



Telecom Sud Paris
Année scolaire 2007/2008
2 juin 2008

Projet informatique 1^{ère} année
CSC 3502

Borne d'annonce Bluetooth



Réalisé par :

RIFAAT BAYER Farouk
MESBAHI Alaaedine
SAFFIR Hamzza
EL BATTI MAHJOUR

Encadré par :

AFIFI HOUSSAM

Sommaire

<u>Introduction</u>	page 1
I/ <u>Cahier des charges</u>	page 2
II/ <u>Etude théorique</u>	page 4
A/ Présentation générale de la technologie Bluetooth	page 4
B/ Présentation de WRAP Access Server	page 9
C/ Architecture client/serveur	page 10
D/ La notion de processus	page 13
E/ Exclusion mutuelle	page 14
F/ Les sockets	page 16
G/ LDAP	page 18
III/ <u>Le développement</u>	
A/ Le cas d'utilisation	page 19
B/ La conception préliminaire	page 20
C/ La conception détaillée.....	page 22
IV/ <u>Les tests</u>	
A/ L'environnement de travail	page 27
B/ La phase des tests	page 27
C/ Les difficultés rencontrées	page 29
Conclusion	page 31
Le code source	page 32
Le planning prévisionnel	page 33
ANNEXE	

Introduction

Notre projet intitulé BAB (Borne d'Annonces Bluetooth) est une reprise du projet AIPS (Accueil Interactif et Personnalisé en Station). Ce dernier, lancé par la société de transport parisien RATP au mois de Janvier 2006 avec la coopération de la société Mobiluck, a été réalisé par Mohamed Chaari l'année dernière dans le cadre de son projet de fin d'études.

Le projet BAB, comme son nom l'indique, consistera à établir et mettre en œuvre une plateforme Bluetooth dans le forum de l'INT et qui permettra de délivrer des contenus aux permanents et aux étudiants de l'INT.

Pour ce faire, une base de donnée devra être créée pour entrer les identités des différentes personnes veillant s'inscrire à ce nouveau type de service. Ainsi, la plateforme doit être capable de détecter les téléphones portables environnant, de leur attribuer leur exacte identité et puis finalement de leur transmettre les informations nécessaires (emploi de temps par exemple, les événements sociaux du jour...)

CHAPITRE 1 : cahier des charges

Le but de ce projet est de développer une application permettant de mettre en oeuvre les diverses fonctionnalités suivantes :

- ✓ La détection des appareils Bluetooth environnant.
- ✓ Accéder à une base de données pour vérifier si l'appareil détecté est souscrit au service ou non.
- ✓ Si l'utilisateur est souscrit, une requête LDAP est envoyée à l'interface Gaspar afin de récupérer des informations supplémentaires sur l'abonné.

Une fois qu'on a effectué ces opérations, une partie optionnelle peut être considérée. Il s'agit de transmettre à l'abonné des informations personnalisées, telles que l'emploi du temps, des événements sociaux, etc.

La plateforme est composée d'un serveur et d'une borne Bluetooth (point d'accès) et est installée dans le forum de Telecom Sud Paris. A travers de la figure 1, nous présentons l'architecture du réseau reliant notre plateforme au service S2IA.

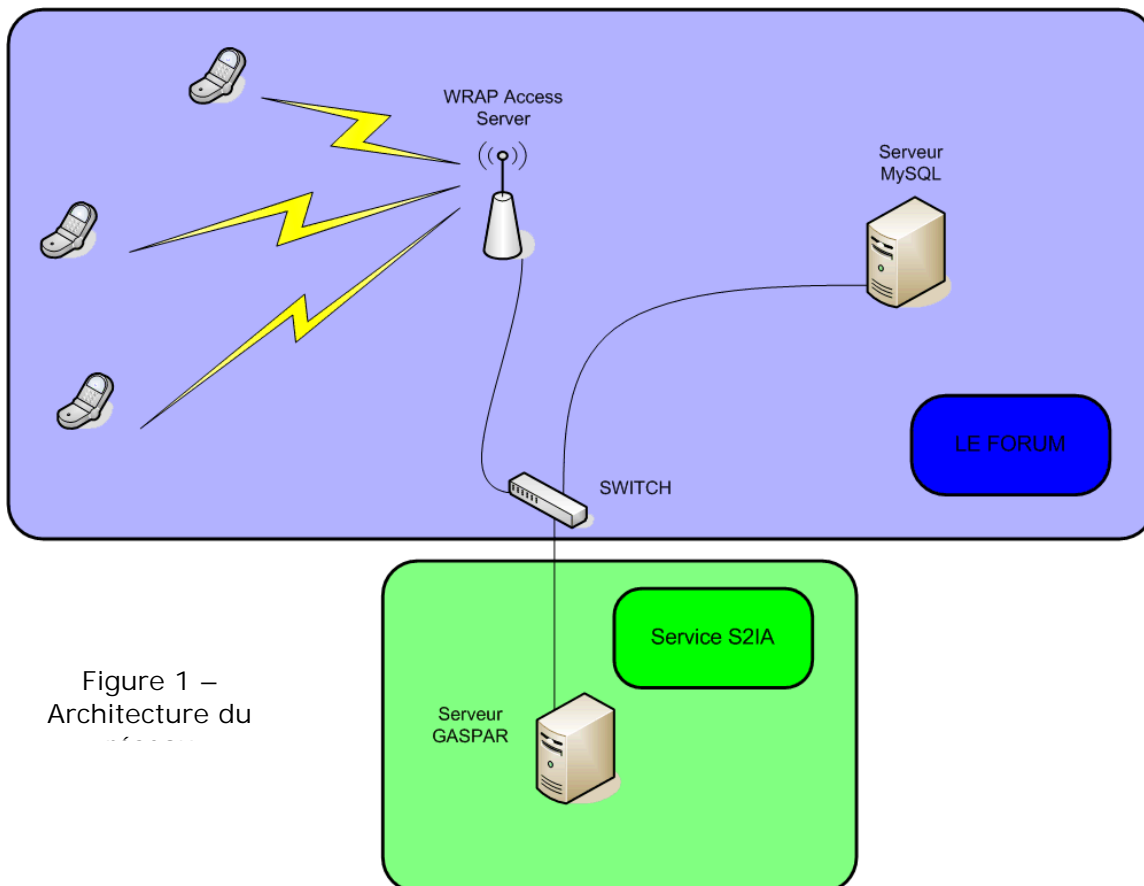


Figure 1 – Architecture du

Ceci dit nous avons considéré un ensemble de règles que notre application devrait respecter, à savoir :

- ✓ L'application doit respecter le principe de l'architecture trois tiers
- ✓ Lancement autonome de l'application
- ✓ Redémarrage périodique de l'application
- ✓ Amélioration des performances de l'application développée

Notons que le dernier point concerne surtout l'implémentation d'un principe de synchronisation entre processus, sachant que ces derniers peuvent accéder simultanément à une ressource critique en lecture ou en écriture, faussant ainsi les résultats.

Or, comme notre application sera implémentée sur une petite échelle, c'est-à-dire que le nombre d'appareils captés reste faible, nous n'allons pas prendre en considération la synchronisation des processus.

CHAPITRE 2 : étude théorique

A/ Présentation générale de la technologie Bluetooth

Un peu d'histoire...

Le nom Bluetooth (« dent bleue ») remonte au X^{ème} siècle, à l'époque où le roi danois Harald Blatand (910-986) a unifié son pays avec la Norvège. Le surnom « Blatand » qui signifie « à la dent bleue » lui a été donné car il aimait manger des myrtilles, et comme les brosses à dents n'existaient pas à cette époque, ses dents avaient la couleur bleu-violet des myrtilles. C'est alors à partir de cette histoire que les industries de l'informatique et de télécommunications ont eu l'idée de réunir leurs périphériques avec une nouvelle technologie nommée Bluetooth.

En 1994, Ericsson Mobile Communication lança une étude de la faisabilité d'une interface radio à faible puissance et à faible coût reliant les téléphones portables et leurs accessoires. Quelques années plus tard, Ericsson, Nokia, IBM, Toshiba et Intel ont formé le SIG (Special Interest Group). Ce groupe est formé par des leaders dans le domaine de la téléphonie mobile, des leaders dans le domaine de l'informatique et des leaders dans le domaine du traitement du signal digital. En 1999, la version 1.0 de la spécification Bluetooth a vu le jour. Aujourd'hui nous sommes à la 2.0. A la même époque, Microsoft, Lucent, 3COM et Motorola ont rejoint le SIG.

C'est le 17 Mai 2000, le consortium a décidé de baser le logo de la technologie Bluetooth sur l'histoire du roi Harald. En effet, ce logo est composé des lettres runiques « H » et « B » pour Harald Bluetooth.



L'objectif de cette technologie de communication sans fil tient en quatre points :

- ✓ Consommation d'énergie réduite
- ✓ Possibilité d'implantation dans des équipements de petite taille
- ✓ Capacité de véhiculer des données à haut débit
- ✓ Coût réduit

Principe de fonctionnement

Bluetooth est une technologie sans fil, les périphériques établissent leurs communications par ondes radio sur la fréquence des 2400 – 2483,5 MHz. Le débit de base est de 1 Mbits/s.

