intéressé à la mesure du temps nécessaire à l'établissement d'une connexion entre un maître et un esclave : Une connexion est établie par un message de type PAGE si l'adresse de l'unité à connecter (unité Esclave) est connue ou alors un message de type INQUIRY (demandant à toutes les unités de répondre) suivi d'un PAGE si l'adresse n'est pas connue. Au cours de cette expérience nous avons utilisés un portable et un PC équipés de cartes xircom. Le logiciel Blueview comme précédemment décrit nous donne la main pour lancer ce genre de commande.

		Temps d'une connexion avec un Inquiry (en seconde)	Temps d'une connexion avec un Page (en seconde)
Distance 0.5m	**Mesure1**	14,94	2,32
	Mesure2	14,22	2,25
	Mesure3	14,32	2,82
	Moyenne	14,4933	2,46333
Distance 3.6m	**Mesure1**	14,47	2,4
	Mesure2	14,72	2,31
	Mesure3	14,75	2,46
	Moyenne	14,646	2,39
Distance 8.3m	**Mesure1**	14,18	2,91
	Mesure2	15,19	3,14
	Mesure3	15,02	2,98
	Moyenne	14,796	3,01

Tableau II-2: Temps d'un Inquiry/ Page

- ***Interprétation :***

En se référant à la norme ces résultats semblent très proche de la théorie. En effet, le temps d'une connexion à travers un Inquiry est estimé à12,84s et celui d'un Page est estimé à 2,6s.

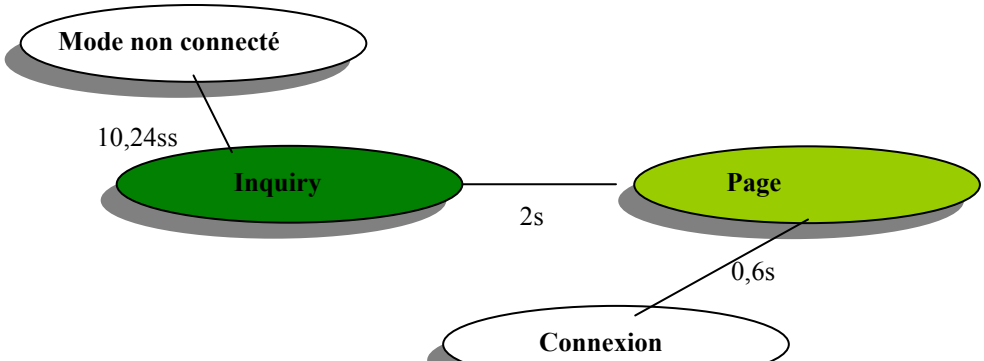


Figure II-9 :Délai d'établissement d'une connexion (la norme)

De plus on remarque que le temps d'établissement d'une connexion est quasiment indépendant de la distance, ce qui est conforme aux spécifications de la norme. En effet, dans l'état Inquiry, le maître émet une suite de paquets envoyés sur la séquence de 32 fréquences.Ces 32 fréquences sont divisées en deux groupes, A et B.

- Le groupe A contient les fréquences f(k-8), f(k-7),….f(k),…..f(k+7)
- Le groupe B contient les fréquences f(k-15), f(k-14),..., f(k-9),…f(k+8)

K est déterminé par l'horloge du maître de l'Inquiry.

Le maître commence par émettre sur les fréquences de la série A. Il utilise 256 fois au minimum cette série. Après cela, il passe à la série B, qu'il parcourt aussi au minimum 256 fois. Un Inquiry contient au total 4 séries A et B parcourues alternativement.

Une procédure Inquiry dure 10.24 s à moins de recevoir assez d'adresses et de décider d'arrêter l'Inquiry.

Durée d'une série, : tsérie=625µs * 16 = 10ms

Chaque série est répétée 256 fois tsérie*256 = 2.56s

Un Inquiry comporte 4 séries 2.56 *4 = 10.24s

Le message Page ressemble très fortement au message Inquiry sauf que ce dernier nécessite un train de paging supplémentaire pour collecter toutes les réponses. Ce qui fait qu'un temps de procédure page se limite à 2.56 s.

3. Mesure de débit

Dans un troisième temps, nous nous sommes intéressés aux performances de la carte en terme de débit. Les mesures de débit sont ceux d'un transfert de fichier de 4.94Mb (une image bmp). On a utilisé à cet effet l'application de transfert de fichier disponible avec le logiciel Intellisync. La connexion entre le maître et l'esclave se fait donc sur le port virtuel du service Intellisync.

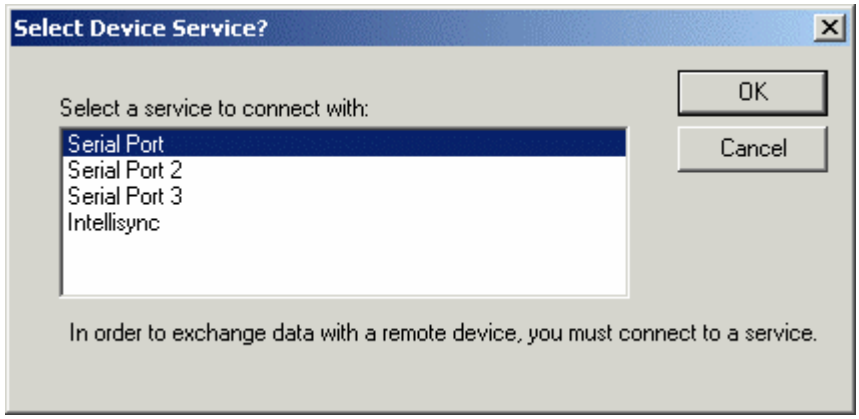


Figure II-10: Services disponibles pour notre équipement

- **Utilisation d'Intellisync avec la carte Xircom Bluetooth**

Intellisync permet de synchroniser et de transférer des données rapidement et facilement entre des PCs et des notebooks équipés de la technologie Bluetooth. En outre, Intellisync permet de :

- synchroniser les lecteurs et les répertoires de deux ordinateurs
- synchroniser les données entre des gestionnaires de données personnelles
- copier, déplacer et supprimer des fichiers sur un système distant à l'aide d'une connexion Bluetooth
- définir et contrôler les droits d'accès d'un système distant à vos données

Le tableau suivant résume les mesures effectués :

	Distance en m	0.5	1.2	3.6	8.3	16
Débit en Kb\s	Mesure1	154	150,262	107	34,82	0
	Mesure2	153,87	148,1	104,37	36,092	0
	Mesure3	153,7	150,3	107,7	35,2	0
	Mesure4	154,24	151,21	107,8	34,56	0
	Moyenne	153,95	149,96	106,71	35,16	0
	Ecart Type	0,197	1,143	1,389	0,579	0

Tableau II-3 : Débit Moyen de la carte Xircom

Ces résultats ont été reporté sur la courbe ci dessous afin de permettre une meilleure analyse de l'influence de la distance sur le débit offert. En particulier le traçage de la courbe de tendance permet d'avoir une approximation mathématique du comportement de notre équipement Bluetooth.

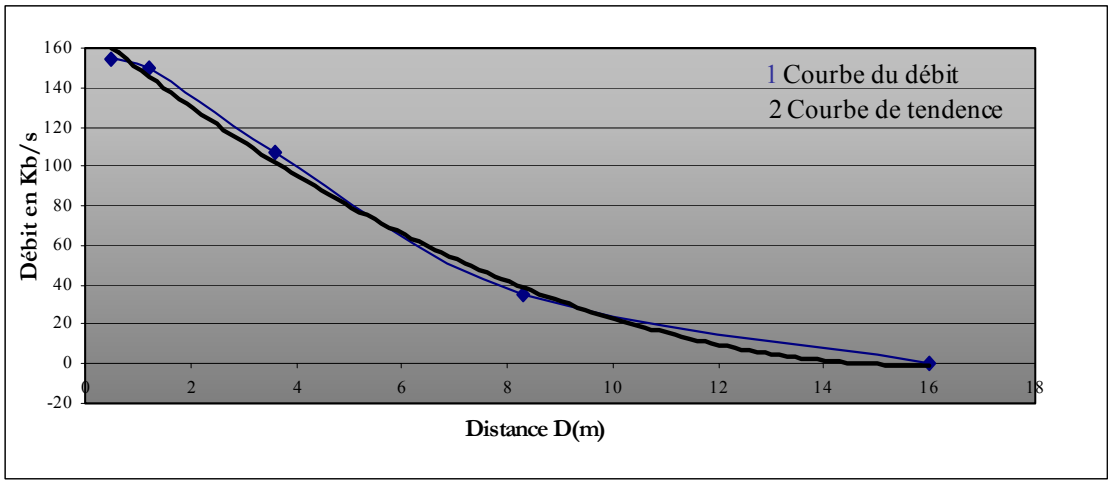


Figure II-11: Variation du débit de transfert en fonction de la distance

- ***Interprétation :***

En principe, une carte fonctionnant en mode synchrone, comme c'était notre cas lors du test, est censée offrir du 432 Kbits/s. Or on est très loin, La valeur maximale de débit dans notre

cas est 153 Kb\s mesuré à une distance très faible (50 cm), on est bien au dessous de la moitié.
L'équation de la courbe de tendance décrit une fonction décroissante sur l'intervalle [0,15] ; en tenant compte des spécifications de la norme quand à la porté maximale des équipements qui est fixé à 10m, on peut affirmer que le débit décroît considérablement en fonction de la distance.
De plus il on a remarqué que dans notre cas la distance à partir de laquelle , il n'y a plus de bonne détection de l'esclave est de l'ordre de 15m(Figure.**II-12**) Et celle au bout de laquelle la détection n'est plus possible est de 16m (>10m).

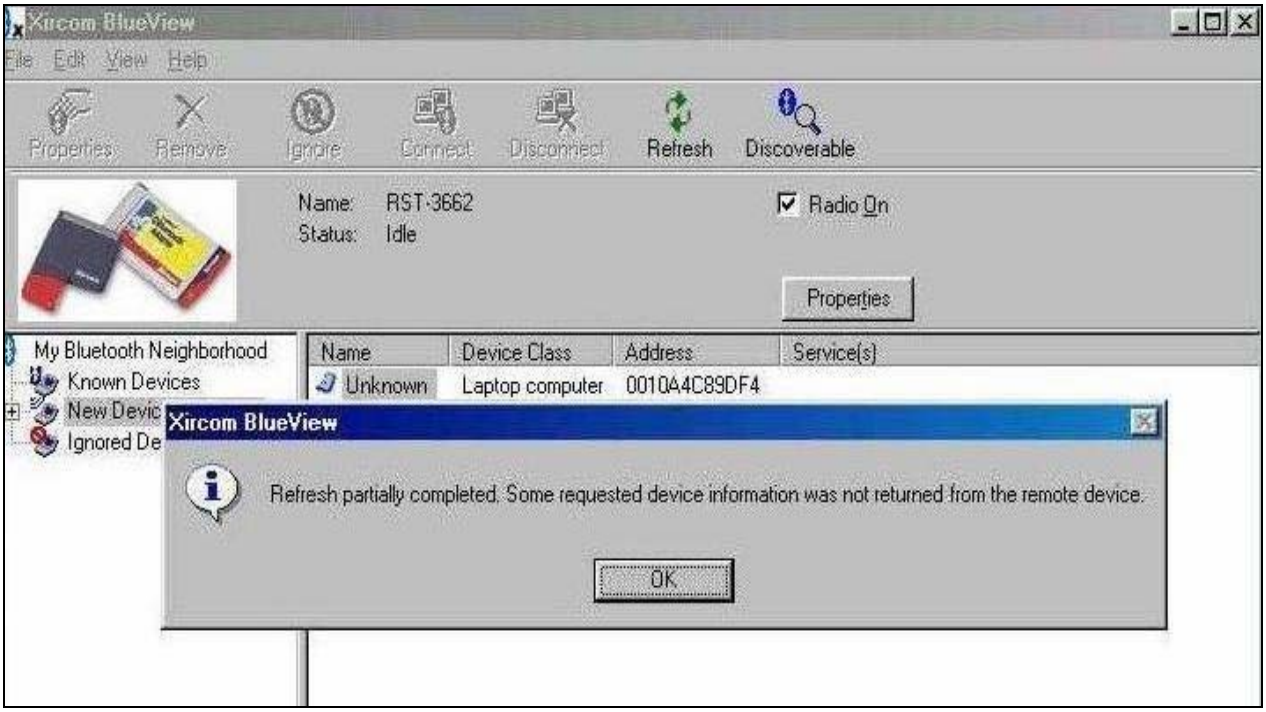


Figure II-12 : Connexion Maître\Esclave incomplète (15 m)

Il est aussi important de souligner que nous avons constater que ces mesures ne sont possibles que dans un milieu ouvert ; c'est à dire en absence d'obstacles tel que des mûrs. En effet, les deux cartes n'arrivent pas à se voir lorsque chacune d'elle est placé dans une chambre avec une séparation métallique entre les deux.
Comme on le voit, la carte ne répond que partiellement à nos attentes. oui, ça marche. oui, c'est simple à installer. Mais c'est aussi très lent. Cependant, le débit mesuré est largement satisfaisant pour le partage d'une connexion Internet en RTC, ou pour le transfert de fichier mais surtout, son point fort est évidemment la connexion sans fil. Solution très mobile, elle permettra à terme de communiquer avec n'importe quel appareil bluetooth sans aucune configuration ou drivers à installer .

4. Comparaison avec le kit BlueTake

Dans un cadre plus large et afin d'avoir une meilleur évaluation des performances de la technologie Bluetooth, une étude comparative a été mené entre les caractéristiques techniques de notre carte xircom et celle des puces Bluetooth du kit de développement BlueTake USB Dongle. Il s'agit d'un kit comprenant deux émetteurs/récepteurs équipés de puces bluetooth, se connectant sur un port USB. Tous deux sont équipée d'un câble de 50 cm et mesure 70 * 40 *11 mm, ce qui en fait de petits appareils peu encombrant et facilement transportables.

