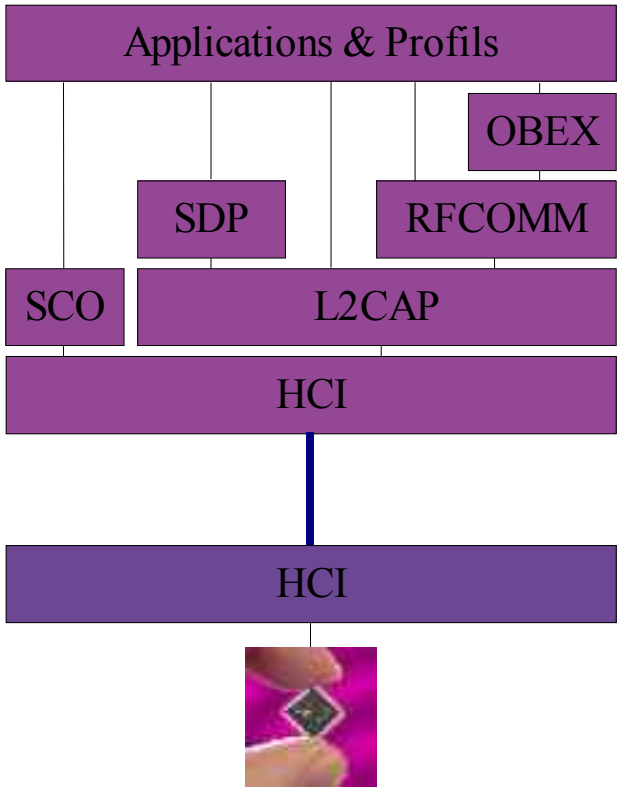
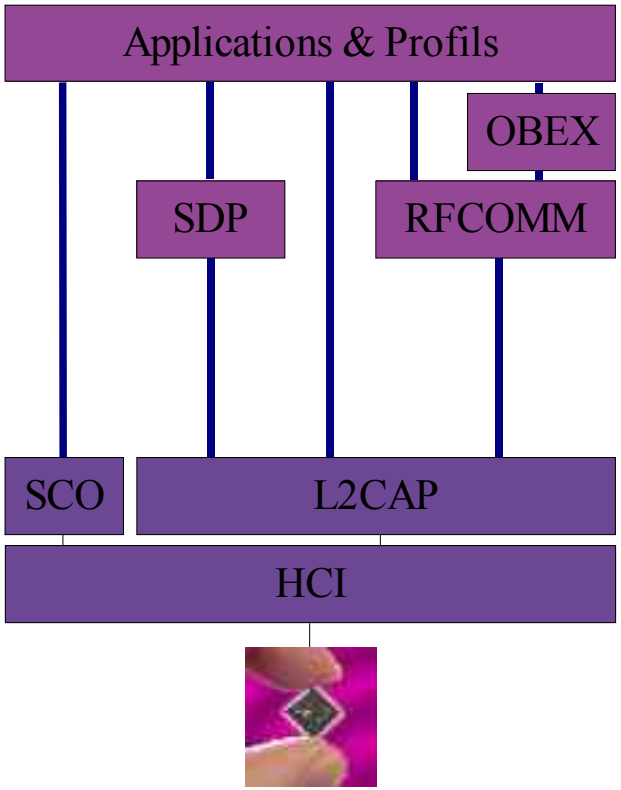
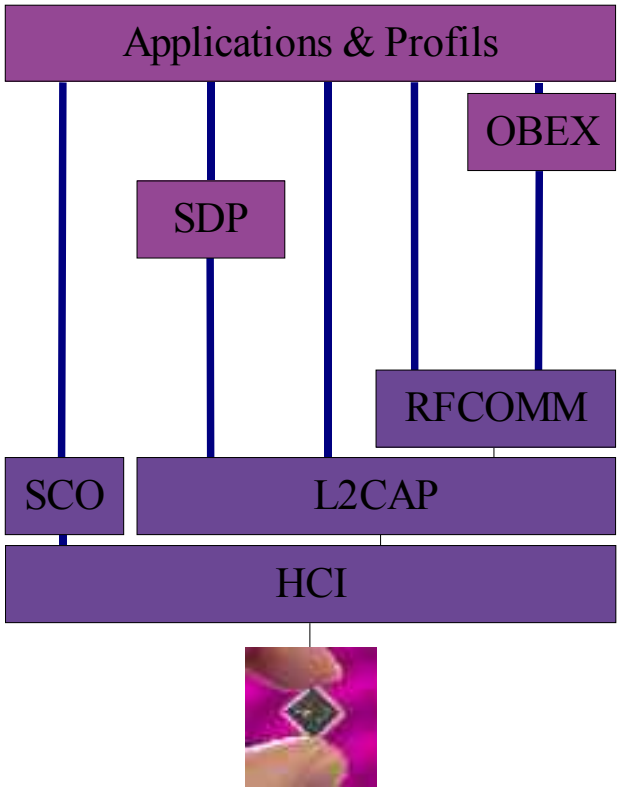
Développement de couches de la pile bluetooth
- logiciel embarqué
- différents composants bluetooth
  - couches partiellement intégrées dans la pile
  - développement des couches manquantes