Le bloc de système Bluetooth

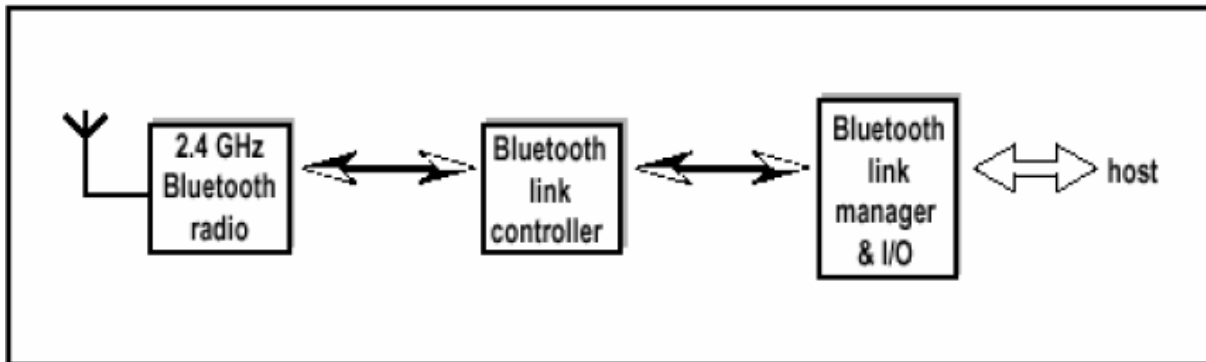


Figure 2 – Bloc de système Bluetooth

Le bloc de système Bluetooth se compose d'une unité radio, d'une unité de commande de liens, et d'une unité de support pour la gestion de liens et les fonctions terminales d'interface de centre serveur (voir figure ci-dessus). L'interface de contrôleur de centre serveur (Host Controller Interface ou HCI) fournit les moyens pour un appareil de centre serveur d'accéder aux matériels Bluetooth.

Les connexions

Pour éviter les collisions, l'envoi des informations s'effectue par paquets, comme lors des communications réseau par IP. Ces paquets sont encadrés de blocs de données de contrôle identifiant notamment l'appareil auquel sont destinées les informations. Ces données de contrôle permettent de plus la mise en réseau des appareils équipés de la puce. On parle du principe maître/esclave(s). Le nombre d'appareils chaînés, limité à 8, forme un picoréseaux ou piconets,

où tous les esclaves utilisent la même séquence de saut de fréquences et sont synchronisés sur l'horloge du maître. Un même appareil peut participer à plusieurs picoréseaux.

L'imbrication de ces picoréseaux forme alors un scatternet qui peut comporter jusqu'à 72 appareils sans perte de débit. 7 appareils esclaves peuvent simultanément dialoguer avec l'ordinateur maître chargé d'orchestrer et d'aiguiller les communications. Le maître est également chargé de superviser la discussion entre deux appareils esclaves.

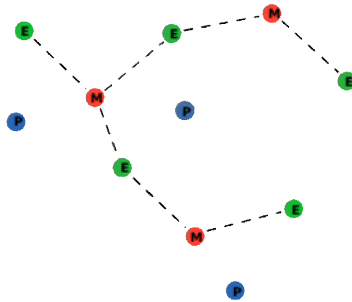


Figure 3 – Réseau scatternet

Les sauts de fréquences

La technologie Bluetooth emploie la méthode du saut de fréquence, qui signifie que chaque paquet est transmis sur une fréquence différente. Dans la plupart des pays, 79 canaux peuvent être utilisés. Avec un taux rapide de sauts (1600 par seconde), une bonne protection d'interférence est réalisée. Un autre avantage est une longueur courte de paquet. Si un autre appareil bloque la transmission d'un paquet, le paquet est renvoyé dans une autre fréquence déterminée par l'arrangement de fréquence du maître.

La fréquence de communication entre deux machines peut changer plusieurs centaines de fois par seconde. En pratique, chaque paquet transmis comporte la fréquence sur laquelle sera transmis le paquet suivant, ce qui fait baisser d'autant le débit potentiel de données le portant à 864 Kbits/s. Comme le débit est bidirectionnel, si deux machines sont connectées, la première peut envoyer des données à la

seconde, et la seconde peut envoyer d'autres données à la première, à un débit pratique maximum de 432 Kbits/s.

La pile protocolaire

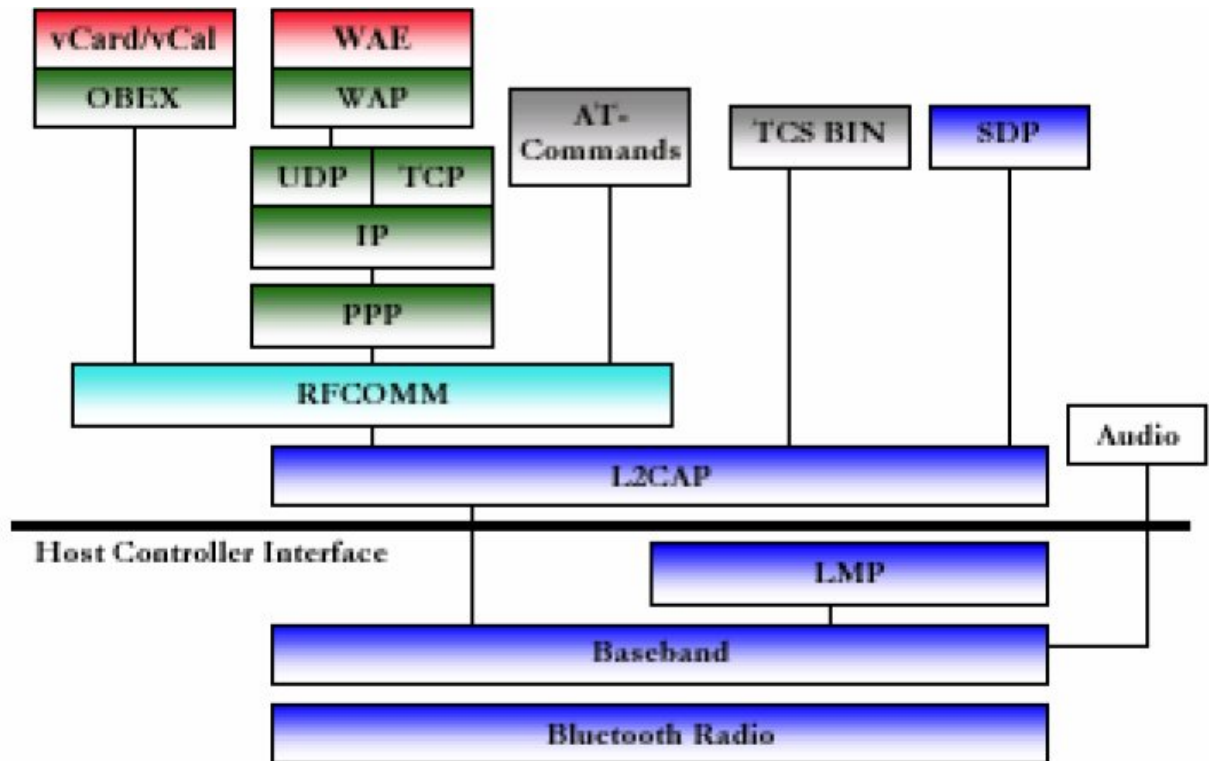


Figure 4 – La pile protocolaire

Protocoles du noyau Bluetooth

La « Baseband » et la commande de lien permettent un lien physique RF entre les unités Bluetooth formant ainsi un picorésau. Cette couche synchronise la fréquence des sauts de transmission et les horloges des différents appareils Bluetooth.

L'acoustique est conduite directement de la « Baseband » et vers la « Baseband ». Deux appareils quelconques Bluetooth soutenant l'acoustique peuvent envoyer et recevoir des données audio entre eux en ouvrant juste un lien audio.

Le protocole de directeur de lien (LMP) est responsable du lien installé (authentification et codage, commande, et négociation des paquets de

« Baseband ») entre les appareils Bluetooth et pour les modes de puissance et les états de connexion d'une unité Bluetooth.

Le contrôle de lien logique et le protocole d'adaptation (L2CAP) s'occupent du multiplexage, du réassemblage, et de la segmentation des paquets.

Le protocole de découverte de service (SDP) est nécessaire lors de la demande d'information sur l'appareil, de services, et des caractéristiques d'autres appareils. Les appareils doivent supporter le même service afin d'établir une connexion avec l'un et l'autre.

Protocoles de remplacement de câbles

RFCOMM émule les signaux RS-232 et peut-être ainsi employé dans les applications qui ont été autrefois mises en application avec un câble série (par exemple, un raccordement entre un ordinateur portable et un téléphone portable).

Protocoles de téléphonie

Protocole de contrôle de téléphonie binaire (TCS-BIN) définit la signalisation d'un contrôle d'appel pour l'établissement de la parole et des données appelée entre les appareils Bluetooth.

Protocoles adoptés

OBEX (échange d'objet) est adopté d'IrDA. C'est un protocole de session qui fournit des moyens pour le transfert simple et spontané d'objet et de données. Il est indépendant du mécanisme de transport et de l'interface de programmation d'application de transport (API).

TCP/UDP/IP est défini pour fonctionner dans les unités Bluetooth leur permettant de communiquer avec d'autres unités reliées, par exemple, à Internet. La configuration du protocole TCP/IP/PPP est employée pour tous les scénarios d'utilisation de pont d'Internet dans Bluetooth 1.0 et pour OBEX dans les futures versions. La configuration UDP/IP/PPP est disponible en tant que transport pour WAP.

PPP, dans la technologie Bluetooth, est conçue pour accomplir les connexions point par point. PPP est un protocole « packet-oriented »

et doit donc employer ses mécanismes de série pour convertir le flux de données de paquets en flux de données série.