Bien évidemment, le test en lui-même, ne repose pas sur les possibilités de dialogue entre les deux appareils BlueTake, mais sur le débit de données transmissible.
La courbe ci dessous illustre les résultats mesurés lors du transfert d'un fichier de 5 Mo en plaçant les puces à 50 cm l'une de l'autre, puis à 5 m.

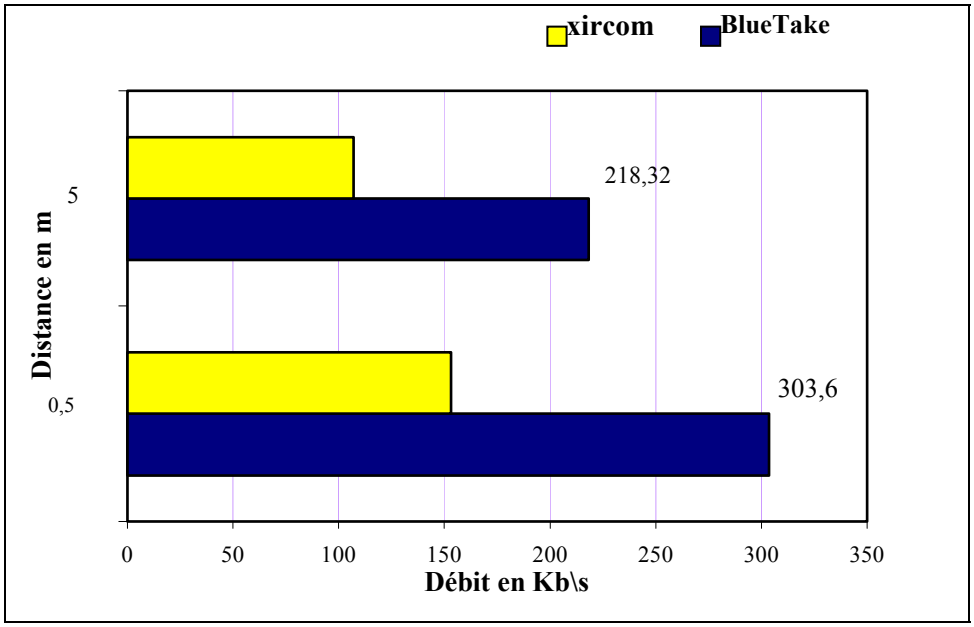


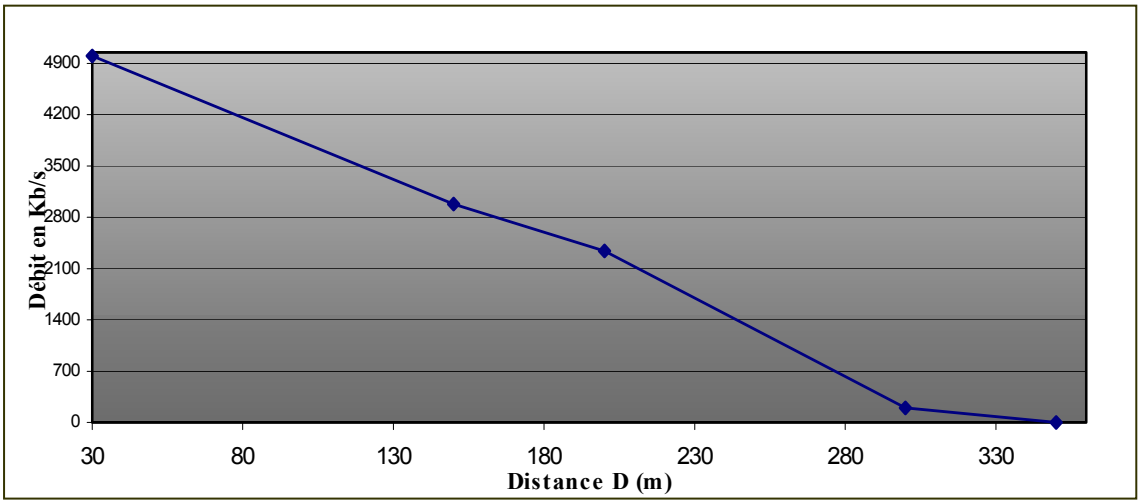
Figure II-13 : Taux de transfert de fichier en Kb\s

- ***Interprétation :***

Cette courbe comparative montre la dépendance des performances de la technologie Bluetooth sous Windows au matériel utilisé et au constructeur. Avec la panoplie de produits qui existent aujourd'hui sur le marché, le choix du matériel à utiliser dépend des exigences des applications à faire fonctionner dessus et des performances souhaitées par l'utilisateur. Toute fois, il est intéressant à remarquer que même si le débit atteint par ces puces BlueTake est le double de la carte xircom, on est encore loin des spécifications de la norme qui prévoit la valeur de 432Kb/s dans le cas d'un transfert ACL. Cette expérience montre que si on désire une évaluation plus précise de la technologie Bluetooth, il serait plus intéressant de passer au système d'exploitation Linux pour ne pas rester esclave des caractéristiques du logiciels sous Windows.

IV. Conclusion

En terme de débit, il est donc clair que Bluetooth ne fait pas le poids face à la norme 802.11b , une norme concurrente autorisant déjà des débits pouvant atteindre 11 Mbits /s, soit à peu près autant que ce que fournira Bluetooth dans sa version 2.0, à venir à une date encore inconnue .



802.11b offre aujourd'hui une bande passante cinq fois plus importante ainsi que des portées plus élevées que Bluetooth, ce qui rend cette technologie plus indiquée dans une utilisation en réseau (les gros réseaux locaux d'entreprise et les environnement informatiques) . En effet, Il est évident que pour interconnecter sans fil un certain nombre de PC entre eux et à Internet, la technologie à utiliser est WiFi puisque cette derniére est une extension naturelle de Ethernet 10 Base T, et assure un débit qui lui correspond.

Mais cet avantage a un coût : plus puissants les émetteurs récepteurs Wi-Fi sont plus gros et consomment beaucoup plus que leurs homologues Bluetooth. Wi-Fi n'est donc pas une solution adaptée pour les outils nomades (Palm ; Pocket PC et autres GSM…).

Cependant, si Bluetooth est limité en débit et en distance de propagation, cette technologie offre l'avantage d'une faible consommation d'énergie grâce à la compacité de la puce. De plus le prix de cette solution est moins élevé. Ainsi, pour connecter des outils nomades, entre eux, et à Internet, la technologie à utiliser est Bluetooth. De plus, Les outils nomades, pour la plupart, ont besoin de supporter de l'audio. Hors Ethernet et TCP/IP ne savent pas gérer la communication de la voix aussi bien que Bluetooth.

Ainsi, de par leurs applications différentes, les technologies Bluettoth et Wi-Fi se révèlent complémentaires et non concurrentes.

Bluetooth et Linux

Le développement assez récent de Linux et du logiciel libre et la rapidité croissante avec laquelle ils infiltrent certains milieux auparavant réservés à des solutions hors de prix montrent à quel point ils sont en passe de devenir les supports majeurs du développement de l'informatique.

La liste des secteurs où les logiciels libres et Linux ont supplanté des solutions payantes serait aujourd'hui infinie. Parmi ces exemples on peut citer le fait que Linux est en train de supplanter son concurrent Windows sur le marché des assistants personnels et autres téléphones portables nouvelle génération. Malgré que, dans les premières étapes du raz-de-marée du sans fil, les fabricants des PDAs et des téléphones portables contournaient généralement Linux en faveur d'autres systèmes d'exploitation tel que Windows CE vu que la plupart des applications sans fil exigent des systèmes d'exploitation temps réel ; ce qui n'est pas le fort de Linux.

Mais, à l'autre bout de la plus part de ces liaisons sans fil il existe un point d'accès - un serveur, avec des utilisateurs mobiles. Or on emploi ces serveurs sans fil pour effectuer des opérations d'autorisation, d'authentification, de facturation, et de roaming. C'est là où Linux est imbattable. C'est pourquoi récemment des sociétés commerciales tel que IBM, Nortel, Ericsson et Nokia ont fait de Linux le centre d'intérêts de leurs recherches.

Ainsi, afin de stimuler le développement des réseaux sans fil, et plus précisément celui de la technologie Bluetooth, plusieurs constructeurs ont essayé de faire fusionner cette dernière avec l'une des technologies les plus performantes du vingtième siècle: Linux.

Parce que le système d'exploitation est source ouverte, les fabricants peuvent rapidement personnaliser son code source en fonction de leurs besoins. Plusieurs piles de protocoles Bluetooth ont vue le jour, parmi lesquelles OpenBT, BlueDrekar, Bluez… citant toujours la plate-forme Linux comme étant le choix évident, vu sa flexibilité, sa puissance et sa popularité croissante.

Aujourd'hui, Linux est le premier OS (Operating System) à avoir un support officiel pour Bluetooth. En effet, la pile de protocole **Bluez** fait maintenant parti intégrante du Kernel depuis sa version 2.4.6. C'est la raison pour laquelle nous avons opté pour l'utilisation de cette dernière.

I. Bluetooth en logiciel

Le plus simple des schémas d'implémentation de la pile de protocole Bluetooth est d'intégrer le tout sur du Hardware, c'est à dire sur une simple et petite puce qui prend tout en charge.
Ceci semble une solution idéale qui permet une abstraction de la complexité de Bluetooth vis à vis du monde extérieur, pour qui Bluetooth est un moyen et non une finalité. Ce qui est le cas de la plus part des produits électroniques commercialisés (Toolkit, Notebook, ...).

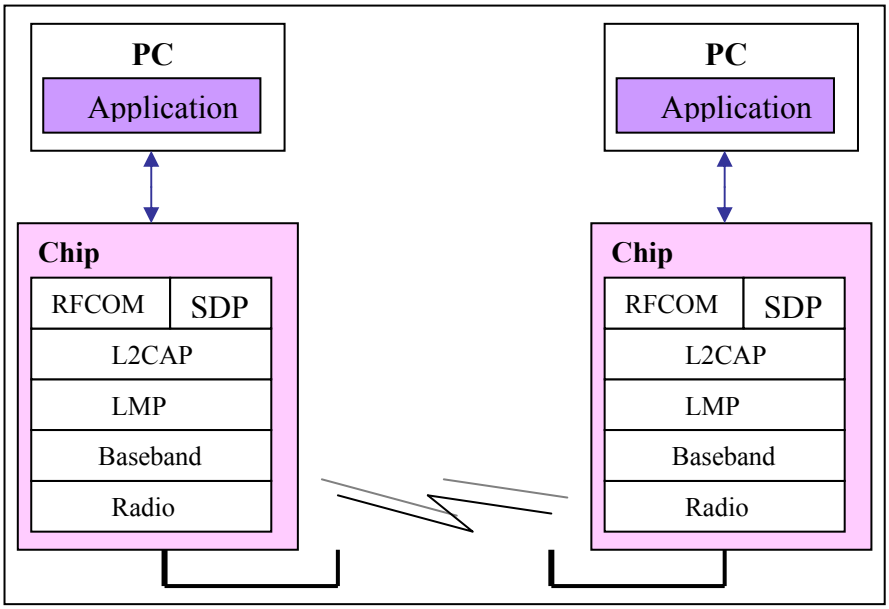


Figure III-1: Implémentation de la pile de protocole dans une puce

Mais dés lors que nous nous intéressons un peu plus à cette boîte noire (puce), aux mécanismes qui régissent son fonctionnement et aux performances souhaitées, il devient pertinent de penser à une implémentation en software. Ceci ne pose aucune difficulté lorsqu'un module Bluetooth est intégré dans un PC, un PDA ou tout autre type de matériel disposant d'un processeur sur lequel on peut faire tourner la partie haute de la pile Bluetooth. En revanche, dans le domaine de l'embarqué et des produits électroniques, il est difficile (voire impossible) de modifier le logiciel embarqué sur le processeur host (pour cause de ressources limitées, d'absence de système d'exploitation dans certains cas ou du fait qu'il n'est pas toujours possible de flasher la ROM).

.

Figure III-2 : Implémentation de la pile sur du logiciel

Une connexion Internet sur un ordinateur banalisé de type Windows ou Linux nécessite un adaptateur réseau, un driver et la pile TCP/IP . De même pour Bluetooth , on a besoin d'un adaptateur Bluetooth, d'un driver et d'une pile de Protocole Bluetooth. Comme la plus part des équipements Bluetooth se présentent comme des équipements série (port UART), on n'a donc souvent pas besoin d'un driver. Pour Linux , plusieurs piles Bluetooth sont disponibles sur le marché parmi les quelles :

- OpenBT

OpenBT fût la première pile Bluetooth sous Linux. Elle a été à l'origine développé par Axis Communications selon la spécification 1.1 de la norme .Elle est maintenant disponible en tant que projet source ouverte accueillit chez Sourceforge.

- BlueDrekar

BlueDrekar est un projet d'IBM qui inclut une pile complète de Bluetooth. Mais, actuellement seulement le Driver de transport HCI UART est source ouverte.

II. BlueZ

BlueZ est la pile officielle de Bluetooth sur Linux. Elle a été à l'origine développée par Qualcom. Il s'agit d'un projet source ouverte « Open Source » distribué sous une licence GNU General Public License (GPL). Concrètement, cela signifie que les codes sources et la structure interne de la pile est disponible au public. Ainsi, toute personne intéressée peut l'utiliser, l'améliorer et la modifier : l'usager hérite donc d'une liberté qui n'existe pas dans le modèle traditionnel du développement de logiciel. La principale conséquence est de transférer le développement du logiciel dans les mains de la communauté des usagers. BlueZ peut être employé avec des dispositifs connectés sur le port USB ou l'interface série du PC. De plus, Bluez fournit un dispositif HCI virtuel le « Vhci » qui permet de simuler une interface HCI et donc de tester des applications Bluetooth sans avoir besoin d'un matériel réel. Bluez offre donc une architecture indépendante du support matériel, mais encore modulaire.