Pilote de périphérique

Interface HCI

Couche L2CAP

Couche RFCOMM

Serveur SDP

Client SDP

Profils et applications

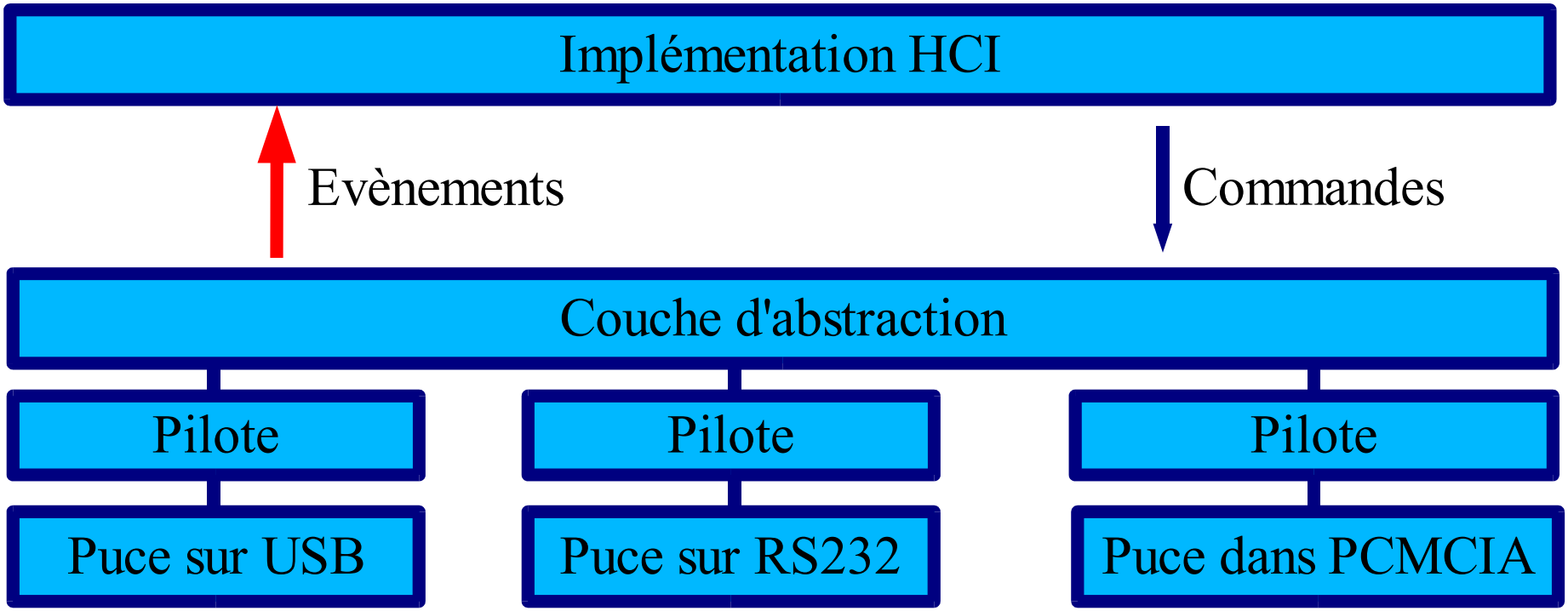
Qualification !