B/ Présentation de Wrap Access Server

Comme nous l'avons mentionné auparavant, nous reprenons un projet qui a été développé l'année dernière dans le cadre d'une thèse professionnelle.

Pour mettre en place la plateforme Bluetooth, le choix des bornes (point d'accès) supportant la technologie Bluetooth est grand. Mais pour le projet de l'année dernière, le choix était basé sur le nombre de connexion aux équipements Bluetooth pour permettre la gestion de plus d'un client à la fois. Cette année aussi, nous développerons notre plateforme en utilisant la même borne.

Dans ce qui suit, nous présentons les caractéristiques de la borne choisie : WRAP Access Server.

Les caractéristiques principales sont :

- ✓ Possibilité de créer 21 connexions Bluetooth simultanément répartis sur 3 antennes
- ✓ Plateforme Linux ouverte aux nouvelles applications locales
- ✓ Joue le rôle d'un routeur ou d'un pont
- ✓ Utilisation possible de toutes les interfaces importantes : Bluetooth, Ethernet, WiFi, GSM et GPRS à carte CF, USB et RS 232
- ✓ Firewall par filtrage de paquets
- ✓ Installation facile et simple



Figure 5 – Le routeur WRAP Access Server

Pour les spécifications techniques, matérielles et logicielles du WRAP Access Server, veuillez vous référer à l'annexe.

C/ L'architecture client/serveur

Dans cette partie, nous présenterons les méthodes et les vocabulaires couramment utilisés dans le cadre des architectures client/serveur et qui nous ont été utiles pour comprendre comment devront être traitées les requêtes entre la borne et le serveur sur lequel la grande partie du traitement sera exécutée.

Présentation

L'architecture client/serveur désigne un mode de communication où des machines clientes (faisant partie du réseau) contactent un serveur - une machine généralement très puissante en termes de capacités d'entrées-sorties - qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion, etc.

Les acteurs principaux d'une architecture client/serveur sont :

- ✓ Le client : processus demandant l'exécution d'une opération à un autre processus serveur. La demande se fait par l'envoi d'un message contenant le descriptif de l'opération à exécuter et attendant la réponse à cette opération par un message en retour.
- ✓ Le serveur : processus accomplissant une opération sur demande d'un client et transmettant la réponse à ce client.
- ✓ Le middleware : ensemble des services logiciels construits au-dessus d'un protocole de transport afin de permettre l'échange de requêtes et des réponses associées entre client et serveur de manière transparente.

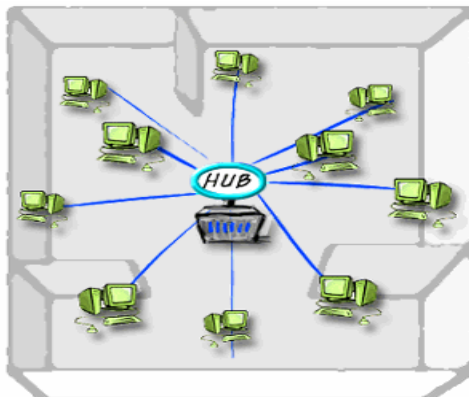


Figure 5 – Exemple d'architecture client/serveur

Avantages du modèle client/serveur

Le modèle client/serveur est particulièrement recommandé pour les réseaux nécessitant un haut niveau de fiabilité; ses principaux atouts sont:

- ✓ Des ressources centralisées. Etant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.
- ✓ Une meilleure sécurité. Le nombre de points d'entrée permettant l'accès aux données est moins important.
- ✓ Une administration au niveau serveur. Les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.
- ✓ Un réseau évolutif. Grâce à cette architecture on peut supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modifications majeures.

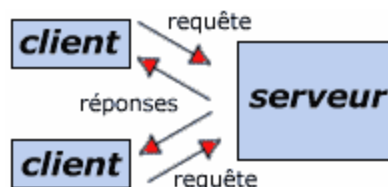
Inconvénients du modèle client/serveur

Malgré les avantages que nous venons de citer, l'architecture client/serveur présente tout de même quelques lacunes.

- ✓ Un coût élevé dû à la technicité du serveur. Un serveur doit être puissant et rapide, afin de pouvoir gérer le plus rapidement possible les requêtes d'un grand nombre de processus.
- ✓ Un maillon faible. Le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui.

Fonctionnement du modèle client/serveur

Un réseau client/serveur fonctionne selon le schéma suivant:



- ✓ Le client émet une requête vers le serveur grâce à son adresse et un port, qui désigne un service particulier du serveur.
- ✓ le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client et son port.

Dialogues synchrones et dialogues asynchrones

Dans une communication synchrone, le dialogue se fait sans file d'attente, où les commandes d'émission et de réception sont bloquantes.

Inversement, le dialogue asynchrone est géré par une file d'attente, où l'une au moins des commandes d'émission ou de réception est non bloquante.

Accès itératif et accès concurrent

Les termes itératif et concurrent sont utilisés pour définir la gestion des requêtes.

Dans le type itératif, un seul processus est créé pour gérer toutes les requêtes de tous les clients, alors que dans le type concurrent, plusieurs processus sont créés dont chacun gère une requête pour un client.

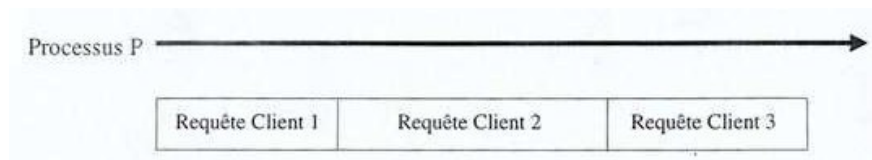


Figure 6 – Exemple d'un accès itératif

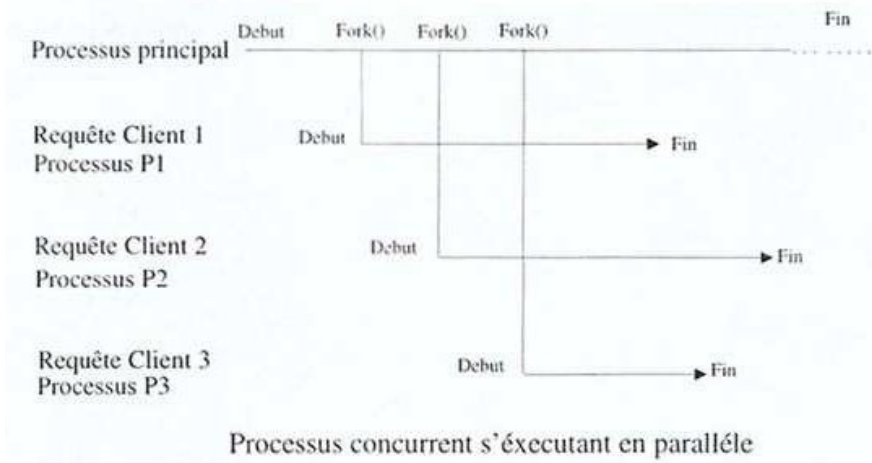


Figure 7 – Exemple d'un accès concurrent

D/ La notion de processus

Définition

Un processus ou process (en anglais), en informatique, est défini par :

- ✓ Un ensemble d'instructions à exécuter
- ✓ Un espace mémoire pour les données de travail
- ✓ D'autres ressources, comme des descripteurs de fichiers, des ports réseau, etc.

Les états d'un processus

A un instant donné, un processus actif doit se trouver dans l'un de ses trois états :

- ✓ Un processus actif est dit élu s'il est en mémoire centrale (c'est-à-dire en cours d'exécution)
- ✓ Un processus est dit prêt s'il est suspendu provisoirement en attente d'exécution pour permettre l'exécution d'un autre processus.
- ✓ Un processus est dit bloqué s'il attend un évènement extérieur pour pouvoir continuer.

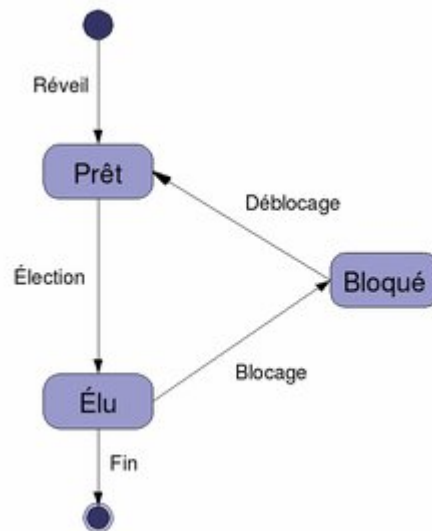


Figure 8 – Les états d'un processus

Après cette brève présentation sur la notion de processus, nous nous penchons dans ce qui suit sur la synchronisation entre les différents processus au sein du système d'exploitation. Ce synchronisme est

nécessaire et utile à mettre en œuvre dans le cas d'un passage à l'échelle, autrement dit, lorsque le nombre d'utilisateurs Bluetooth augmente et créant ainsi plusieurs processus qui veulent accéder à la même ressource pour l'utiliser simultanément. Par exemple, il se peut que plusieurs processus accèdent en même temps à la base de données en écriture et en lecture, ce qui provoquera une mauvaise exécution du code.

Pour notre projet, nous n'allons pas prendre en considération ce critère puisque nous travaillons sur une petite échelle qui ne nécessite pas une gestion de synchronisation entre les processus créés.