1. Architecture de Bluez :

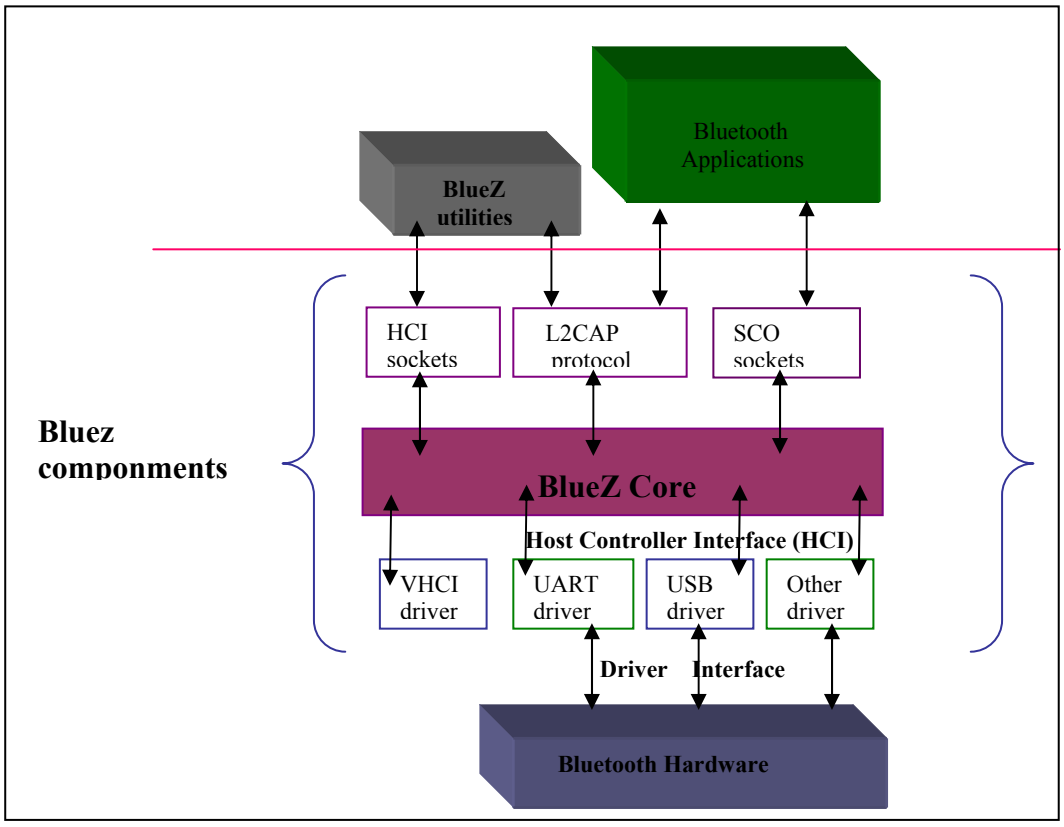


Figure III-3: Composants de Bluez

Bluez est constitué des couches suivantes :

- Un noyau HCI '*HCI Core*' qui remplit les fonctionnalités de la couche HCI. Cette dernière n'est présente que dans le cas où la couche L2CAP est implémentée en software. Il s'agit de l'interface qui régit les communications avec les couches basses résidants dans le Hardware. Ce qui permet de simplifier le schéma de communication: en effet, dans ce cas L2CAP peut directement accéder au protocole Basebande sans intervention de la couche LMP.

- Des *drivers HCI* qui jouent le rôle d'interfaces avec le hardware : un module HCI UART pour les périphériques Bluetooth avec une interface série, un module HCI USB pour ceux ayant un port USB et le module de gestion de périphérique virtuel *Vhci* qui permet comme son nom l'indique de simuler une interface HCI.

  Dans notre cas et vu que nous possédons une carte PCMCIA, nous allons nous intéresser plutôt au module HCI UART.

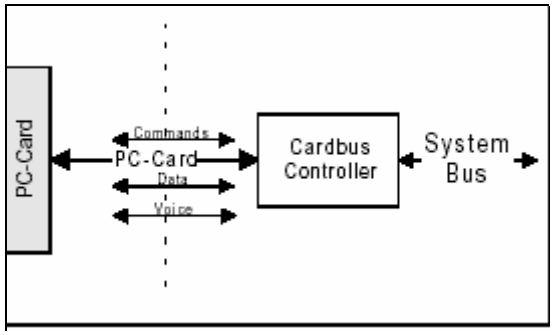


Figure III-4: UART HCI DRIVER

- Le '*L2CAP Module*' qui est un module implémentant le protocole L2CAP (logical Link Control and Adaptation Protocol). Comme précédemment décrit au niveau du premier chapitre ce protocole fournit la couche transport en mode connecté ou non connecté. Il permet d'interagir avec les applications et d'effectuer des opérations de multiplexage de protocoles, de segmentation et de ré assemblage . Cette couche ne procure aucune fiabilité et utilise l'ARQ au niveau de la couche bande base pour assurer un minimum de fiabilité. Des identifiants des canaux sont utilisés pour repérer chaque point de connexion.

- Des utilitaires de configuration et de test dans l'espace utilisateur :exemple **hciconfig** et **hcid**.(Voir Annexe)

  La figure ci dessous illustres le principe de fonctionnement globale de la pile de protocole dans le cas où le matériel Bluetooth exemple la carte est connectée sur le port USB.

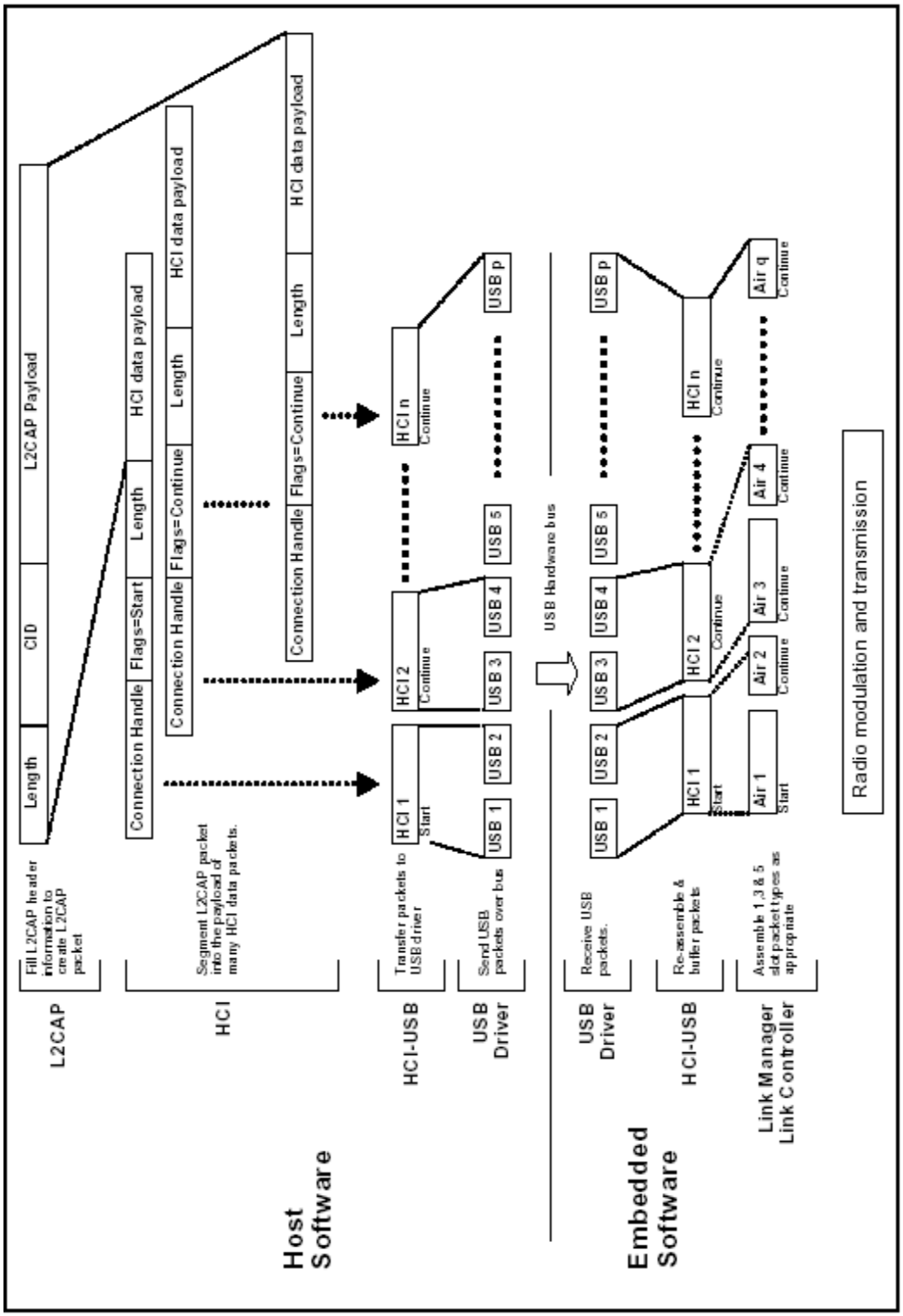


Figure III-5: Principe de fonctionnement de la pile de protocole

2. Installation :

Pour pouvoir installer et par la suite tester les performances de la pile de protocole Bluez, la première étape à franchir est l'installation de notre carte xircom sur notre PC Linux.
Nous avons eu besoin de l'adaptateur ISA de ORINCO, la première difficulté qui surgit dés lors est la détection de notre carte PCMCIA par le PC et ceci indépendamment du faite qu'il s'agisse d'une carte Bluetooth ou 802.11 ou autre. Un PC Linux reconnaît une telle carte si au moment de l'insertion de celle ci, il émets deux bipes successifs. Ce qui ne fût pas notre cas.
Pour palier à ce problème, il fallait ajouter des modules responsables de la gestion des interfaces PCMCIA sur un PC Linux mais aussi et surtout qu'ils soient compatible avec notre version du kernel. Or, sachant que Bluez fait parti intégrante du noyau depuis sa version 2.4.6 nous avons opté dés le départ pour un kernel dont la version était assez récente 2.4.18.

Ainsi et en tenant compte de toute ces contraintes, nous avons utiliser le package pcmcia-cs-3.1.29. Après installation de ce dernier (voir Annexe), il était devenu possible d'insérer les modules requis: i82356, ds, serial_cs ; et d'initialiser la détection de notre carte PCMCIA en lançant le script suivant: **/etc/rc.d /init.d/pcmcia start**. le PC arrive finalement à reconnaître notre carte xircom et on peut vérifier son bon fonctionnement grâce au script **cardmgr status** (voir Annexe).
Ces différentes étapes fût réaliser avec succès sur un des deux PC, mais lors de la même installation sur le second, on s'est rendu compte que les fichiers source de Linux n'existaient pas. Il a donc fallut réinstaller Linux .Cependant, le modules i82365 fût toujours introuvable, ce qui était dû à un problème de version entre les différents logiciels. Après plusieurs manipulations, nous avons décidé d'installer la même version de Mandrake 8.1 que celle existante sur le premier PC.

Il s'agit maintenant de faire fonctionner notre pile de protocole Bluez avec la carte xircom et tester des communications entre nos deux PCs. Pour cela , nous avons choisit au départ de travailler avec la version 1.0 de Bluez qui est intégrée dans le noyau.
Nous avons donc insérer les modules de Bluez (voir Annexe) et nous avons essayer d'attacher la carte xircom ; c'est à dire de l'initialiser. Pour que cette étape soit concluante il faut que la pile reconnaisse la carte et lui attribue une adresse Bluetooth (sur 48 bits). Cette adresse n'est autre que l'adresse MAC de notre carte bluetooth (Selon les spécifications de la norme). Mais, la pile n'arrivait pas à voir notre carte : **Can't init device no such file or directory**.
Après plusieurs manipulations de fichiers de configurations et en se basant sur les messages d'erreurs dans le fichier /var/log/messages, on en a conclue que la pile ne reconnaissait pas l'existence d'une carte sur l'interface série.

Notre dernier recours fût et comme pour tout projet open source de se référer aux forums Bluetooth et plus précisément celui de Bluez. Après une lecture approfondie des archives, il s'est avéré que notre carte figure parmi celles qui nécessite un driver particulier **bt_uart**. Or ce dernier ne fonctionne qu'avec la dernière version de Bluez 2.1 et donc pas avec celle du noyau. Il fallait donc télécharger cette nouvelle version de Bluez, enlever les anciens modules et recompiler le noyau pour qu'il tienne compte de ces changements et des éventuelles dépendances qui autrement pourraient engendrer des conflits de version.

Mais avant cela il fallait aussi télécharger de nouvelles librairies et utilitaires pour faire la mise a jour de certain fichiers(update).

Une fois, cette étape passée avec succès, on peut recharger les modules de Bluez 2.1. Avant d'essayer d'attacher la carte xircom, il faut a ajouter des lignes de commandes au niveau de certain fichier de configuration tel que : /etc/modules .conf , /etc/hcid.conf ... (Voir Annexe).