=> droit d'utilisation de la norme

=> droit d'utilisation du logo

=> qualification au niveau profil (GAP, SDAP, ...)

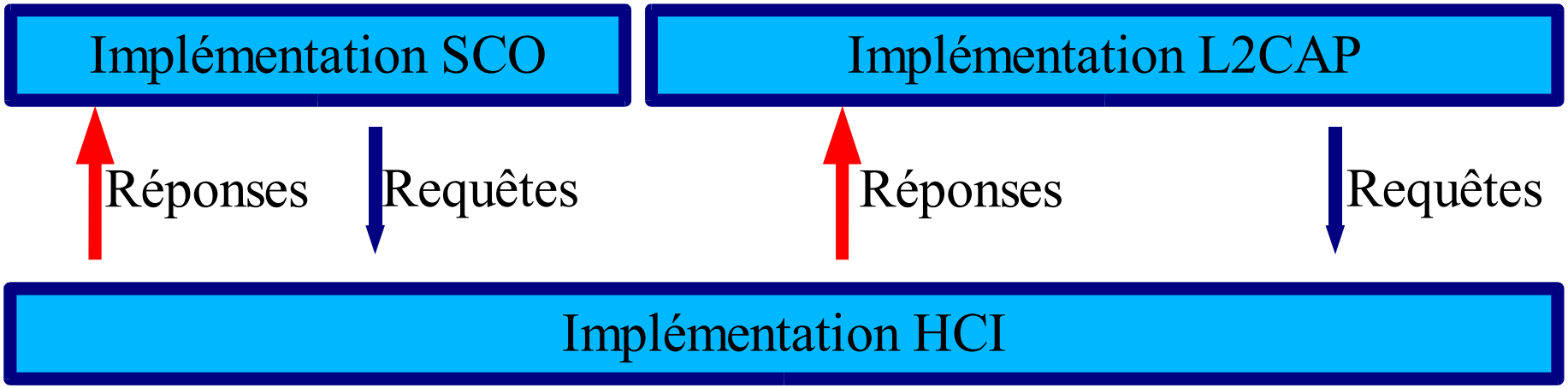
=> Tout profil implémenté doit être conforme et interopérable.


Implémentation HCI
Evènements
Commandes
Couche d'abstraction
Pilote
Pilote
Pilote
Puce sur USB
Puce sur RS232
Puce dans PCMCIA

L2CAP :

Commandes:
- Connexion
- Configuration
- Envoi de données
- Déconnexion

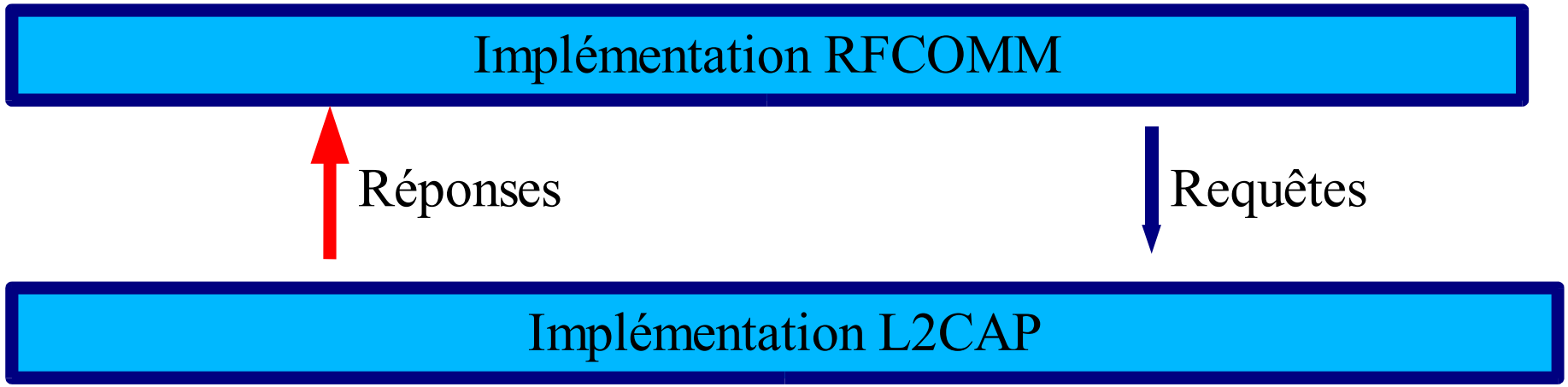
Evènements:
- Connexion
- Configuration
- Arrivée de données
- Déconnexion