E/ La synchronisation des processus : exclusion mutuelle

Définition de l'exclusion mutuelle

Lorsqu'une ressource est critique, c'est-à-dire qu'elle possède qu'un seul point d'entrée, on parle d'exclusion mutuelle.

On appelle section critique la partie d'un code où la ressource est seulement accessible par le processus en cours. Il faut s'assurer que deux processus n'entrent jamais en même temps en section critique sur une même ressource.

Il faut garantir qu'un seul processus entre dans la section critique. Mais pour que l'exclusion mutuelle soit réalisée, il faut que les conditions suivantes soient remplies :

- ✓ Tout moment, un processus au plus en section critique (exclusion)
- ✓ Si des processus demandent la section critique, et si la section critique est libre, alors l'un d'entre eux doit y rentrer au bout d'un certain temps (accès)
- ✓ Le blocage par cette section critique doit être indépendant des autres types de blocage : par exemple, si un processus est terminé, ou bloqué en attente d'une imprimante, il ne doit pas empêcher un autre processus d'utiliser un CD ROM (indépendance)
- ✓ Aucun processus ne doit jouer de rôle privilégié (uniformité)

Ceci dit, il existe plusieurs solutions qui permettent de garantir la synchronisation entre les différents processus, nous expliquerons dans ce qui suit la méthode des sémaphores qui a été utilisée l'année

dernière par Mohammed Chaari dans le cadre de sa thèse professionnelle.

Les sémaphores : définition

La notion de sémaphore a été introduite dans les années 1968 par Dijkstra.

On appelle sémaphore, le regroupement de :

- ✓ Une variable entière: $val(S)$
- ✓ Une file d'attente: $F(S)$
- ✓ Deux primitives P et V tels que: P décrémente $val(S)$ d'une unité et si $val(S)$ est inférieur à 0 alors le processus appelant la sémaphore est mis dans la file d'attente $F(S)$; V incrémente $val(S)$ d'une unité et que si $val(S)$ est inférieure ou égal à 0 alors on débloque un processus dans la file d'attente $F(S)$.

Avant d'entrer dans sa section critique, tout processus doit faire une demande explicite. Notons que cette demande peut être bloquante si jamais la section critique est occupée. Intuitivement cette demande se traduit par l'exécution de l'opération P . Le processus sortant doit libérer la section critique permettant à un autre processus bloqué à l'entrée d'y accéder. Cette libération se traduit par l'exécution de l'opération V , seule opération pouvant s'accompagner d'un déblocage de processus. Lors de l'utilisation d'un sémaphore dans le cas de l'exclusion mutuelle, la valeur entière de celle-ci doit être initialisée à « 1 ».

mutex = 1



Figure 9 – Fonctionnement des sémaphores mutex

Les sémaphores : explication du principe de fonctionnement

Un processus qui veut accéder à la ressource critique utilise le sémaphore mutex de la façon suivante: il teste la valeur entière du sémaphore afin de vérifier si la ressource est disponible ou pas. Si après exécution de P(mutex), mutex devient négatif, alors le processus en question sera placé en file d'attente sinon cela signifie que la ressource est libre et peut donc y accéder.

Une fois cela accomplie, il peut libérer la ressource en exécutant son V(mutex).

F/ [Les sockets](#)

Présentation générale

La notion de sockets a été introduite dans les distributions de Berkeley (un fameux système de type UNIX, dont beaucoup de distributions actuelles utilisent des morceaux de code), c'est la raison pour laquelle on parle parfois de sockets BSD (Berkeley Software Distribution).

Il s'agit d'un modèle permettant la communication inter processus (IPC - Inter Process Communication) afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP.

La communication par socket est souvent comparée aux communications humaines. On distingue ainsi deux modes de communication :

- ✓ Le mode connecté (comparable à une communication téléphonique), utilisant le protocole TCP. Dans ce mode de communication, une connexion durable est établie entre les deux processus, de telle façon que l'adresse de destination n'est pas nécessaire à chaque envoi de données.
- ✓ Le mode non connecté (analogue à une communication par courrier), utilisant le protocole UDP. Ce mode nécessite l'adresse de destination à chaque envoi, et aucun accusé de réception n'est donné.

Les sockets sont généralement implémentés en langage C, et utilisent des fonctions et des structures disponibles dans la librairie <sys/socket.h>

Déroulement d'une communication

Comme dans le cas de l'ouverture d'un fichier, la communication par socket utilise un descripteur pour désigner la connexion sur laquelle on envoie ou reçoit les données. Ainsi la première opération à effectuer consiste à appeler une fonction créant un socket et retournant un descripteur (un entier) identifiant de manière unique la connexion. Ainsi ce descripteur est passé en paramètres des fonctions permettant d'envoyer ou recevoir des informations à travers le socket.

L'ouverture d'un socket se fait en deux étapes :

- ✓ La création d'un socket et de son descripteur par la fonction `socket()`
- ✓ La fonction `bind()` permet de spécifier le type de communication associé au socket (protocole TCP ou UDP)

Un serveur doit être à l'écoute de messages éventuels. Toutefois, l'écoute se fait différemment selon que le socket est en mode connecté (TCP) ou non (UDP).

En mode connecté, le message est reçu d'un seul bloc. Ainsi en mode connecté, la fonction `listen()` permet de placer le socket en mode passif (à l'écoute des messages). En cas de message entrant, la connexion peut être acceptée grâce à la fonction `accept()`. Lorsque la connexion a été acceptée, le serveur reçoit les données grâce à la fonction `recv()`.

En mode non connecté, comme dans le cas du courrier, le destinataire reçoit le message petit à petit (la taille du message est indéterminée) et de façon désordonnée. Le serveur reçoit les données grâce à la fonction `recvfrom()`.

La fin de la connexion se fait grâce à la fonction `close()`.

Voici le schéma d'une communication en mode connecté:

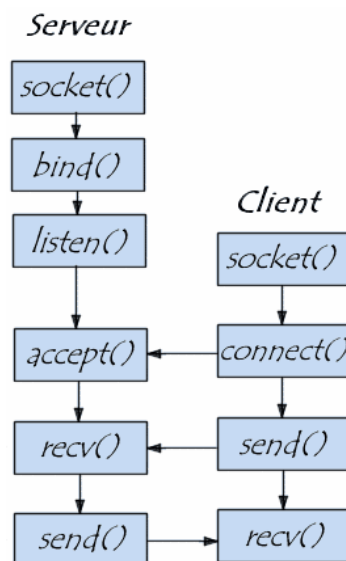
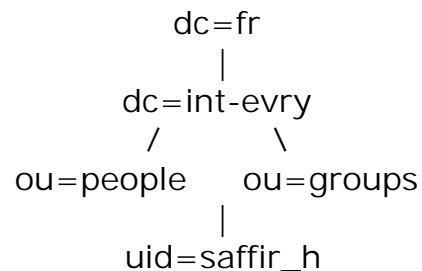


Figure 10 – Déroulement de la communication en mode connecté

G/ LDAP

LDAP (Lightweight Directory Access Protocol) est un protocole pour la gestion des services d'annuaire électronique. Ce dernier est un arbre d'entrées constituées elles-mêmes d'attributs, possédant chacun un nom, un type et plusieurs valeurs définis dans des schémas. Chaque entrée a un identifiant unique, le Distinguished Name (DN). Il est constitué à partir de son Relative Distinguished Name (RDN) suivi du DN de son parent. C'est une définition récursive. On peut faire l'analogie avec une autre structure arborescente, les systèmes de fichiers ; le DN étant le chemin absolu et le RDN le chemin relatif à un répertoire. En règle générale le RDN d'une entrée représentant une personne est l'attribut UID :



L'accès à la base de données LDAP nous a permis d'obtenir des informations concernant les futurs usagers du service BAB (leur école, leur année).

La base LDAP de l'ex-INT permet d'accéder à des informations concernant la scolarité des usagers du service BAB

L'accès aux données de la base LDAP est possible grâce à l'utilitaire "ldapsearch" sous Linux.