A ce stade, tout fonctionne parfaitement, et il y a reconnaissance et attribution d'une adresse Bluetooth à notre carte.

```
[root@machin root]# lsmod
Module                  Size  Used by
hci_uart                5456   0  (unused)
l2cap                  14752   0  (unused)
bluez                  29456   0  [hci_uart l2cap]
serial_cs               5552   0  (unused)
ds                      6896   2  [serial_cs]
i82365                 23728   2
pcmcia_core            43936   0  [serial_cs ds i82365]
af_packet              12560   1  (autoclean)
ip_vs                  62000   0  (autoclean)
usb-uhci               21232   0  (unused)
usbcore                50752   1  [usb-uhci]
pcnet32                12144   1  (autoclean)
rtc                     5600   0  (autoclean)
[root@machin root]# hciattach /dev/ttyS0 xircom 115200 flow
[root@machin root]# hciconfig hci0 up
[root@machin root]# hciconfig hci0
hci0:   Type: UART
        BD Address: 00:10:A4:C8:9E:1A ACL MTU: 128:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:70 acl:0 sco:0 events:7 errors:0
        TX bytes:34 acl:0 sco:0 commands:7 errors:0
```

Et les deux carte arrive a se voir :

```
[root@machin tools]# l2ping 00:10:A4:C8:9D:F4
Ping: 00:10:A4:C8:9D:F4 from 00:10:A4:C8:9E:1A (data size 20) ...
20 bytes from 00:10:A4:C8:9D:F4 id 200 time 42.58ms
20 bytes from 00:10:A4:C8:9D:F4 id 201 time 38.22ms
20 bytes from 00:10:A4:C8:9D:F4 id 202 time 46.94ms
20 bytes from 00:10:A4:C8:9D:F4 id 203 time 45.65ms
20 bytes from 00:10:A4:C8:9D:F4 id 204 time 44.35ms
5 sent, 5 received, 0% loss
```

Nous venons ainsi par ces tests de valider le bon fonctionnement de Bluez avec notre carte Bluetooth, des expériences d'évaluation de performances seront présentées plus loin.
Voulant aller plus loin en greffant plus de couches protocolaires dans le soft. La premiere idée à laquelle nous avons pensé est de réduire les fonctionnalités de notre carte xircom. D'après l'implémentation de Bluez , il est clair que la carte prend en charge les deux couches basses de la pile Bluetooth c'est a dire RF et Bande base . Ainsi, nous avons pensé à remplacer notre carte xircom par une autre dont les fonctionnalités se limiteraient à la couche radio et substituer la couche bande base par un programme en C permettant d'effectuer les opérations de codage /décodage. Mais vu que actuellement, il n'existe pas de telles cartes radio sur le marché, nous avons pensé à utiliser un analyseur de réseau disponible à l'INT.
En effet, ce dernier permet de démoduler n'importe quel signal en spécifiant le type de modulation à utiliser et la porteuse du signal. Or, il s'est avéré que notre analyseur ne peut effecteur la démodulation que sur une porteuse fixe or Bluetooth suit un mécanisme de sauts

de fréquences, et donc la porteuse change de fréquence avec un rythme de1600 sauts par seconde.

On a donc essayer de résonner autrement, vu qu'on émet sur la même fréquence pendant 625 µs et vu qu'on effectue 1600sauts par secondes, on a de forte chance de revenir a la même fréquence pendant un temps très limité pendant lequel l'analyseur n'aurait pas eu conscience du changement. Mais ce temps aussi court soit il est suffisant pour que l'analyseur perde la synchronisation.

Donc et par manque de moyen matériel et temporel nous avons fini par abandonner cette alternative et on s'est consacré sur l'implémentation de couches supérieurs tel que IP. L'impact de la différence entre les environnements mobiles et les environnements fixes se retrouve au niveau des applications : celles-ci ne peuvent pas être transposées telles quelles pour une exécution sur ou depuis un équipement sans fil. Une adaptation des applications est donc nécessaire. La plupart des approches actuelles consistent à modifier tout ou une partie des applications pour prendre en compte quelques aspects de la mobilité en supposant un environnement sans fil connu. Dans le cas de la technologie Bluetooth et comme déjà précisé dans la norme (chapitre **I**) nous avons besoin d'un protocole intermédiaire PPP. Mais aussi de l'utilisation du protocole RFCOMM qui permet d'émuler une connexion sur le port série. RFCOMM requière qu'une extrémité se comporte comme un hôte série "*serial host"* tandis que c'est à l'autre extrémité d'initier la connexion. Le démon *rfcommd* assure ces fonctionnalités puisqu'il écoute les requêtes RCOMM puis établit la connexion série entre les deux entités, il donne ensuite la main au démon *pppd* pour l'établissement de la connexion PPP.

Cette étape fût encore plus délicate que l'installation de la pile Bluez vu l'absence quasi totale de documentations qui se limitaient à quelques réponses à certain e-mails du forum de Bluez. Après plusieurs tentatives de configurations et de tests, d'ajouts de fichiers et de lignes de commandes (Voir Annexe); nous sommes parvenu à ouvrir une session RFCOMM entre un poste client et un poste serveur. Puis, grâce au démon pppd on a établissement d'une connexion PPP et d'une interface ppp0.

- Côte serveur RFCOMM:

```
[root@machin rfcommd-1.1]# ps -aux |grep rfcomm
root      2408  0.0  0.2  1476  560 ?        S    16:47   0:00 rfcommd[s]: waiting
root      2441  0.0  0.3  1940  764 pts/2    S    16:51   0:00 grep rfcomm

May 24 16:47:55 machin rfcommd[2408]: RFCOMMd server ver 1.1 05/22/2002
May 24 16:48:32 machin rfcommd[2409]: Connection from 00:10:A4:C8:9D:F4
May 24 16:48:32 machin rfcommd[2409]: Session na[00:10:A4:C8:9D:F4] opened
May 24 16:48:32 machin kernel: CSLIP: code copyright 1989 Regents of the University of California
May 24 16:48:32 machin kernel: PPP generic driver version 2.4.1
May 24 16:48:32 machin pppd[2411]: pppd 2.4.1 started by root, uid 0
May 24 16:48:33 machin pppd[2411]: Using interface ppp0
May 24 16:48:33 machin pppd[2411]: Connect: ppp0 <--> /dev/pts/6
```

```
[root@machin rfcommd-1.1]# ps -aux |grep ppp
root     2411  0.0  0.4  1936  880 pts/6      S    16:48   0:00 /usr/sbin/pppd /d
root     2415  0.0  0.4  1928  872 pts/6      S    16:48   0:00 /usr/sbin/pppd /d
root     2445  0.0  0.3  1940  764 pts/2      S    16:52   0:00 grep ppp
```

- Côte client :

```
May 24 18:51:43 dadou rfcommd[2646]: RFCOMMd client ver 1.1 05/22/2002 started
May 24 18:51:43 dadou rfcommd[2646]: Connecting to 00:10:A4:C8:9E:1A
May 24 18:51:44 dadou rfcommd[2646]: Session na[00:10:A4:C8:9E:1A] opened
May 24 18:51:44 dadou kernel: CSLIP: code copyright 1989 Regents of the University of California
May 24 18:51:45 dadou kernel: PPP generic driver version 2.4.1
May 24 18:51:45 dadou pppd[2650]: pppd 2.4.1 started by root, uid 0

[root@dadou rfcommd-1.1]# ps -aux |grep ppp
root     2650  0.0  1.3  1920  852 ttyS2    S    18:51   0:00 /usr/sbin/pppd /d
root      2654  0.0  1.2  1936  760 pts/2    S    18:54   0:00 grep ppp
```

3. Tests et Mesures :

- Analyse de la trame HCI :

Les modules de Bluez disposent donc d'une interface bas niveau (HCI) et nécessitent d'être commandé par un processeur extérieur (host) au module PC , qui échange avec celui-ci des commandes spécifiques au monde Bluetooth : Il existe quatre types de paquets HCI qui peuvent être envoyer sur la couche de transport UART :des paquets de commandes HCI, des paquets d'événements HCI, des paquets HCI ACL et finalement des paquets HCI SCO. Un champ id au niveau du paquet HCI permet de les différencier.

Type de paquet HCI	**Identificateur**
Paquet de commande	0x01
Paquet de donnée ACL	0x02
Paquet de donnée SCO	0x03
Paquet d'événement	0x04

Tableau III-1: Identificateurs des différents types de paquets

Les paquets de commandes HCI sont utilisés pour envoyés des commandes au contrôleur de Hôte (la carte Bluetooth) à partir du hôte (la pile de protocole) . Les commandes HCI ne prennent pas toutes le même temps pour s'effectuer. C'est pourquoi le résultat d'une commande sera retourné à l'hôte sous forme d'un HCI event.

L'ensemble de ces échanges peut être suivit grâce a un analyseur de trame HCI « Hcidump » qui enregistre dans un fichier journal une trace du traffic.

- **Inquiry :**

La procédure Inquiry permet de découvrir les appareils qui sont dans le rayon d'action d'un appareil Bluetooth. Une station procédant à un I*nquiry* envoie sur un canal radio une trame de courte durée invitant les stations à l'écoute à envoyer leurs informations personnelles.
Côté récepteur, la station procédant à l'*inquiry scan* passe dans l'état d'*inquiry response* et envoie ses informations personnelles à la station procédant à l'*inquiry*. Ces informations incluent l'adresse IEEE802, la classe de station (Les classes d'appareils ne sont pas encore normalisées) et le déphasage de l'horloge interne(la différence entre l'horloge de l'émetteur et celle du maître) .

```
[root@machin root]# hcitool inq
Inquiring ...
```

00:10:A4:C8:9D:F4 clock offset: 0x564e class: 0x000000

Le plus de bluez par rapport au logiciel Blueview de Windows est que sous Linux on peut voir séparément l'Inquiry suivi de l'étape de connexion, et donc mesuré le temps propre à l'exécution de chacune tâches.

		Temps d'un Inquiry
Distance 0.5m	**Mesure 1**	9.98s
	Mesure2	9.21s
	M esure3	10.19s
	Mesure4	9.8s
	Moyenne	9.795s
Distance 3.5m	**Mesure 1**	8.51s
	Mesure2	8.48s
	M esure3	9.52s
	Mesure4	10.26 s
	Moyenne	9.1925s
Distance 5m	**Mesure 1**	10.54s
	Mesure2	10.37s
	M esure3	11.16s
	Mesure4	10.33s
	Moyenne	10.6s

Tableau III-2: Mesure du temps d'un Inquiry

Ce temps est comme nous l'avons expliqué dans le chapitre précédent (**III.2**) indépendant de la distance, chose qui se confirme encore une fois ici sous Linux. Cependant, il est à remarqué que ces mesures montre un temps relativement petit par rapport à celui spécifier par la norme (10.26s) mais ceci est dû au fait que Bluez est configuré par défaut à fin d'arrêter la procédure d'Inquiry dés qu'un équipement répond à cette requête.

Pour avoir une idée plus clair sur cette procédure nous avons procédé à l'analyse de la trame au moyen de l'utilitaire Hcidump.

```
[root@machin root]# hcidump
HCIDump - HCI packet analyzer ver 1.2 05/22/2002.
device: hci0 snap_len: 1028 filter: 0xffffffff
< HCI Command: Inquiry(0x01|0x0001) plen 5
> HCI Event: Command Status(0x0f) plen 4
> HCI Event: Inquiry Result(0x02) plen 15
> HCI Event: Inquiry Complete(0x01) plen 2
```

Quand la commande Inquiry parviens au contrôleur d'hôte qui est dans notre cas la carte, ce dernier renvoie un Status event pour signaler à la pile que la procédure d'exécution de l'Inquiry viens de commencer. A chaque fois qu'une station distante répond à cette procédure, un message Inquiry Result est retourné vers l'hôte. A la fin de la procédure un Inquiry complet event est retourné vers la pile Bluetooth. Ce qui est à noter dans notre cas, est que comme précédemment signalé l'implémentation de la pile Bluez est configuré par défaut afin d'arrêter la procédure d'Inquiry dés qu'un premier équipement répond à cette requête : ce qui se voit au niveau de la commande « HCI Command: Inquiry(**0x01**|0x0001) ».

Value	Parameter Description
0x00	Unlimited number of responses.
0xXX	Maximum number of responses from the Inquiry before the Inquiry is halted. Range: 0x01 – 0xFF

Tableau III-3: Paramétres de la requête Inquiry

- **Page**

Les procédures de *page* et de *page scan* permettent l'établissement d'une connexion entre deux stations. Elles fonctionnent sur le même principe que les procédures d'I*nquiry*.
Pour accélérer la procédure de connexion, l'émetteur va exploiter les informations recueillies lors de l'I*nquiry* précédent. En particulier, il va tenter de prédire le canal sur lequel le récepteur sera à l'écoute lors de la transmission de la demande de connexion. Ainsi les trames de demande de connexion seront envoyées en premier lieu sur le canal radio prédit.
Lorsqu'une station va répondre avec succès à la procédure de *page* elle va quitter l'état de veille pour passé en état connecté. Elle sera alors intégrée, en tant qu'esclave, au sein du piconet dont le maître n'est autre que l'initiateur de la procédure de *page*.

Dans le cas de la pile de Bluez, l'établissement d'une connexion avec un équipement Bluetooth découvert se fait au moyen de l'utilitaire Hcitool. Supposons qu'on configure notre équipement en tant que maître :

```
[root@dadou root]# hcitool cc --role=m 00:10:A4:C8:9E:1A
[root@dadou root]# hcitool con
Connections:
        < ACL 00:10:A4:C8:9E:1A handle 39 state 1 lm SLAVE
```

On a donc ensuite procédé au mesure du temps nécessaire à l'établissement de notre connexion pour nous retrouver avec des résultats similaires à ceux de la norme :

		Temps d'un Page
Distance 0.5m	**Mesure 1**	1.48s
	Mesure2	2.08s
	M esure3	2.14s
	Mesure4	1.84s
	Moyenne	1.885s
Distance 3.5m	**Mesure 1**	2.49s
	Mesure2	2.6s
	M esure3	1.65s
	Mesure4	2.91s
	Moyenne	2.4125s
Distance 5m	**Mesure 1**	2.76s
	Mesure2	3.19s
	M esure3	2.51s
	Mesure4	2.97s
	Moyenne	2.8575s

Tableau III-4 : Mesure du temps d'un Page

Toujours afin d'avoir une idée sur l'échange entre la pile et la carte au niveau de la couche HCI, nous avons utilisé l'analyseur de trame HCI :Hcidump

- Au niveau de l'esclave:

```
[root@machin root]# hcidump
HCIDump - HCI packet analyzer ver 1.2 05/22/2002.
device: hci0 snap_len: 1028
< HCI Command: Create Connection(0x00|0x0005) plen 13
> HCI Event: Command Status(0x0f) plen 4
> HCI Event: Connect Complete(0x03) plen 11
```

En se référant à la norme on peut interpréter la signification des paramètres de chaque commande HCI, en particulier quand au premier champ de la commande « Create Connection (**0x01**|0x0005) », qui indique que notre station jouera le rôle d'un esclave tout le long de cette connexion. Mais aussi, qu'à tout moment cette station peut basculer de cette état pour devenir maître.

Value	Parameter Description
0x00	Become the Master for this connection. The LM will perform the Master/ Slave switch.
0x01	Remain the Slave for this connection. The LM will NOT perform the Master/Slave switch.