RFCOMM :

Commandes & évènements :

- Création de session
- Connexion de canal
- Configuration de canal
- Envoi de signaux
- Envoi de données & négotiation de crédits
- Déconnexion

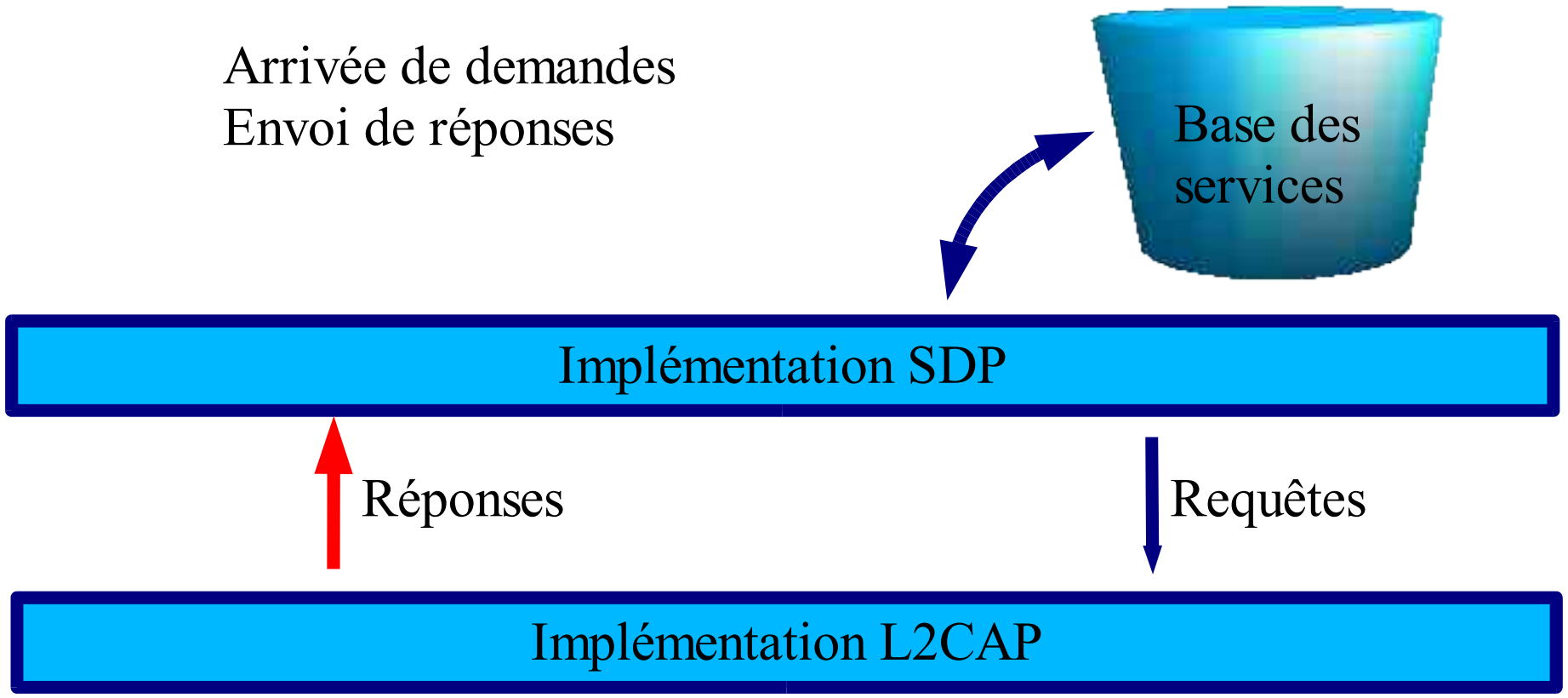