Voici un exemple d'utilisation:

```
$ldapsearch -x uid=saffir_h -D uid=mesbahi,ou=people,dc=int-evry,dc=fr -h ldap3.int-evry.fr -b
dc=int-evry,dc=fr -W title
# extended LDIF
#
# LDAPv3
# base <dc=int-evry,dc=fr> with scope subtree
# filter: uid=saffir_h
# requesting: title
# saffir_h, People, int-evry.fr
dn: uid=saffir_h,ou=People,dc=int-evry,dc=fr
title: CL_FI-E11

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

CHAPITRE 3 : le développement

A/ Le cas d'utilisation

Le diagramme de cas d'utilisation représente les relations entre les acteurs et les fonctionnalités du système.

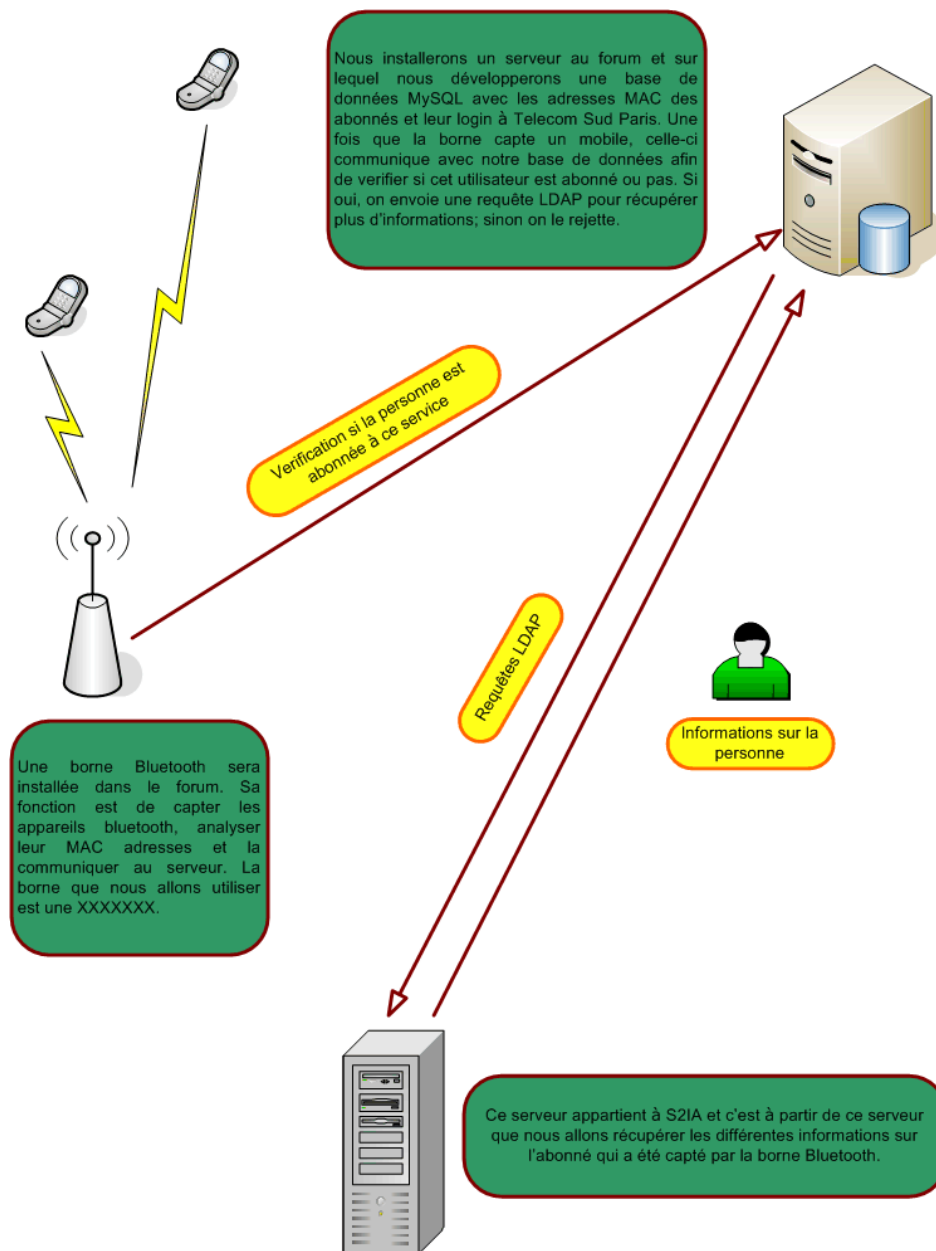


Figure 11 – Diagramme d'utilisation

Pour une question de bonne mise en page, nous avons réduit la taille de l'image. Pour une meilleure visibilité de la figure 10, voire l'annexe.

Dans ce qui suit nous allons traiter une à une les étapes mises en œuvre par l'application.

Première étape : la détection des appareils environnant

La détection est la fonction primaire de la borne.

Nous développerons un code qui sera lancé sur la borne et qui grâce à l'utilitaire BTCLI lancera la fonction de détection propre aux puces Bluetooth.

Ce qui nous intéresse à ce niveau est de pouvoir récupérer les différentes adresses physiques des appareils Bluetooth environnant.

Deuxième étape : la procédure de vérification

Durant cette étape, la borne envoie les adresses physiques captées vers le serveur, où elles seront comparées avec une base de données MySQL. La comparaison permet de vérifier si un appareil est abonnée au service ou pas.

Si l'adresse physique reçue existe déjà, c'est-à-dire que l'utilisateur en question est souscrit au service ; sinon il ne l'est pas.

Troisième étape : connexion au serveur GASPAR

Une fois que la vérification est réalisée, le code que nous implémenterons sur le serveur, doit communiquer avec le serveur GASPAR pour récupérer plusieurs informations concernant l'utilisateur souscrit au service. Par exemple, nous pouvons récupérer l'année d'étude, l'adresse email à Telecom Sud Paris, etc.

[B/ La conception préliminaires](#)

Les installations nécessaires

- Le package MySQL
 - mysql.i386 version 5.0.51
 - mysql-devel.i386 version 5.0.51
 - mysql-server.i386 version 5.0.51
 - mysql-client.i386 version 5.0.51
 - mysql-shared.i386 version 5.0.51
 - mysql-test.i386 version 5.0.51

➤ Le package phpMyAdmin

Pour faciliter la gestion de notre base de donnée, nous avons installé phpMyAdmin et dont l'accès se fait- en temps que « root » - en tapant l'URL <http://localhost/phpMyAdmin/index.php>

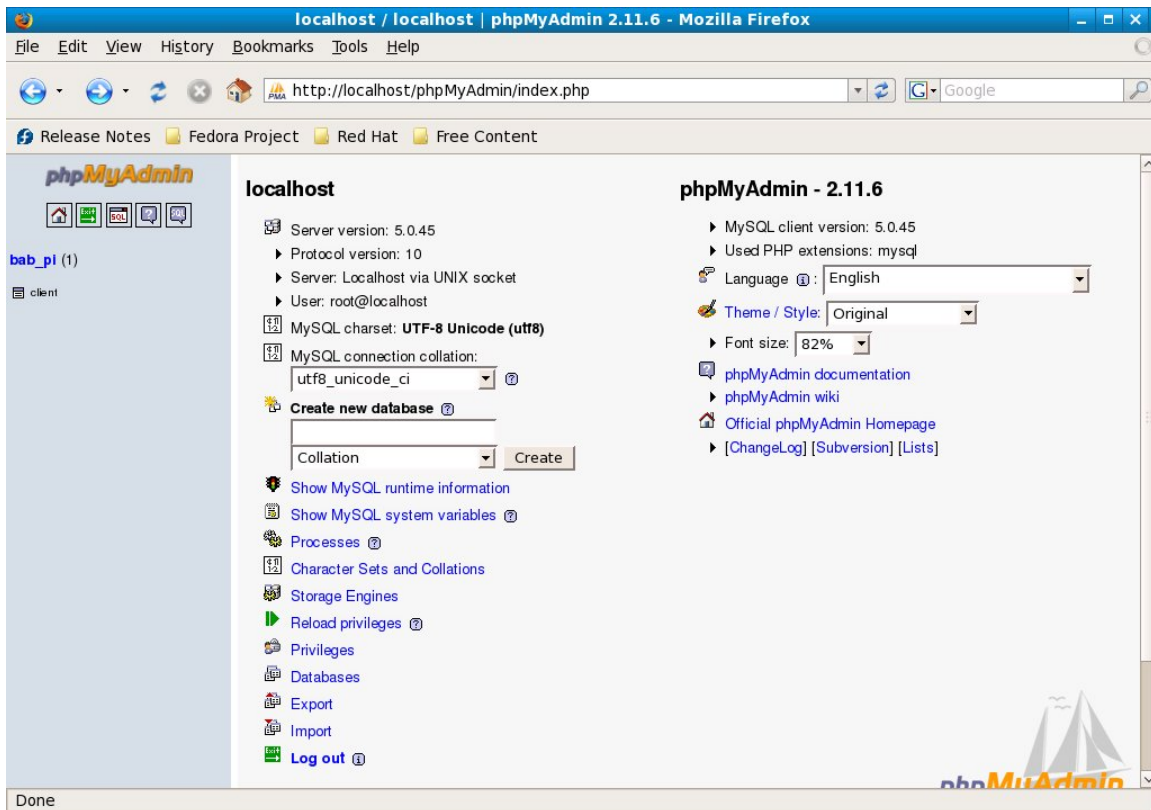


Figure 12 – La page d'accueil de phpMyAdmin

Création de la base de donnée MySQL

Notre base de donnée « bab_pi » est constituée d'une seule table et que nous avons appelé « client ». Cette dernière identifie tous les clients souscris au service par l'adresse MAC de leur portable et par leur login à Telecom Sud Paris.

Pour un tutoriel complet comment utiliser phpMyAdmin afin de créer, modifier, supprimer une base de donnée, veuillez visiter le site : <http://www.siteduzero.com/tuto-3-4-0-un-site-dynamique-avec-php.html>

Voici à quoi ressemble notre base de donnée :

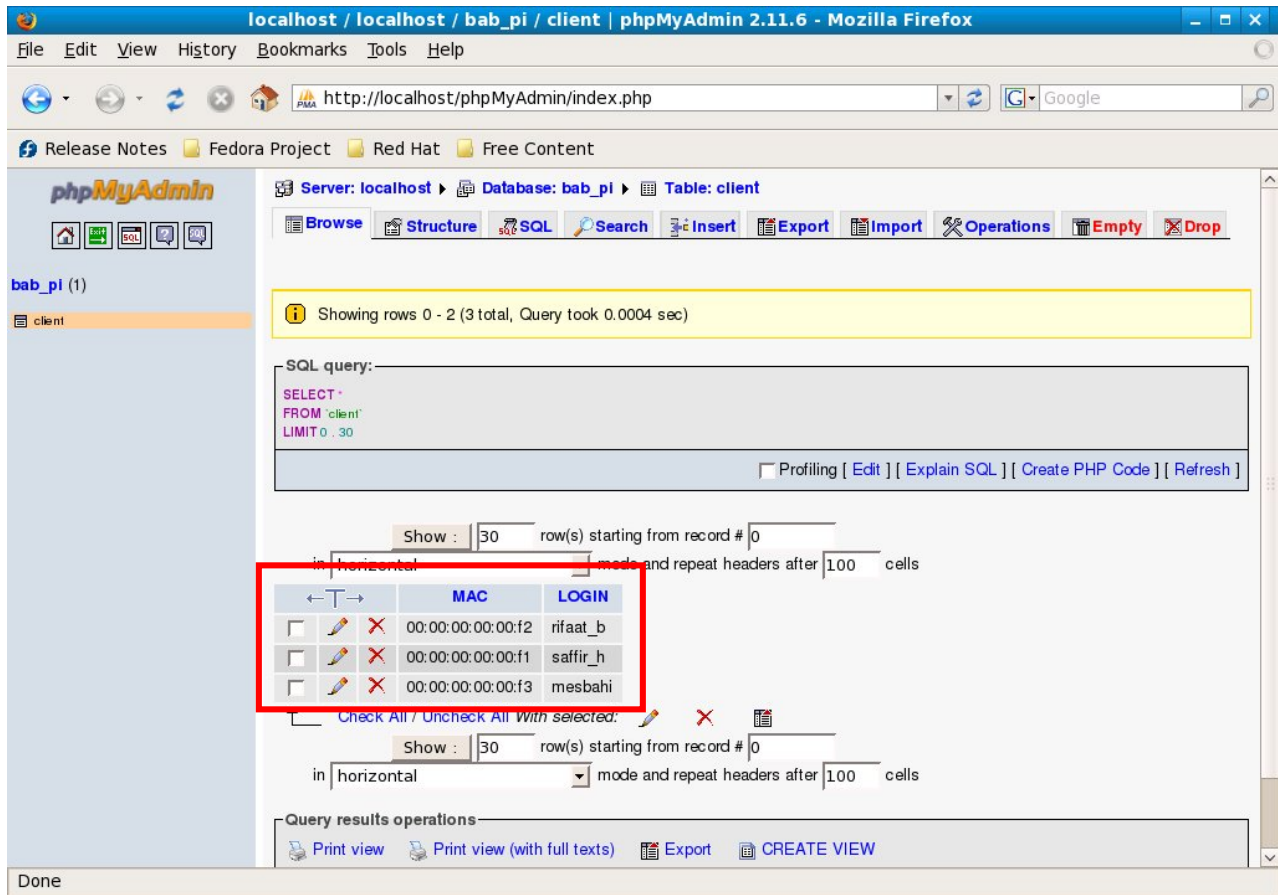


Figure 13 – Exemple de base de donnée

C/ La conception détaillée

Code côté borne

Comme nous l'avons signalé auparavant, le code sur la borne doit permettre la détection et l'envoi des adresses physiques captées vers le serveur MySQL.

Pour ce faire, la fonction détection repose principalement sur l'utilitaire BTCLI.

Le BTCLI est une application qui permet d'envoyer des commandes à la borne Bluetooth et dont les résultats seront affichés sur la sortie standard.

Ceci dit, la commande qui permet de faire la détection est : inquiry

Dans notre code nous l'avons implémentée de la façon suivante:

```
Btcli inquiry 4 -port 10102 | sed '/INQUIRY_PARTIAL/ !d' | awk '{print $2}' > mac.txt
```

Afin de récupérer que la partie qui nous intéresse du resultat de la commande inquiry, nous avons fait appel aux commandes SED et AWK.

La commande sed est généralement utilisée pour modifier l'affichage d'une ligne. Dans notre cas, nous avons demandé à ce que le programme n'affiche que les lignes commençant par INQUIRY_PARTIAL.

Par ailleurs, la commande awk est définie comme un mini programme qui exécute plusieurs instructions. Dans notre cas d'utilisation, cette commande va afficher que le champ numéro 2 de chaque ligne, et qui correspond à l'adresse physique de l'appareil Bluetooth détecté.

Enfin, les adresses ainsi récupérées seront redirigées (« > ») vers un fichier mac.txt.

Une fois que le fichier contenant les adresses physiques a été créé au niveau de la borne Bluetooth, il ne reste plus qu'à l'envoyer vers le serveur MySQL.

Pour ce faire, nous avons développé une fonction qui permet, à travers d'un socket, d'envoyer une à une les adresses du fichier mac.txt.

Dans le code, nous pouvons distinguer les parties suivantes :

- ✓ La création de la socket
- ✓ La demande de connexion au serveur MySQL
- ✓ La lecture du fichier mac.txt
- ✓ L'écriture sur la socket

Les étapes de l'établissement de la connexion avec le serveur sont mises en œuvre de la façon suivante :

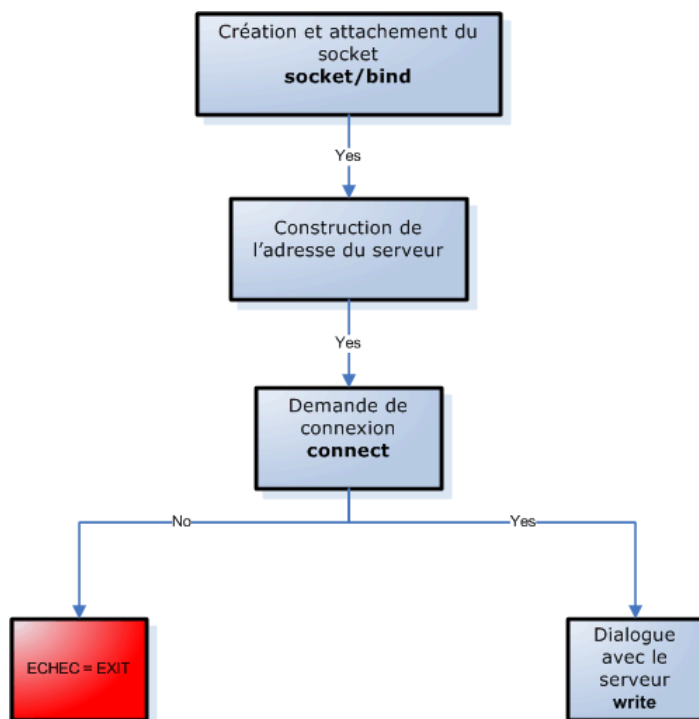


Figure 14 – Etablissement de la connexion socket

Les primitives mises en jeu dans cette partie du code sont :

- ✓ La primitive socket qui permet la création du socket et qui renvoie le descripteur de ce socket.
- ✓ La primitive connect qui réalise l'établissement d'une connexion bidirectionnelle de bout en bout entre le client et le serveur en associant socket serveur et socket client
- ✓ La primitive write qui permet d'écrire dans les sockets (envoyer des messages TCP)

Avant l'écriture dans le socket, nous avons décidé de faire une lecture ligne par ligne du fichier mac.txt avec la fonction fgets. Pour cela, nous avons imposé que la taille de chaîne de caractère lue sur chaque ligne soit de 18 caractères, c'est-à-dire les 17 caractères constituant l'adresse physique + le caractère « \n ».

Ainsi, ce sont ces 18 caractères qui sont envoyés dans le socket vers le serveur MySQL.

Code côté serveur

Pour le serveur, le code que nous avons développé est divisé en trois grandes parties :

- ✓ La réception des adresses physiques et création d'un fichier mac.txt
- ✓ La vérification des adresses reçues avec la base MySQL
- ✓ Communication avec GASPARD via le protocole LDAP

Premièrement, pour que la réception soit effective, nous allons faire en sorte que le serveur soit toujours en écoute sur son socket de tout éventuel message émis par la borne.

Pour ce faire - comme pour la borne – une demande d'établissement d'une connexion doit être effectuée suivant le diagramme suivant :

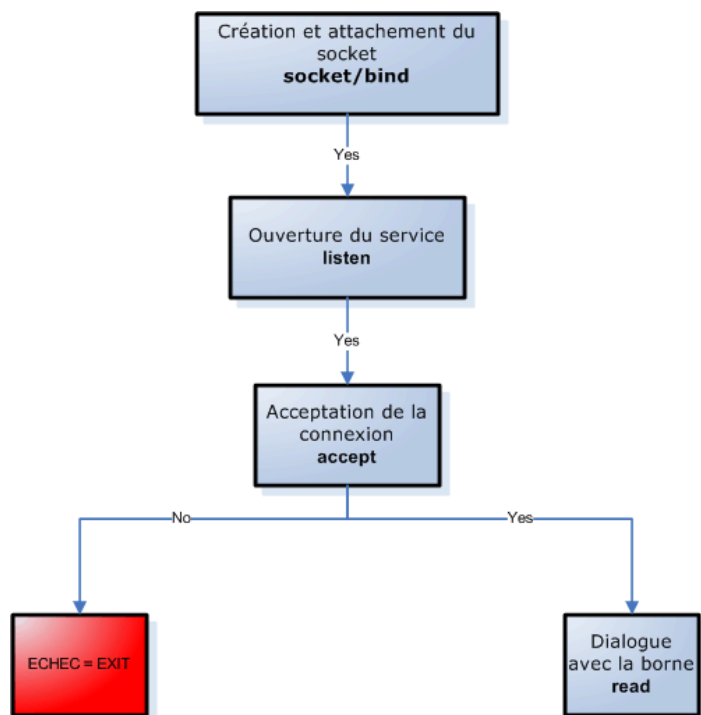


Figure 15 –
Etablissement de la
connexion côté serveur

La primitive listen permet d'indiquer que le serveur est en état d'écoute sur un port TCP.

Lorsqu'un message est détecté, la lecture de ce message est réalisée par la primitive read. Ensuite, une fonction search_sql est appelée avec le message lu sur la socket en paramètre.

La fonction search_sql permet comme son nom l'indique, de rechercher si l'adresse physique reçue par la borne existe déjà dans notre base de donnée.

Rappelons que la base de donnée, contient les adresses physiques des appareils Bluetooth des personnes souscrites à notre service et leur login (celui de l'élève à Telecom Sud Paris). Si jamais l'adresse existe, la fonction renvoie le login associé à cette adresse ; sinon on affiche le message suivant : « Adresse MAC non reconnue ! »

Enfin, une dernière partie du code côté serveur concerne la communication avec le serveur GASPARD du service S21A.

Rappelons que l'objectif principal de notre projet est de pouvoir récupérer des informations supplémentaires sur l'utilisateur souscrit au service grâce à GASPARD.

Pour récupérer ces informations, nous avons développé la fonction ldap_s_b qui prend en paramètre le login renvoyé par la fonction search_sql.

Dans ce qui suit, nous allons clarifier le principe de fonctionnement de cette fonction.

Le makefile