Tableau III-5 : Configuration de la station en tant que Maître/Esclave

❖ Au niveau du maître:

```
[root@dadou root]# hcidump
HCIDump - HCI packet analyzer ver 1.2 05/06/2002.
device: hci0 snap_len: 1028
> HCI Event: Connect Request(0x04) plen 10
< HCI Command: Accept Connection Request(0x01|0x0009) plen 7
> HCI Event: Command Status(0x0f) plen 4
> HCI Event: Connect Complete(0x03) plen 11
```

On a la même configuration du côté de la station distante, qui est dans notre cas le maître avec toujours cette possibilité de changer de rôle Maître/ Esclave. Ce paramétrage se fait au niveau de la commande « Accept Connection Request(**0x01**|0x0009) ».

Value	Parameter Description
0x00	The local device will be a master, and will not accept a master-slave switch requested by the remote device at the connection setup.
0x01	The local device may be a master, or may become a slave after accepting a master-slave switch requested by the remote device at the connection setup.
0x02-0xFF	Reserved for future use.

Tableau III-6 : Description du paramétre de la commande Accept Connection Request

- Influence d'une variation de MTU sur le débit mesuré:

Comparé à d'autres média physiques, les paquets de données du protocole Base bande sont limités en taille. L'utilisation du paramètres unité maximale de transmission « MTU » (Maximum Transmission Unit) au niveau d'une couche intermédiaire L2cap permet d'améliorer l'utilisation de la largeur de la bande par les protocoles de couches supérieurs qui sont conçus pour employer de plus grands paquets (MTU > taille d'un paquet basebande) ; mais aussi de réduire les opérations de segmentation et de ré-assemblage en limitant la taille des paquets de niveau supérieur à une valeur maximale .

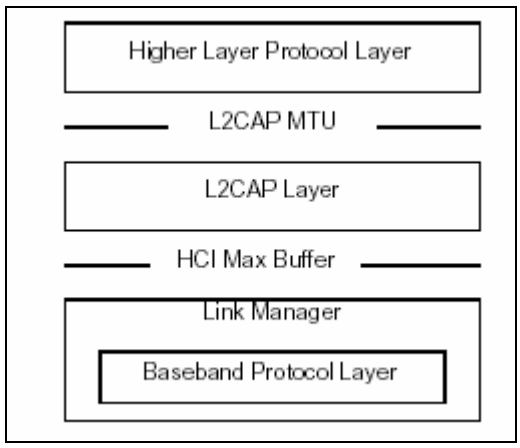


Figure III-6: Unité Maximale de Transmission

Afin de mieux comprendre l'influence de ce paramètre sur les performances du protocole, nous avons procédé à des mesures de débit en variant la valeur du Maximum Transmission Unit. Ceci fût possible grâce à l'utilitaire l2test qui permet de générer des données l2cap en agissant sur un grand nombre de leur caractéristiques y compris la valeur de MTU.

Les résultats de notre expérience ont été reporté sur la courbe ci dessous:

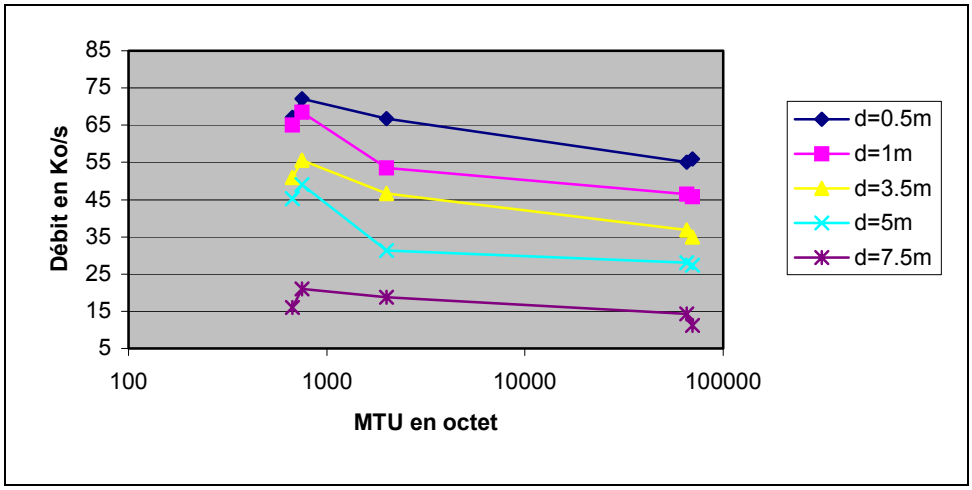


Figure III-7: Influence de la variation de MTU sur le débit mesuré

- **Interprétation :**

Nous remarquons que le débit augmente en augmentant la valeur de MTU pour atteindre une valeur maximale de 72 Ko/s (576 Kb/s) pour une valeur de MTU de l'ordre de 750 octet, au delà de laquelle il chute considérablement. Ce qui est tout à fait prévisible : Pour de petites valeurs de MTU, les opérations de segmentations et de ré assemblage sont réduite lors du passage de la couche l2cap vers la couche Base bande et inversement ,ce qui permet d'une part de gagner en temps de traitement et d'autre part en terme de charges utiles (entêtes de segmentations réduites) :

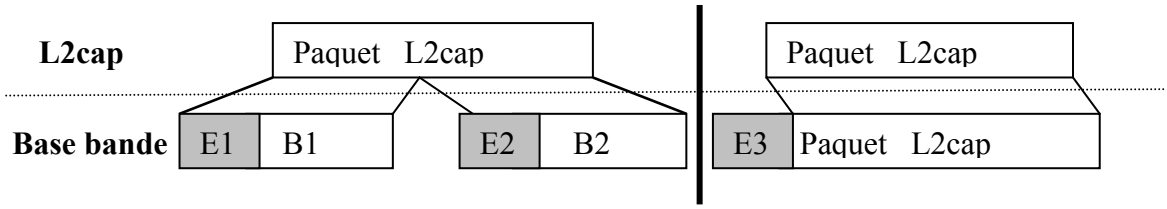


Figure III-8: Mécanisme de segmentation du message L2cap

Par contre, plus les valeurs de MTU augmentent et plus le débit chute vu que dans la plupart des implémentations, la couche HCI vérifie l'entête de chaque paquet reçu afin d'avoir : l2cap-hdr.data_length= Σ longueurs des données ACL de tous les fragments ; autrement il y a rejet de toute la trame l2cap. Ce qui signifie qu'ils doivent stocker au niveau d'un buffer tous les fragments. Or, si la taille du paquet l2cap est importante (en d'autre terme MTU important), la taille du buffer risque d'être insuffisante et donc on a perte de certains fragments et par la suite perte du paquet l2cap initial, d'ou retransmission et donc diminution du débit. Il est aussi intéressant à noter que, dans le cas de plusieurs connexions, l2cap ne

peux mixer des fragments qui appartiennent à des CID différents. Ce qui veux dire quand on envoie 10000 octet qu'on ne peut pas envoyer d'autres données relatifs à d'autre connections jusqu'à la fin de notre trame. D'ou la performance des autres canaux chute aussi.

Au cours de notre expérience nous avons aussi remarquer que la valeur de MTU ne peut dépasser un seuil minimal de 672 octet, mais aussi une valeur maximale fixée à 65535 octet.

- Valeur max:

[root@machin tools]# ./l2test **-I 10000000000000000000** -r
l2test[2353]: Waiting for connection on psm 10 ...
l2test[2344]: Connect from 00:10:A4:C8:9D:F4 [**imtu 65535**, omtu 672, flush_to 65535]
l2test[2352]: 672 bytes in 0.01 sec, 119.38 kB/s
l2test[2352]: 672 bytes in 0.01 sec, 116.73 kB/s
l2test[2352]: 672 bytes in 0.01 sec, 91.60 kB/s

- valeur min :

[root@dadou tools]# ./l2test **-O 650** -s 00:10:A4:C8:9E:1A
l2test[2558]: Connected [imtu 672, **omtu 650**, flush_to 65535]
l2test[2558]: Sending ...
l2test[2558]: Send failed. Invalid argument(22)

Or, si on se reporte à l'architecture d'un paquet l2cap en mode orienté connexion ou sans connexion ,on remarque que la longueur maximale du champ information (payload) est de 2^{16} soit donc 65536 octet ce qui donne que MTU =< 65536 .

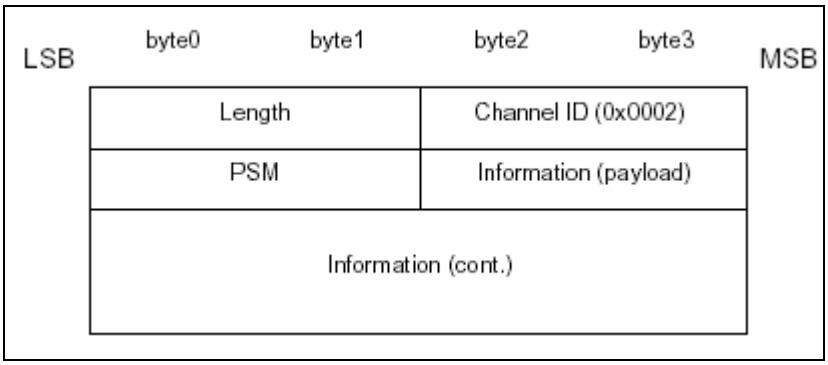


Figure III-9 : Paquet l2cap en mode sans connexion

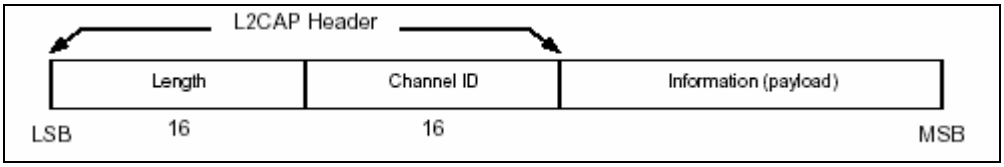


Figure III-10: Paquet l2cap en mode orienté connexion

La norme spécifie aussi une valeur minimal de MTU qui est fixé à 670 octet soit 5360 bit. Pour essayer de retrouver le pourquoi de cette valeur, on peut se reporter au format du paquet base bande dont le champ réservé aux données à transmettre à pour longueur maximale 2746 bit soit la moitié de la valeur minimale de MTU. Ce qui est tout à fait logique car en supposant que la valeur minimale de MTU soit égale à celle de la charge utile d'un paquet base bande, la couche l2cap perdrait tout son intérêt : les applications de haut niveau serait de nouveau obligé de ne laisser passer que des données de taille très limitée.

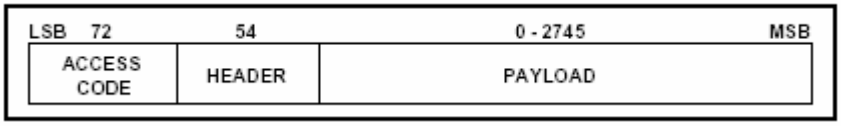


Figure III-11 : Format d'un paquet base bande

- Influence du types de paquets sur le débit mesuré :

Au sein d'un piconet, deux types de liens sont possibles. Les liens SCO (*Synchronous Connection-Oriented link*) et les liens ACL (*Asynchronous Connection-Less link*). Le lien ACL est un lien point-à-multipoint entre le maître et l'ensemble des esclaves. Il n'en existe qu'un par piconet. Les liens SCO sont utilisés pour le transport de la voix. Ils sont établis après négociation entre le maître et l'esclave concerné.
Le lien ACL est utilisé pour le transfert de données en tout genre. Il existe essentiellement deux types fondamentaux de paquets sur ce lien: les paquets DM (*Data Medium*), les paquets DH (*Data High*).
Au cours de cette expérience, nous avons procédé à deux série de tests, la première consiste à mesuré le débit d'un flux de donnée l2cap d'une station1 vers une station distante2 en variant la distance les séparant et le type de paquets émis, la seconde consiste en l'envoi simultanée dans les deux directions d'un flux de données l2cap et bien sûr en variant toujours le type de paquets transmis . Les résultats de nos mesures sont reportés sur les figures **III-12** et **III-14**.

- **Nature du transfert effectué:**

Il s'agit d'un transfert ACL puisque c'est un échange de paquets de données et non de voix , on peut vérifier ça grâce à l'utilitaire de Bluez Hciconfig:

- Station esclave:

```
[root@dadou tools]# hciconfig hci0
hci0:   Type: UART
        BD Address: 00:10:A4:C8:9D:F4 ACL MTU: 128:8  SCO MTU: 64:8     UP
RUNNING PSCAN ISCAN
        RX bytes:63013 acl:92 sco:0 events:6581 errors:0
        TX bytes:1067435 acl:15604 sco:0 commands:52 errors:0
```

- Station maître:

```
[root@machin tools]# hciconfig hci0
hci0:   Type: UART
        BD Address: 00:10:A4:C8:9E:1A ACL MTU: 128:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:1048382 acl:11487 sco:0 events:248 errors:0
```

TX bytes:9003 **acl:115 sco:0** commands:52 errors:0

- **Lien asymétrique :**

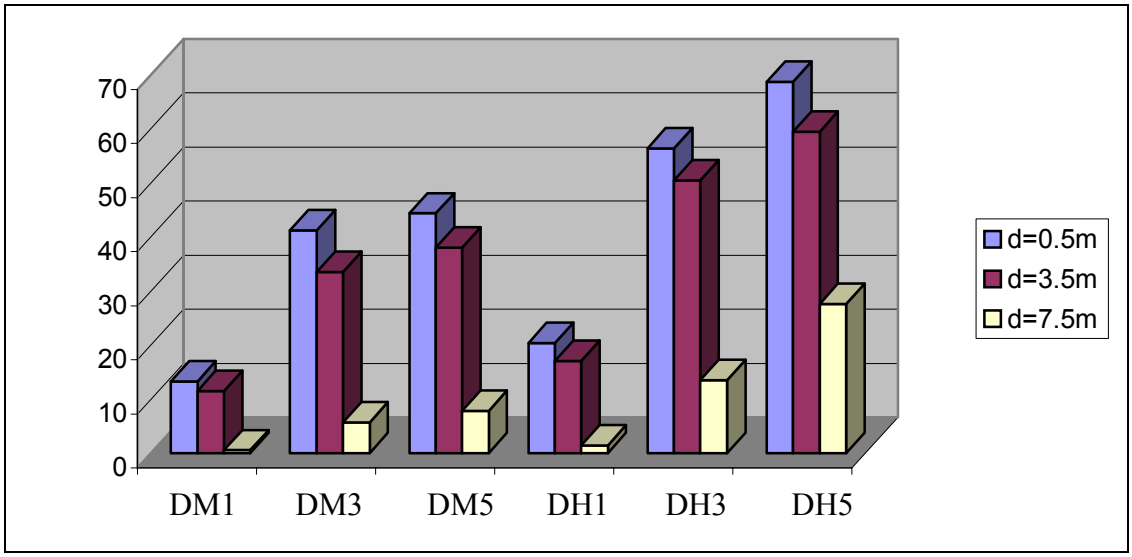


Figure III-12 : Influence du type de paquets sur le débit mesuré

- **Interprétation**

Les débits sur le lien sont rendus variables par l'utilisation de paquets plus ou moins longs. Ceux-ci peuvent occuper 1, 3 ou 5 slots ce qui permet lorsqu'une station a beaucoup de données à émettre d'améliorer le débit en réduisant la taille des entêtes. La figure **III-13** illustre bien ce résultat. On peut citer comme exemple le débit d'un paquet DM1 à une distance de 0.5m est de 11.22Ko/s soit 105.76 Kb/s alors que celui d'un paquet DM3 est de 33Ko/s soit presque le triple.