SDP :

Commandes & évènements :
- Recherche de services
- Arrivée de réponses

- Arrivée de demandes
- Envoi de réponses

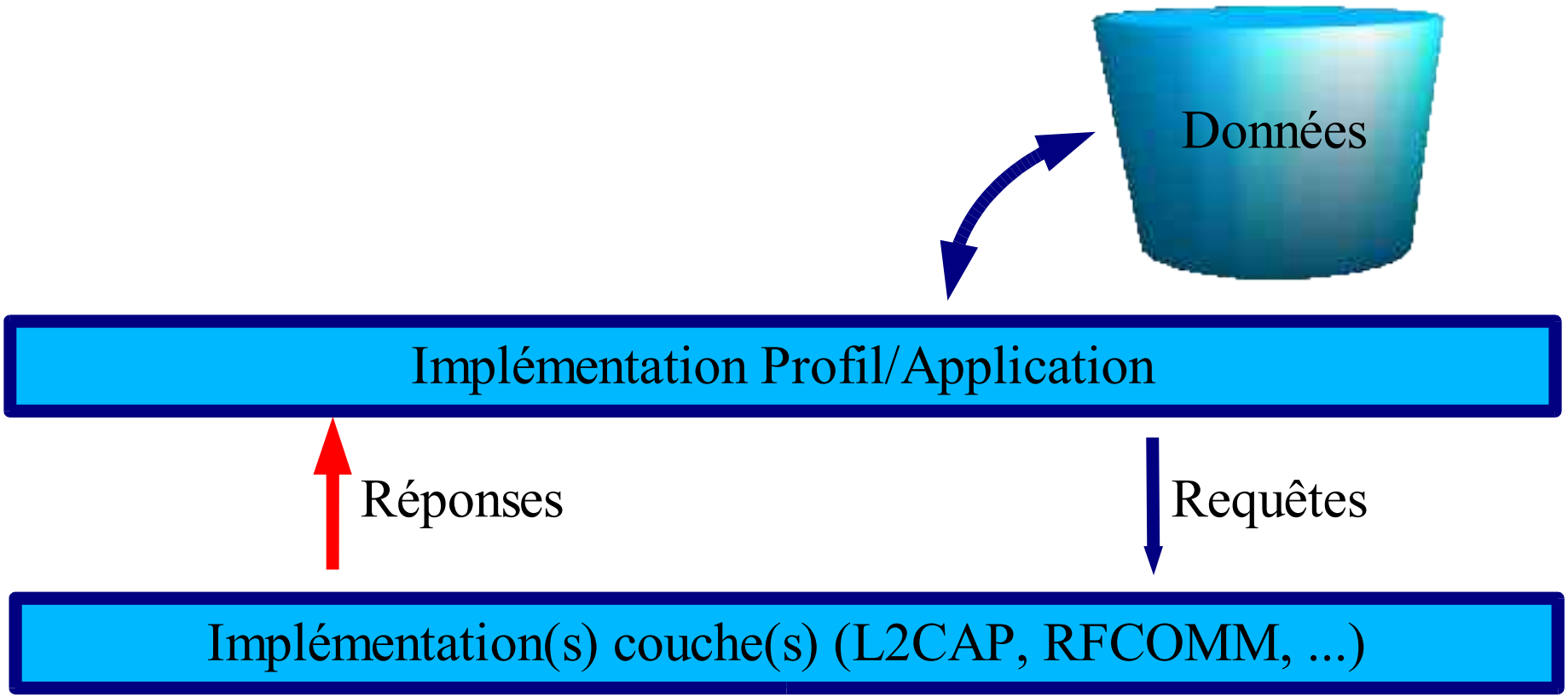
Utilitaires :
- Enregistrement de service
- Suppresion de service