```
DEBUG=yes
SERVER=yes
WRAP=yes
CC=gcc

ifeq($(DEBUG),yes)
CFLAGS=-Wall -g
@echo "mode DEBUG"
else
CFLAGS=-Wall
@echo "mode release"
endif

DFLAGS_LDAP=-lldap -D LDAP_DEPRECATED
DFLAGS_SQL=-lmysqlclient

ASDK_CC=/usr/local/arm/2.95.3/bin/arm-linux-gcc
ASDK_CFLAGS=-Os -Wall -fomit-frame-pointer -D__linux__
ASDK_LIB=-L/root/asdk/lib
ASDK_INCLUDE=-I/root/asdk/include
ASDK_STRIP=/usr/local/arm/2.95.3/bin/arm-linux-strip

ifeq($(SERVER),yes)

@echo "compilation du code source SERVER"

server: server.c
    $(CC) $(CFLAGS) $(DFLAGS_LDAP) $(DFLAGS_SQL) $^ -o $@
stripSERVER:
    strip server
endif

ifeq($(WRAP),yes)

@echo "compilation du code source WRAP"

wrap: wrap.o
    $(ASDK_CC) $(ASDK_CFLAGS) $(ASDK_LIB) -o $@ $^

wrap.o: wrap.c
    $(ASDK_CC) $(ASDK_CFLAGS) $(ASDK_LIB) -c -o $@ $^

stripWRAP:
    $(ASDK_STRIP) wrap

clean:
    @rm -rf *.o
endif
```

CHAPITRE 4 : implémentation et tests

Dans ce chapitre, nous allons tout d'abord présenter l'environnement logiciel sur lequel nous avons implémenté et tester notre application. Ensuite, nous exposerons les tests que nous avons réalisés sur la borne et sur le serveur et enfin, nous terminerons par une partie concernant les difficultés rencontrées.

A/ L'environnement du travail

Nous avons développé notre application sous un environnement UNIX, on a développé avec le langage C. Nous avons utilisé en plus les outils suivants.

Le compilateur GCC

En informatique, GCC, abréviation de GNU Compiler Collection, est le compilateur créé par le projet GNU.

Il s'agit d'une collection de logiciels libres intégrés capables de compiler divers langages de programmation, dont C, C++, Objective-C, Java, Ada et Fortran.

GCC est utilisé pour le développement de la plupart des logiciels libres. GCC a été conçu pour remplacer le compilateur C fourni en standard sur le système d'exploitation Unix, qui s'appelle CC. GCC signifiait à l'origine GNU C Compiler, soit le " compilateur C de GNU ".

ASDK

ASDK est l'abréviation de " Alternative Software Development Kit ", il s'agit d'un kit de développement prévu spécialement pour les bornes " Bluegiga ". Ce kit permet de générer un exécutable qui s'exécute sur la plateforme des bornes.

B/ La phase des tests

Tout au long de la phase développement de l'application, nous avons réalisé des tests sur chaque partie dans le but de contrôler le bon fonctionnement de l'application finale. Dans la suite nous exposerons les résultats issus de l'exécution du code sur le serveur et sur la borne.

Test du code côté borne

```
[program@localhost code_finale]$ ./envoi 157.159.16.198
La socket 3 est maintenant ouverte en mode TCP/IP.
La connexion au serveur 157.159.16.198 sur le port 34835 est etablie!
Deroulement de la lecture ligne par ligne...
00:00:00:00:00:ff
00:00:00:00:00:f1
00:00:00:00:00:f2
Fermeture du fichier.
Fermeture de la socket
L'emission est effectuee.
[program@localhost code_finale]$ █
```

Afin de tester la fonction envoie du code, nous avons écrit manuellement un fichier mac.txt contenant différentes adresses MAC, comme nous pouvons le voir sur la figure précédente. Au fur et à mesure qu'une lecture du fichier est effectuée, la ligne lue est émise vers le serveur en question (ici l'adresse IP du serveur est 157.159.16.198). A la fin de la lecture, le fichier et la socket sont fermés.

Test du code côté serveur

Comme le code du serveur contient des traitements de requêtes SQL et des requêtes LDAP, il le fallait compiler de la façon suivante :

```
gcc -Wall -g -lldap -D LDAP_DEPRECATED -lmysqlclient reception.c -o reception
```

En parallèle avec l'émission des adresses du fichier mac.txt, le serveur les reçoit sous forme de message (une adresse par message), et les traite une à une.

Dans le test ci-dessous, la première adresse reçue par le serveur est la 00 :00 :00 :00 :00 :ff. Ensuite, ce dernier se connecte à la base de donnée MySQL pour vérifier si cette adresse existe ou pas. Comme

cette adresse n'existe pas, elle est alors rejetée. Le serveur le fait savoir par « L'adresse MAC non reconnue ! »

Le serveur traite alors le deuxième message reçu (ou la deuxième adresse physique reçue). Comme l'adresse 00:00:00:00:00:f1 existe dans la base de donnée, le programme extrait le login associé à cette adresse et fait appel à LDAP. Dans la partie verte, il s'agit d'une connexion LDAP. Le paramètre étant le login récupéré de la base de donnée. Nous pouvons observer les différentes étapes mises en jeu (connexion et filtrage) et le résultat renvoyé.

```
[Farouk@(none) Desktop]$ reception_stable
Le serveur veut recevoir des données...
La socket 3 est connectee en mode TCP/IP.
Attente de connexion...

Ecoute sur le port 5000...
accept...!!!
```

Le serveur est en attente
d'un message

```
received message: 00:00:00:00:00:ff
Attente de la connexion a la base SQL...
Connexion a la base reussie!
La base demandee existe!
adresse mac non reconnu !!!received message: 00:00:00:00:00:f1
Attente de la connexion a la base SQL...
Connexion a la base reussie!
La base demandee existe!
```

Connexion a la BD
et vérification

```
search=saffir_h
Connecting to host ldap1.int-evry.fr at port 389...

Searching the directory for entries
starting from the base DN ou=people, dc=int-evry,dc=fr
within the scope 2
matching the search filter uid=saffir_h...