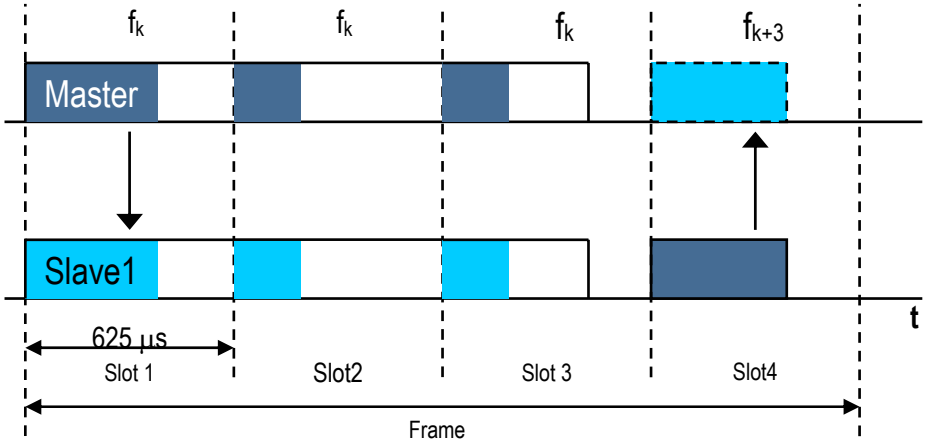


Figure III-13:Mécanisme d'émission d'une multi-trame (3slots)

De plus, les paquets DM incluent une redondance de type FEC (*Forward Error Correction*) pour permettre la correction d'éventuelles erreurs de transmission alors que les paquets DH (*Data High*) ne supportent pas de cette protection et peuvent donc, avec des tailles totales

équivalentes, transporter plus de données utilisateur. Ce qui implique que les débits assurés par ces derniers sont supérieur à ceux des paquets DM. Comme exemple on peut cité le débit mesuré par un paquet DH5 à une distance de 0.5m qui est de 66.10 Ko/s soit prés de 528 Kb/s tant dis celui mesuré à la même distance pour un paquet DM5 est de 44.31 Ko/s soit 354.48 Kb/s.
Cependant, cette courbe montre la vulnérabilité du débit face à la distance: indépendamment du type de paquets choisit le débit est sensiblement touché par le facteur éloignement.

- **Lien symétrique : D=3.5m**

les résultats reportés sur la courbe ci dessus sont ceux relatifs au sens Maître/Esclave :

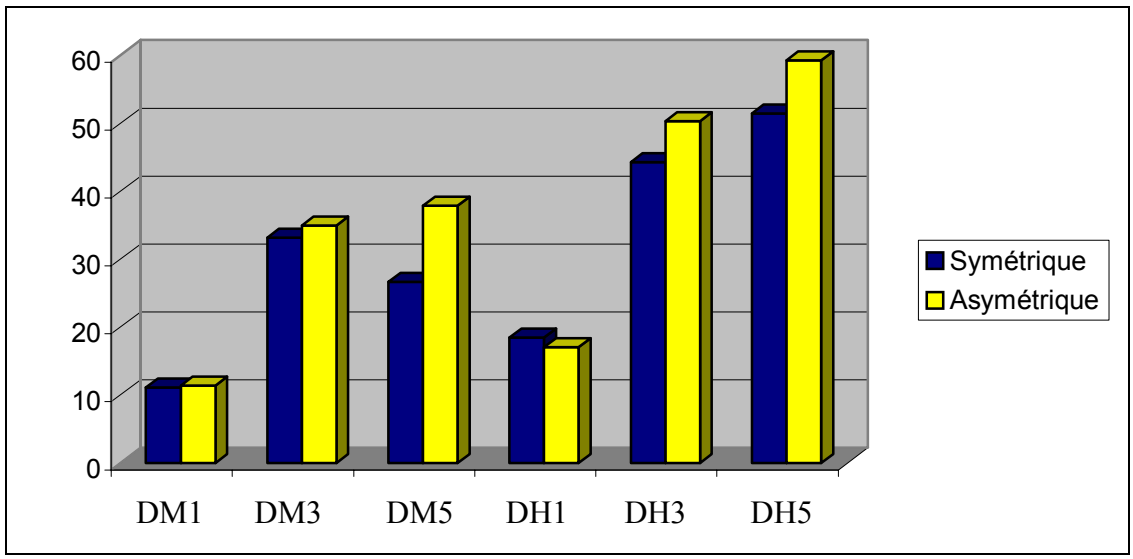


Figure III-14: Comparaison du débit mesuré sur un lien symétrique/asymétrique

- **Interprétation :**

Dans une étude comparative entre le débit mesuré dans le cas d'un lien symétrique et celui asymétrique, on remarque que le débit chute sur une liaison symétrique vu que les deux stations se partagent le support de façon équitable. Ces mesures sont très proche de ceux de la norme vu que par exemple pour un paquet DM1 on a des débits de l'ordre du 80 Kb/s , pour des paquets DM3 de l'ordre de 248 Kb/s .Plus précisément et en se reportant à la norme, on remarque qu'on a dans le cas d'un lien symétrique :

Débit mesuré=(débit asymétrique Maître/Esclave+débit asymétrique Esclave/Maître)/2

Type de paquet	Nombre de slot occupés	Taille des données utilisateurs (octets)	Débit symétrique max. (kbit/s)	Débit asymétrique max. (kbit/s)	
DM1	1	0-17	108,8	108,8	108,8
DH1		0-27	172,8	172,8	172,8
DM3	3	0-121	258,1	387,2	54,4
DH3		0-183	390,4	585,6	86,4
DM5	5	0-224	286,7	477,8	36,3
DH5		0-339	433,9	723,2	57,6

Tableau III-7: Caractéristiques des différents paquets ACL

- Mesure du débit en fonction de la taille des paquets:

On a fixé la valeur de MTU à 2000 octet et on a essayer de varier la taille des paquets l2cap à transmettre, afin d'évaluer l'influence de ce dernier sur les performances du protocole . La première expérience fût bien évidemment de vérifier que pour des paquets dont la taille était supérieur à la valeur limite MTU ne sont pas transmis. Le résultat fût concluant :

❖ Côte esclave:

```
[root@machin tools]# ./l2test  -I 2000   -r
l2test[2505]: Waiting for connection on psm 10 ...
l2test[2506]: Connect from 00:10:A4:C8:9D:F4 [imtu 2000, omtu 672, flush_to 65535]
l2test[2506]: Receiving ...
l2test[2506]: Disconnect
```

❖ Côte maître:

```
[root@dadou tools]# ./l2test -O  2000 -b 2200  -s 00:10:A4:C8:9E:1A
l2test[2692]: Connected [imtu 672, omtu 2000, flush_to 65535]
l2test[2692]: Sending ...
l2test[2692]: Send failed. Invalid argument(22)
```

La courbe suivante résume l'ensemble des mesures effectuées :

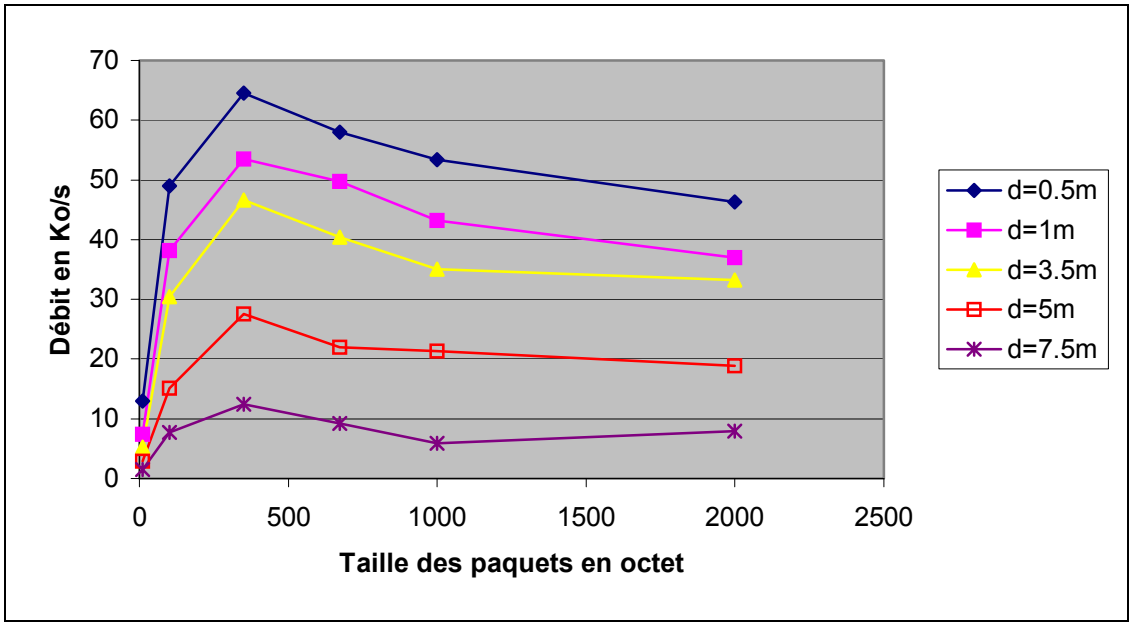


Figure III-15: Variation du débit en fonction de la taille des paquets l2cap

- **Interprétation**

Pour les courtes distances le débit croit avec l'augmentation de la taille du paquet parce que le sur-débit devient de plus en plus négligeable par rapport à la charge utile lors des transmissions des gros paquets. Pour des distances limites de 7.5m le débit décroît avec l'augmentation de la taille du paquet car les paquets de tailles importantes sont plus

susceptibles d'être corrompu , si la durée d'occupation du support est relativement importante, à cause des variations brusques de l'état du support tel que l'apparition du bruit de mouvement des objets. En plus, pour des distances plus importantes le signal est atténué et donc vu l'importance des taux d'erreur pour ces distances ceci provoque des retransmissions excessives. Le débit atteint une valeur maximale pour une taille des paquets L2cap de l'ordre de 300 octet soit 2400 bit qui correspond approximativement à la taille maximale du champ information au niveau du paquet Base bande 2756 bits, si on considère l'entête des couches intermédiaires. Donc c'est la taille limite pour la quelle on exploite toute la charge utile du paquet Base bande et on ne procède pas aux opérations de segmentations et ré-assemblages au passage entre la couche L2cap et la couche Base bande.

III. Conclusion

Installer une carte réseau, paramétrer ensuite le protocole s'avère un peu lourd sous Linux. Mais une fois cette étape effectuer, Bluez nous donne la main pour manipuler la pile de protocoles, accéder aux différentes couches (L2cap, HCI), analyser le traffic entre les entités protocolaires, effectuer des requêtes d'Inquiry et de Page; bref comprendre les spécificités de la norme. Choses qui n'était pas possible sous Windows.

Les débits mesurés sont important de l'ordre de 430 Ko/s, mais il ne faut pas oublier que ces mesures sont effectuées au niveau de la couche L2cap et si on considère les over-heads dû aux encapsulations des couches hautes tel que TCP et IP ce débit va chuter. Ainsi donc, les débits promis (par la norme) au niveau des couches applications sont mesurés au niveau de la couche liaison. Un autre paramètre important est la sensibilité du protocole sans fil face à la distance. Pour des raisons de logistiques (on n'a pas de portable avec Linux dessus) nous n'avons pas pu tester la distance limite au bout de laquelle il n'y a plus de détection de la station distante, mais les différentes courbes ci dessus montre que le débit chute considérablement en fonction de la distance.

L'inconvénient majeur de Bluez est qu'il n'implémente que les deux couches HCI et L2cap, récemment un package rfcommd-1.1 est devenu disponible sur Internet et peut être télécharger. Il permet d'implémenter les fonctionnalités de la couche RFCOMM, mais vu le manque de documentation cette tâche se révèle assez délicate.

L'implémentation de IP sur Bluetooth en utilisant la couche PPP n'est pas la solution optimale, plusieurs recherches visent actuellement d'essayer d'implémenter IP directement sur Bluetooth. Le groupe de travail RoHC (Robust Header Compression) de l'IETF a développé un mécanisme de compression d'en-têtes qui supporte les liens avec de grands taux d'erreurs et délais de transmission comme ceux rencontrés dans les réseaux cellulaires. Un tel mécanisme de compression d'en-têtes est indispensable pour assurer des performances raisonnables aux applications interactives à cause de la bande passante limitée et du coût des en-têtes d'IP sur les liens cellulaires. Le problème n'est pas seulement de réduire la taille des en-têtes pour l'efficacité, mais aussi de supporter les caractéristiques spécifiques aux réseaux cellulaires.

Conclusion

Parallèlement à l'usage d'équipements à forte mobilité (terminaux GPRS, UMTS), ou à mobilité plus réduite (terminaux IEEE 802.11, HiperLAN/2), la technologie Bluetooth cherche à affranchir les terminaux des problèmes de câblage. Le standard Bluetooth a déjà été adopté par un grand nombre d'entreprises et de nombreux produits utilisant cette technologie sont apparus. Le succès de Bluetooth est dû en partie à l'engouement dont bénéficient les technologies radio en ce moment et au fait que cette technologie permet, au contraire de 802.11 de connecter des appareils de nature variée (PC, PDA…).
Cependant, de nombreux problèmes pourraient ralentir le développement de Bluetooth : l'interopérabilité de produits Bluetooth créés par des marques différentes ne serait pas totale, même avec des équipements basés sur la même spécification technique.
Par ailleurs, la bande passante étant limitée à 1 Mbits/s, cela restreint les applications possibles de Bluetooth, notamment pour la vidéo en temps réel, mais les évolutions futures de la norme viendront sans doute combler ce manque. La sécurité a elle aussi été critiquée par certains experts, mais elle a été jugée suffisante par les membres du Bluetooth SIG.
La norme Bluetooth 2.0 est déjà en cours d'élaboration. Elle repoussera les distances de communication à une centaine de mètres et devrait encore étendre le nombre de périphériques communicants au sein des réseaux Bluetooth. Le débit autorisé devrait également passer à 10 Mb/s.
Est ce que Bluetooth réussira son pari et deviendra la norme de la connexion sans fil ? seul l'avenir nous le dira. L'année 2003 sera une année cruciale pour Bluetooth puisque l'on va pouvoir observer si les ventes d'équipements utilisant cette technologie décollent réellement, ou si elle est condamnée à disparaître.

Par ailleurs, cet accroissement du nombre de technologies de communication sans fil disponibles ou émergentes rend nécessaire la conception de mécanismes de coopération entre ces réseaux. Le protocole IP, transporté par l'ensemble des technologies, s'impose naturellement comme le protocole réseau capable d'assurer cette coopération. Cela permet d'unifier l'accès des applications aux services de transport de l'information à travers la couche IP quelle que soit la technologie sous-jacente. Cependant, la gestion optimisée et unifiée de plusieurs interfaces radio n'est pas prise en compte dans les terminaux.
Dans ce contexte il serait intéressant de penser de mettre en place des solutions technologiques permettant aux terminaux mobiles d'avoir un accès simultané ou successif à plusieurs technologies sans fil. Ces nouvelles fonctionnalités permettront d'offrir une gestion intelligente de la mobilité des équipements . Une interface générique interne au terminal permettra l'accès aux informations spécifiques à chaque technologie d'accès sans fil ce qui facilitera la gestion des handovers et le choix d'une interface radio. Lorsque plusieurs interfaces permettront simultanément une connexion au réseau, l'équipement choisira le support en fonction de différents critères comme la qualité de service et le coût d'accès au support.
Cette architecture pourra être utilisée par les opérateurs afin d'offrir une solution globale de mobilité qui permettra de s'affranchir de toute considération de couverture et de configuration répétée des équipements sans fil et d'optimiser l'usage de la ressource spectrale. D'autre part, elle leur permettra de définir de nouveaux services, qui prendront en compte la gestion des déplacements du client non seulement au sein d'un seul réseau mais aussi entre différents réseaux en optimisant le choix des points d'accès en fonction du service, de la qualité, de la sécurité...

Annexe

- **Installation de la carte Xircom:**

L'installation est pour Mandrake 8.1
La première étape à faire est de vérifier qu'on a « linux source code » installée dans notre système sinon on doit le télécharger et l'installer ou le cas échéant refaire l'installation de linux et inclure « linux source code ».
Dans le premier cas on doit être sûre que la version de « linux source code » est celle de notre noyau cela est possible par la commande **uname**.
Dans notre cas on a un Linux-2.4.18
On le télécharge dans /usr/src/

```
tar xzf Linux-2.4.18.tar.gz
make xconfig
```

xconfig est une interface qui permet d'accéder à tout les modules que linux peut supporter on doit activer ceux qui ont une relation avec pcmcia.
Une fois ceci fait, on lance les commandes suivantes :

```
make dep
make modules
make modules install
lilo
reboot
```

Ces commandes permettent de recompiler le noyau en insérant les modules nécessaires.
On télécharge ensuite les driver de l'adaptateur pcmcia sur port ISA et ceux de la carte xircom
La version de pcmcia qu'on a installé est pcmcia-cs-3.1.29

```
tar xzf pcmcia-cs-3.1.29.tar.gz
```

On les télécharge dans **/usr/src/**

```
cd /usr/src/pcmcia-cs-3.1.29
make clean
make config # On répend aux questions posées, les réponses par défaut doivent fonctionner
make all
make install
reboot
```

on édite le fichier **/etc/sysconfig/pcmcia**
il doit être comme suit :

```
PCMCIA=yes
PCIC=i82365
PCIC_OPTS=i365_base=0x3e2
CORE_OPTS=
CARD_MGR=-f
```

Par la suite on doit charger les modules nécessaires pour pouvoir utiliser l'adaptateur pcmcia.
Si tout a très bien fonctionner le module `pcmcia_core` doit être présent dans le noyau.

On doit ensuite charger les modules **i82365**, **ds** et **serial_cs**.

La commande utilisée est **insmod**

```
Insmod /lib/modules/2.4.18/pcmcia/i82365
Insmod /lib/modules/2.4.18/pcmcia/ds
Insmod /lib/modules/2.4.18/pcmcia/serial_cs
```

Reste maintenant à lancer le script qui permet la détection de l'interface pcmcia :

```
/etc/rc.d/init.d/pcmcia start
```

Pour s'assurer du bon déroulement de cette étape on peut vérifier l'état de notre interface pcmcia grâce aux commandes suivantes :

```
[root@machin root]# cardctl status
Socket 0:
  5V 16-bits PC Card
  Function 0: [ready]
Socket 1:
  no card

[root@machin root]# cardctl ident
Socket 0:
  product info: "Xircom", "CreditCard-Bluetooth-Adapter", "CBT", "0.04"
  manfid: 0x0105, 0x080a
  function: 2 (serial)
Socket 1:
  no product info available
```

- ## Installation de Bluez :

1. Pré requis

Afin de pouvoir utiliser BlueZ, il faut être en possession d'un kernel dont la version >= 2.4.4. Il est à noter que BlueZ fait aujourd'hui partie intégrante du kernel 2.4.6 .

2. .Compilation et installation

La premiére étape de l'installation est de décider quand à la version que nous allons utiliser. Comme nous l'avons déjà indiquer, pour des problèmes d' incompatibilité entre la carte xicom et bluez 1.0 (version du noyau), nous avons utiliser la version de Bluez 2.1 que nous avons télécharger d'Internet à l'adresse http://bluez.sourceforge.net/download.html
Avant toute étapes d'installations ou de configuration du Bluez 2.0 il faut télécharger des librairies et des utilitaires de mise à jour de certains fichier de Bluez .
Pour cela il faut suivre les étapes suivantes :

Télécharger le packages bluez-libs-2.0-pre9.tar.gz
Il faut ensuite le décompresser :

```
tar xvzf  bluez-libs-2.0-pre9.tar.gz
```

Puis compiler et installer ces librairies :

```
cd bluez-libs-2.0-pre9
./ Configure
make all
make install
```

Une fois cette étape passée avec succès, il faut télécharger la dernière version de Bluez.

Pour configurer BlueZ , il faut lancer la commande suivante :

```
./configure
```

Cette commande de configuration recherche automatiquement tous les composants et les packages nécessaires à l'installation. On peut ajouter les options suivantes:

`--enable-debug`	Permet le debuggage
`--with-kernel=<path>`	Chemin du code source du noyau par défaut c'est **`/usr/src/linux`**

Une fois la configuration s'est déroulée avec succès, on passe à l'étape de compilation et d'installation du BlueZ, on lance alors la commande:

```
make install
```

Il est aussi bon de lire les messages du fichier **`/var/log/messages`** pour voir s'il y a des messages d'erreurs.

Vu que nous avons déjà manipuler au paravent le Bluez du noyau, il était prudent de recompiler le noyau et de faire les dépendances entre les modules pré existants et ceux que nous venons d'installer et de compiler :

```
makedepend
make all
make modules
make modules install
Lilo
Reboot
```

3. Chargement des modules De BlueZ

Les lignes suivantes doivent être ajoutées dans le fichier **`/etc/modules.conf`** afin de permettre à BlueZ de fonctionner convenablement:

```
alias net-pf-31 hci
alias bt-proto-0 l2cap
alias tty-ldisc-15 hci_uart
alias char-major-10-250 hci_vhci
```

Après avoir effectué les changements ci-dessus, on peut alors lancer la commande "**depmod -a**" afin de permettre le chargement automatique des modules de BlueZ.

Le chargement manuel des modules peut être fait en lançant:

```
modprobe hci
modprobe hci_uart
modprobe hci_usb
modprobe l2cap
```

Pour vérifier que cette étape s'est bien déroulée on peut lancer la commande **`lsmod`** .On peut ainsi voir les modules de BlueZ.
Pour vérifier s'il y a eu des erreurs d'exécution, on regarde toujours le fichier **`/var/log/messages`**.

Une autre méthode pour vérifier le bon fonctionnement de Bluetooth est de lancer le script
```
/etc/rc.d/init.d/bluetooth start
starting Bluetooth :                                [OK]
```

4. Initialisation des dispositifs

i. Dispositifs UART :

Il faut premièrement vérifier que notre carte figure dans le fichier de configuration /etc/pcmcia/bluetooth.conf :

```
device "dtl1_cs"
  module "dtl1_cs"
device "bluecard_cs"
  module "bluecard_cs"
device "bt3c_cs"
  module "bt3c_cs"
card "Xircom CreditCard Bluetooth Adapter"
  version "Xircom", "*", "CBT"
  bind "serial_cs"
card "Xircom RealPort2 Bluetooth Adapter"
  version "Xircom", "*", "R2BT"
  bind "serial_cs"
card "Brain Boxes BL-620 Bluetooth Adapter"
  version "Brain Boxes", "Bluetooth PC Card"
  bind "serial_cs"
card "COM One Platinium Bluetooth PC Card"
  version "COM1 SA", "MC310 CARD"
  bind "serial_cs"
card "AmbiCom BT2000E Bluetooth PC/CF Card"
  version "AmbiCom,Inc", "BT2000E"
  bind "serial_cs"
card "Sphinx PICO Card"
  version "SPHINX", "BT-CARD"
  bind "serial_cs"
card "H-Soft blue+Card"
  version "H-Soft", "Blue+CARD"
  bind "serial_cs"
card "Compaq iPAQ Bluetooth Sleeve"
  version "CF CARD", "GENERIC"
  bind "serial_cs"
card "Nokia Bluetooth Card"
  version "Nokia Mobile Phones", "DTL-1"
  bind "dtl1_cs"
card "Socket Bluetooth Card"
  version "Socket", "CF+ Personal Network Card"
  bind "dtl1_cs"
card "LSE041 Bluetooth PC Card"
  version "BlueCard", "LSE041"
  bind "bluecard_cs"
card "LSE039 Bluetooth Compact Flash Card"
  version "WSS", "LSE039"
  bind "bluecard_cs"
card "3Com Bluetooth PC Card"
  version "3COM", "*", "Bluetooth PC Card"
  bind "bt3c_cs"
```

Ensuite, afin de pouvoir utiliser le driver bt_uart, il faut ajouter les lignes suivantes dans le fichier **/etc/pcmcia/config** :

```
device "btuart_cs"
  module "btuart_cs"

card "Xircom CreditCard Bluetooth Adapter"
  version "Xircom", "*", "CBT"
  bind "btuart_cs"
card "Xircom RealPort2 Bluetooth Adapter"
  version "Xircom", "*", "R2BT"
  bind "btuart_cs"
card "Brain Boxes BL-620 Bluetooth Adapter"
  version "Brain Boxes", "Bluetooth PC Card"
  bind "btuart_cs"
card "COM One Platinium Bluetooth PC Card"
  version "COM1 SA", "MC310 CARD"
  bind "btuart_cs"
card "AmbiCom BT2000E Bluetooth PC/CF Card"
  version "AmbiCom,Inc", "BT2000E"
  bind "btuart_cs"
card "Sphinx PICO Card"
  version "SPHINX", "BT-CARD"
  bind "btuart_cs"
card "H-Soft blue+Card"
  version "H-Soft", "Blue+CARD"
  bind "btuart_cs"
```

Il faut aussi ajouter le type de matériel que nous possédons ainsi que ses propriétés (tty, vitesse, flux, etc..)dans le fichier **/etc/hcid.conf** . On peut ensuite lancer le deamon **hcid,** ceci aura pour effet d'initialiser les dispositifs UART définit dans le fichier de configuration.

```
# HCI daemon configuration file.
# $Id: bluezhowto.tex,v 1.3 2001/08/24 20:04:39 maxk Exp $
# HCId options
options {
        # Automaticaly initialize new devices
        autoinit yes;
}
# Default settings for HCI devices
default {
        # Local device name
        name BlueZ;
        # Local device class
        class 0x100;
        # Default packet type
        pkt_type DH1,DM1;
}
# HCI devices with UART interface
uart {
        /dev/ttyS2 115200 flow xircom;
   }
```

Maintenant on peut procéder à l'installation du driver bt_uart, pour cela on le télécharge d'Internet, puis on lance les commandes suivantes:

```
tar xvzf btuart-0.2.tar.gz
cd btuart-0.2.tar.gz
./Configure
make install
```

Une fois correctement configurés, nos dispositifs doivent pouvoir être montés automatiquement. On peut aussi monter un dispositif manuellement en employant la commande **hciconfig :**

```
hciconfig hci0 up
```

5. Outils

i. hciconfig

C'est un utilitaire de configuration du dispositif HCI.