dn: uid=saffir_h,ou=People,dc=int-evry,dc=fr
title: CL_FI-FI1
```

Envoie requête LDAP
vers le serveur de
S2IA avec le login en
paramètre

C/ Les difficultés rencontrées

Au cours du développement de l'application, nous avons rencontré un petit problème technique avec la borne. En effet, cette dernière ne pouvait être branchée que dans le laboratoire. Ceci freinait l'avancement du projet puisque nous n'avions pas toujours accès à cette salle.

Pour surmonter ce problème, nous avons eu recours à la fonction netcat de Unix, qui nous a permis de simuler des envoies vers le serveur. De plus, comme notre fonction envoie n'était pas tout à fait mise au point, nous avons réussi à développer le code réception en s'appuyant sur des tests avec netcat et sans avoir recours au code envoie qui était en train d'être développé en parallèle.

Par ailleurs, comme nous l'avons dit auparavant, notre projet reprend une étude qui a été développée dans le cadre d'une thèse professionnelle. L'application était gérée par un code de plus de 1000 lignes ! Nous avons pensé tout d'abord à étudier les fonctions principales de ce code afin de comprendre comment l'application était construite. Mais c'était compliqué car nous sommes tombés sur des notions nouvelles telles que sémaphores, mutex, LDAP etc. Après avoir réalisé des recherches pour comprendre ces différentes notions, nous avons essayé de simplifier au maximum le code, sachant que notre application va tourner sur une beaucoup plus petite échelle.

Nous arrivons ainsi à la fin de ce chapitre. Nous avons présenté l'environnement logiciel utilisé, nous avons aussi montré quelques exemple d'exécutions au coté serveur ainsi que client, et enfin, nous avons résumé les difficultés que nous avons rencontré lors de la réalisation de ce projet.

CONCLUSION

L'application développée permet de détecter, via une borne Bluetooth, les téléphones portables voisins (à la borne), d'envoyer leurs adresses MAC au serveur. De là, au travers de la base SQL, on vérifie si l'adresse MAC correspond à un identifiant connu. Si tel est le cas, on fait une requête LDAP pour avoir plus de détails sur cette personne (son login, son école, ainsi que son année).

Pour améliorer le fonctionnement de l'application, on pourrait notamment utiliser des « forks » qui permettront d'optimiser le temps de détection. Ces « forks » ont pour but de créer des processus fils, qui auront pour rôle de rechercher dans la base SQL ainsi la fonction « réception » et « search_sql » se feront en parallèle. Pour ce faire, il faudra utiliser des « pipes » qui empileront les adresses MAC reçues. Afin de réguler l'accès à ces « pipes », il est nécessaire d'utiliser des sémaphores (mutex).







Le projet informatique fut pour l'ensemble de notre équipe la première expérience d'un projet informatique en équipe de grande envergure. Tout au long de ce projet, nous avons énormément appris et progressé sur le plan technique. En effet notre projet touchait des domaines que nous ne maîtrisions pas, ou que nous connaissions que très sommairement. Ainsi, nous avons pu découvrir les bases de données SQL, les requêtes client/serveur, la syntaxe de LDAP, et maîtriser d'avantage les différents outils de développement qu'offrent Linux (le « débogage » DDD, wireshark, netcat).

Ce fut également une expérience humaine enrichissante qui est, à nos yeux, un premier pas vers le monde du travail, où nous serons amenés à réaliser des projets en équipe. Nous avons pu apprendre des uns et des autres, s'entraider, critiquer et conseiller afin de mener à bien le projet. Cela nous a permis aussi de démontrer à nous-mêmes que nous étions capables de créer des applications plus au moins « complexes » et de démystifier ainsi la programmation.



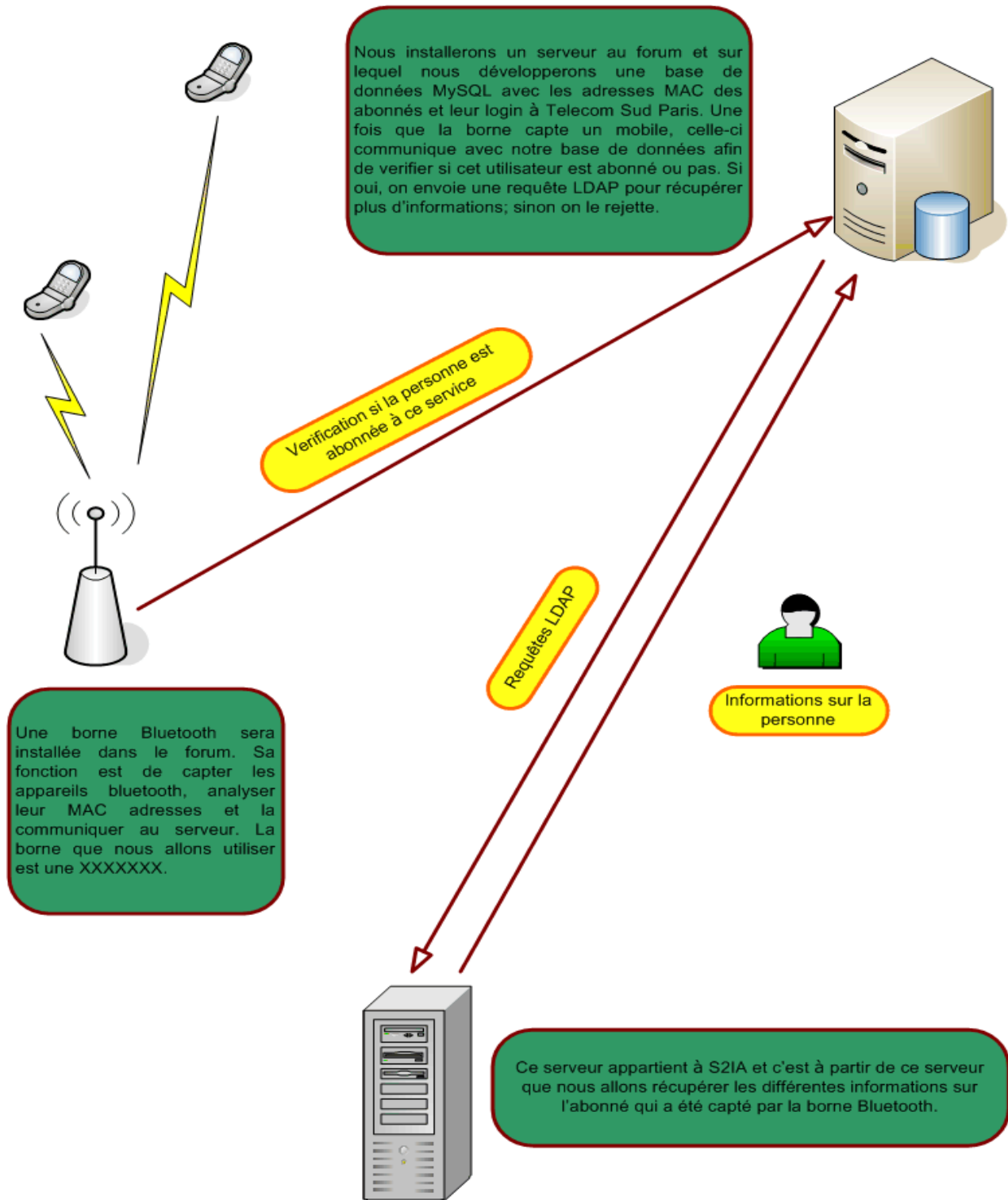
Le code source

Le planning prévisionnel

	Février	Mars	Avril	Mai	Juin
Documentation					
Spécification					
Conception					
Réalisation					
Test					
Rapport et soutenance					

ANNEXE

A/ Le diagramme d'utilisation



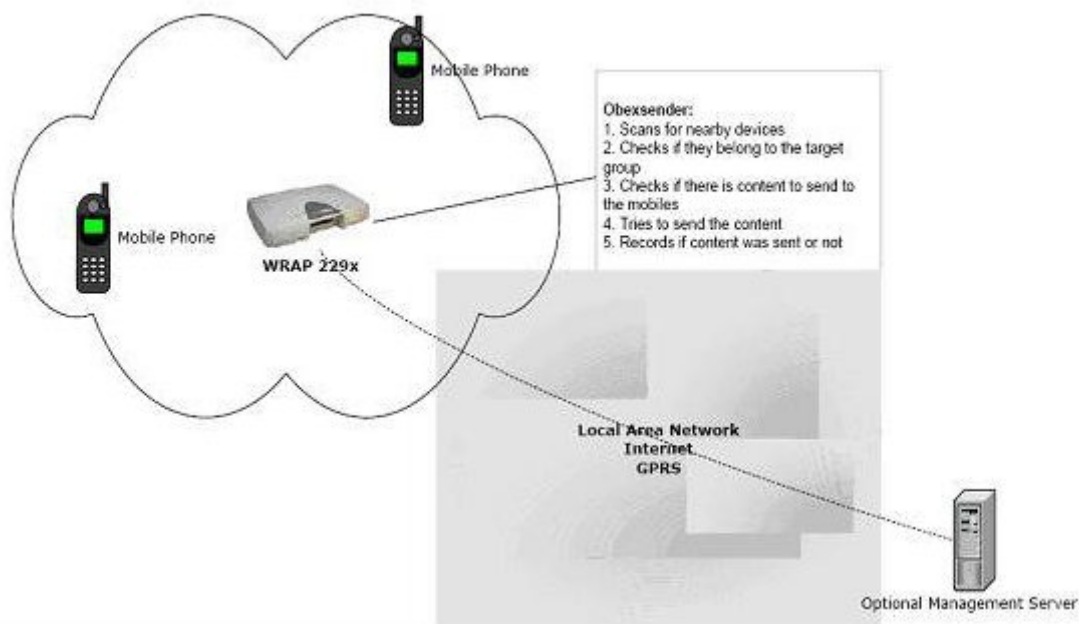
B/ Le WRAP Access Server

Domaines d'utilisation principaux

- ✓ Réseaux de point de vente,
- ✓ Systèmes de logistique et de transport,
- ✓ Systèmes de télémétrie,
- ✓ Systèmes médicaux,
- ✓ transfert des contenus sur les PDAs et téléphones portables.

Données Techniques

- Réseau
 - Bluetooth 1.1 et 1.2
 - RFCOMM, HCI, L2CAP, SDP, LAP, SPP, PAN, FTP,
 - Point-point, point-multipoint, piconet et scatternet
 - Classe1 jusqu'à 100 mètres, et peut être configuré pour 10m (classe 2)
 - Supporte TCP/IP
 - carte compact flash optional
 - WiFi 802.11 a/b
 - GSM et GPRS
 - Télécommande et monitoring



WRAP Access Server et son environnement

- Logiciel
 - Logiciel Bluegiga WRAP
 - Système d'exploitation Linux
 - Serveurs Bluetooth, HTTP, FTP et Telnet
 - Client et serveur SSH
 - Client et serveur DHCP
 - WRAP SMS Gateway
 - Plusieurs modèles d'application et de configuration

- Matériel
 - Consommation d'énergie dépendant de l'application
 - Mémoire : 323 MB RAM, 16 MB Flash
 - Horloge, batterie
 - Température de fonction : de 0 à +55°C
 - Système de protection
 - Taille : 130x80x35 mm
 - Poids : 450 g

- Interface/Connectique
 - Ethernet 10/100 Mbps, RJ-45
 - Port de carte compact (WiFi et GSM/GPRS)
 - USB
 - port série, RS-232, D9
 - Alimentation, 9-24 VDC, 400 mA

- Développement d'application
 - Milieu approprié pour Linux
 - Supporte du C et C++