Hciconfig hciX [up	Montez le dispositif
|down	Démonter le dispositif
|reset	Dispositif Réinitialisé
|auth	Activer l'authentification
|noauth	Désactiver l'authentification
|piscan	Requête de Pagination et Activer le mode Inquiry
|iscan	Activer seulement le mode Inquiry
|pscan	Lancer une requête Page
|noscan	Arrêter le mode scan
|inq	envoie d'une requête Inquiry aux dispositifs
|features	
|conn	Montrez les connexions actives
|ptype	Type de paquet
|rstat]	Remise a zéro du stat conters

Exemple :
Pour connaître le type du paquet par défaut, lancer la commande:

```
Hciconfig hci0 ptype
```

ii. l2ping

L2CAP ping

```
l2ping [-s size] [-c count] [-f] <bd_addr>
```

iii. l2test

Utilitaire de test du protocole L2CAP

```
l2test <mode> [-b bytes] [-P psm] [-I imtu] [-O omtu] [bd_addr]
```

Modes:

- d	Dump(server) : débrancher le serveur
- c	Reconnect(client) : reconnecter le client
- m	Multiple connects(client) : connections multiples du client
- r	Receive (server) :recevoir (serveur)
- s	Send (client) :envoyer (client)

Options:

- I	Incoming MTU accepted
- O	Minimum outgoing MTU
- b	Taille des 'data chunks' en KB
- P	Employer ce PSM

Exemple :
Serveur: `l2test - I 2000 - r`
Client: l`2test - O 2000 - s < bd_addr >`

iv. Outils complémentaires

a) hcidump

Analyseur de paquet HCI
`hcidump < - hciX de I > [ - h ]`

b) hcitool

Permet le contrôle de l'interface HCI
`Hcitool [-i hciX] OGF OCF param..`

c) hciemud

Démon d'émulation HCI
`Hciemud [-n] local_address`

v. Exemples D'utilisations des outils

Mesure de debit :

- **Mesure du débit en fonction de la taille de paquets:**

1. taille des paquets = 10 :
côté client:
[root@dadou tools]# ./l2test -O 2000 -b 10 -s 00:10:A4:C8:9E:1A
l2test[2863]: Connected [imtu 672, omtu 2000, flush_to 65535]
l2test[2863]: Sending ...

côté serveur:
[root@machin tools]# ./l2test -I 2000 -b 10 -r
l2test[2408]: Waiting for connection on psm 10 ...

l2test[2407]: 10 bytes in 0.00 sec, 10.77 kB/s
l2test[2407]: 10 bytes in 0.00 sec, 11.16 kB/s
l2test[2407]: 10 bytes in 0.00 sec, 12.36 kB/s
l2test[2415]: 10 bytes in 0.00 sec, 14.01 kB/s
l2test[2415]: 10 bytes in 0.00 sec, 14.73 kB/s
l2test[2415]: 10 bytes in 0.00 sec, 14.45 kB/s
l2test[2415]: 10 bytes in 0.00 sec, 14.89 kB/s
l2test[2407]: 10 bytes in 0.00 sec, 8.80 kB/s
l2test[2407]: 10 bytes in 0.00 sec, 11.40 kB/s
l2test[2407]: 10 bytes in 0.00 sec, 11.42 kB/s
l2test[2407]: 10 bytes in 0.01 sec, 1.66 kB/s
l2test[2407]: 10 bytes in 0.00 sec, 8.46 kB/s
l2test[2407]: 10 bytes in 0.00 sec, 6.49 kB/s

2. taille des paquets = 100

côté client:
[root@dadou tools]# ./l2test -O 2000 -b 100 -s 00:10:A4:C8:9E:1A
l2test[2864]: Connected [imtu 672, omtu 2000, flush_to 65535]
l2test[2864]: Sending ...
côté serveur: on diminue a 100
l2test[2410]: 100 bytes in 0.00 sec, 43.36 kB/s
l2test[2410]: 100 bytes in 0.02 sec, 6.09 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 44.09 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 47.99 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 48.61 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 30.64 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 48.44 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 46.50 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 30.84 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 30.75 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 48.30 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 48.51 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 47.92 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 30.87 kB/s
l2test[2410]: 100 bytes in 0.00 sec, 22.75 kB/s

3. taille des paquets= 200

client:
[root@dadou tools]# ./l2test -O 2000 -b 200 -s 00:10:A4:C8:9E:1A
l2test[2867]: Connected [imtu 672, omtu 2000, flush_to 65535]
l2test[2867]: Sending ...

serveur:
[root@machin tools]# ./l2test -I 2000 -b 200 -r
l2test[2418]: Waiting for connection on psm 10 ...
l2test[2417]: 200 bytes in 0.00 sec, 43.31 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 51.26 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 60.33 kB/s
l2test[2417]: 200 bytes in 0.01 sec, 57.31 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 57.07 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 67.23 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 57.13 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 88.97 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 47.30 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 43.19 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 34.26 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 27.45 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 38.87 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 40.36 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 43.15 kB/s
l2test[2417]: 200 bytes in 0.00 sec, 46.43 kB/s
l2test[3136]: 200 bytes in 0.01 sec, 50.62 kB/s

4. Taille des paquets=672

client:
[root@dadou tools]# ./l2test -O 2000 -s 00:10:A4:C8:9E:1A
l2test[2865]: Connected [imtu 672, omtu 2000, flush_to 65535]
l2test[2865]: Sending ...

serveur:
[root@machin tools]# ./l2test -I 2000 -r
l2test[2413]: Waiting for connection on psm 10 ...
l2test[3136]: 672 bytes in 0.01 sec, 33.38 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.92 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.39 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.42 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.36 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.41 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.39 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 43.43 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 43.41 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.35 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.40 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.39 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.40 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.40 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.35 kB/s
l2test[3136]: 672 bytes in 0.01 sec, 53.39 kB/s

5. Taille des paquets=750
l2test[2475]: 750 bytes in 0.01 sec, 59.58 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.77 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.38 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.81 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 63.59 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 64.61 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.61 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 62.58 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 61.62 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.62 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.59 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.53 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.59 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 59.45 kB/s
l2test[2475]: 750 bytes in 0.01 sec, 64.57 kB/s

6. Taille des paquets=1000
l2test[2478]: 1000 bytes in 0.01 sec, 66.00 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.01 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.00 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.01 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 65.95 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.02 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.01 kB/s

l2test[2478]: 1000 bytes in 0.01 sec, 66.00 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.04 kB/s
l2test[2478]: 1000 bytes in 0.02 sec, 57.74 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 78.06 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.02 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.02 kB/s
l2test[2478]: 1000 bytes in 0.01 sec, 66.03 kB/s

7. Taille des paquets=2000

l2test[2504]: 2000 bytes in 0.03 sec, 70.85 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.47 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.71 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.65 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 68.42 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.60 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.73 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.46 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.56 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.60 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 65.57 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.40 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 72.05 kB/s
l2test[2504]: 2000 bytes in 0.03 sec, 71.59 kB/s

- **Mesure du débit en fonction du type de paquet :**

on fixe MTU = 672 et la taille des paquets= 672 (par défaut) et on modifie le type de paquets par la commande hciconfig hci0 ptype .
Liste des types de paquets disponibles par défauts :
[root@machin tools]# hciconfig hci0 ptype
hci0: Type: UART
BD Address: 00:10:A4:C8:9E:1A ACL MTU: 672 SCO MTU: 64:8

Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3

Changement du type de paquets :
[root@machin tools]# hciconfig hci0 ptype DM1
hci0: Type: UART
BD Address: 00:10:A4:C8:9E:1A ACL MTU: 672 SCO MTU: 64:8
Packet type: DM1

- **Type de paquets =DM1**

Débit :
l2test[2249]: 672 bytes in 0.05 sec, 13.19 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.18 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.19 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.18 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 12.86 kB/s

l2test[2249]: 672 bytes in 0.05 sec, 13.18 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.18 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.22 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 12.86 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.18 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 12.87 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.18 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.19 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.18 kB/s
l2test[2249]: 672 bytes in 0.05 sec, 13.19 kB/s

- **Type de paquets =DM3**

l2test[2322]: 672 bytes in 0.01 sec, 44.35 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.34 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.34 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.40 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.30 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.44 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.39 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.30 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.39 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.53 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.38 kB/s
l2test[2322]: 672 bytes in 0.01 sec, 44.34 kB/s

- **Type de paquets =DM5**

l2test[2294]: 672 bytes in 0.01 sec, 44.28 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.39 kB/s
l2test[2294]: 672 bytes in 0.03 sec, 22.03 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.33 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.31 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.36 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.32 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.55 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.37 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.35 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.34 kB/s
l2test[2294]: 672 bytes in 0.02 sec, 35.39 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.35 kB/s
l2test[2294]: 672 bytes in 0.01 sec, 44.32 kB/s

- **Type de paquets =DH1**

l2test[2304]: 672 bytes in 0.03 sec, 20.13 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.82 kB/s
l2test[2304]: 672 bytes in 0.04 sec, 16.50 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.39 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.49 kB/s

l2test[2304]: 672 bytes in 0.04 sec, 16.38 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.33 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.32 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.32 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.31 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.40 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 19.51 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.54 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 19.40 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.47 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.31 kB/s
l2test[2304]: 672 bytes in 0.03 sec, 20.12 kB/s

- **Type de paquets =DH3**

l2test[2288]: 672 bytes in 0.01 sec, 67.19 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 65.50 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 36.23 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 44.14 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 48.75 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 44.86 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 47.05 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 53.19 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 52.41 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 51.09 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 67.12 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 65.54 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 56.33 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 58.61 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 62.23 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 67.03 kB/s
l2test[2288]: 672 bytes in 0.01 sec, 57.01 kB/s

- **Type de paquets =DH5**

l2test[2291]: 672 bytes in 0.01 sec, 69.50 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 64.08 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 69.03 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 77.02 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 60.00 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 109.43 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 87.99 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 69.50 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 97.63 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 89.53 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 89.68 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 79.52 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 77.72 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 89.35 kB/s
l2test[2291]: 672 bytes in 0.01 sec, 77.67 kB/s

- **Mesure du débit en fonction de la taille de MTU:**

1. MTU =672

Côte client:
t@dadou tools]# ./l2test -O 672 -s 00:10:A4:C8:9E:1A
l2test[2549]: Connected [imtu 672, **omtu 672**, flush_to 65535]
l2test[2549]: Sending ...

Côte serveur:
l2test[2316]: 672 bytes in 0.01 sec, 66.98 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 66.91 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 67.03 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 66.95 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 62.24 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 58.59 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 66.94 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 66.92 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 67.05 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 67.03 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 66.84 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 64.32 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 65.84 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 71.25 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 66.86 kB/s
l2test[2316]: 672 bytes in 0.01 sec, 67.00 kB/s

2. MTU=10000

Côté client :
t@dadou tools]# ./l2test -O 10000 -s 00:10:A4:C8:9E:1A
l2test[2551]: Connected [imtu 672, **omtu 10000**, flush_to 65535]
l2test[2551]: Sending ...

Côte serveur:
[root@machin tools]# ./l2test **-I 10000** -r
l2test[2343]: Waiting for connection on psm 10 ...
l2test[2344]: Connect from 00:10:A4:C8:9D:F4 [**imtu 10000**, omtu 672, flush_to 65535]
l2test[2344]: Receiving ...
l2test[2320]: 672 bytes in 0.01 sec, 70.65 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 70.79 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 67.02 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 68.16 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 67.56 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 70.77 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 70.70 kB/s

l2test[2320]: 672 bytes in 0.01 sec, 67.04 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 70.75 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 70.72 kB/s
l2test[2320]: 672 bytes in 0.01 sec, 122.07 kB/s

6. Installation de RFCOMMd et PPP sur BlueZ

Tout d'abord il faut télécharger le packages rfcommd-1.1, le décompresser :

```
tar xvzf  rfcommd-1.1.tar.gz
```

Pour configurer et installer RFCOMMd il faut lancer les commandes suivantes:

```
 cd rfcommd-1.1
./configure
make install
```

Pour lancer l'exécution du serveur RFCOMM:

Serveur: **rfcommdd <-s> [-f file] [-P port]**

Pour lancer l'exécution du client RFCOMM:

Client : **rfcommdd [-f file] [-P port] [-L local address] [-p] [-t timeout] <host> <server adress>**

En employant l'option **- n** RFCOMMd ne se détachera pas du terminal ça permet de voir les messages d'erreurs s'il y en a.

❖ Etablissement d'une Liaison PPP

Cette étape est très importante, il faut essentiellement savoir ajouter les commandes appropriées au niveau du fichier rfcommd.conf du client et du serveur. Après plusieurs tentatives et rectifications voici la version finale de notre fichier de configuration.

- **Côté de serveur :**

```
options {
   psm 3;         # Listen on this psm.

   ppp            /usr/sbin/pppd;
   ifconfig       /sbin/ifconfig;
   route          /sbin/route;
   firewall       /sbin/ipchains;
}

# Network Access
na {
   channel 8;
   up {
         ppp "noauth 10.0.0.1:10.0.0.2";
   };
}
```

On lance le serveur en utilisant l'adresse IP 10.0.0.1 , et ce grâce à la commande suivante dans laquelle il faut préciser l'emplacement du fichier rfcommd.conf car par défaut ce dernier

est supposé se trouver sous /etc/rfcommd.conf mais vu que nous avons télécharger le package rfcommd-1.1 sous root notre chemin est différent:

```
Rfcommd -s –f /root/rfcommd-1.1/rfcommd.conf na
```

- **Côté de client :**

```
options {
   psm 3;         # Listen on this psm.

   ppp            /usr/sbin/pppd;
   ifconfig       /sbin/ifconfig;
   route          /sbin/route;
   firewall       /sbin/ipchains;
}

# Network Access
na {
   channel 8;
   up {
         ppp "noauth";
   };
}
```

On lance le client en utilisant l'adresse IP 10.0.0.2 , et ce grâce à la commande suivante:

```
rfcommd –f /home/admin/rfcommd-1.1/rfcommd.conf na 00:10:A4:C8:9E:1A
```

Ceci nous permet d'avoir une liaison PPP entre 10.0.0.1 (le serveur PPP) et 10.0.0.2 (le client PPP).

```
[root@dadou root]#ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:E2:03:C5:FA
          inet       addr:157.159.50.109              Bcast:157.159.50.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:240 errors:0 dropped:0 overruns:0 frame:0
          TX packets:30 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0xfca0

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.0.0.1  P-t-P:10.0.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
```