Profils et applications :

Commandes & évènements

Utilitaires



Un exemple embarqué

Montre pour enfants

Contraintes encombrement/RAM/CPU

Absence de Système d'exploitation

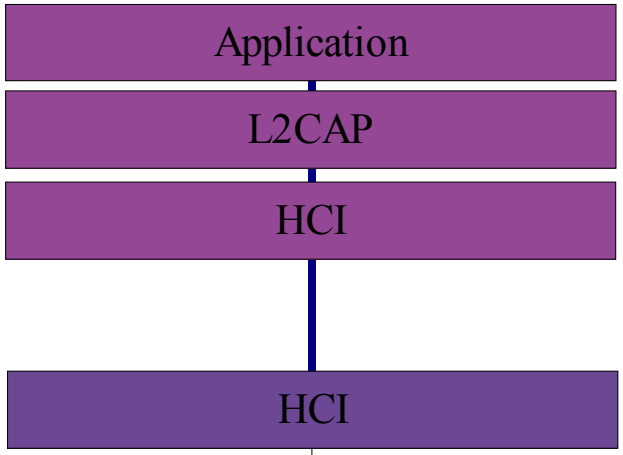
Pile développée en ASM

Compatible profil GAP

Protocole sur liaison L2CAP

Pas de SDP

PSM fixe



CSR BlueCore

BlueZ :

- pile bluetooth du noyau linux (2.4.x et 2.6.x)

Sur toutes architectures supportant linux :

- Processeurs x86
- AMD64 (x86-64)
- SUN SPARC 32/64bit
- PowerPC 32/64bit
- Intel StrongARM and XScale
- Motorola DragonBall
- ...

PC, PowerPC, PocketPC, Téléphones et PDA

Harald Blatand
(Harold Bluetooth en anglais)
Roi Danois du X^{ème} siècle
A unifié des forces de
Suède, Norvège, Danemark

Bluetooth permet à des périphériques différents de travailler ensemble.

D'où le nom de la norme !

Autres initiatives de support bluetooth pour linux :

OpenBTStack (AXIS) – 1999 – Mis à jour en 2001

BlueDrekar (IBM) – 2000 – Mis à jour en 2001

Nokia Affix – 2001 – encore actif

Bluez – 2001 – encore actif – pile officielle

Bluez commencé en 2001

- Après Affix
  - Affix non GPL, passé GPL depuis

- Société Qualcomm

- Marcel Holtmann mainteneur depuis 2003

Implémentation mixte
- Couches protocolaires dans le kernel
  (HCI, L2CAP, RFCOMM, BNEP, ...)
- Outils utilisateurs et démons
  (hciconfig, hcitool, sdptool, sdpd, rfcomm, ...)
- APIs pour développeurs

APIs Socket
- HCI – RAW
- L2CAP
  - SEQPACKET – connecté
  - DGRAM – non connecté
- SCO – SEQPACKET
- RFCOMM – STREAM
  - Conversion possible en port série (TTY)

APIs utilitaires
- gestion des périphériques, adresses
- gestion sdp

Code intégralement libre, sous GPL ou LGPL

Implémentation des versions 1.0b et 1.1 de la norme

Support de la version 1.2 en cours

Listes de diffusion très actives, chez sourceforge.net
- bluez-users
- bluez-devel

Evolution rapide et support efficace

Exemples de codes simples et clairs

Utilise le protocole H:4 comme protocole interne

Configuration et interface semblable à tcp/ip avec hciconfig:

```
ZazouMobile:/home/zazou# modprobe hci-usb
ZazouMobile:/home/zazou# hciconfig hci0 up piscan
ZazouMobile:/home/zazou# hciconfig hci0 -a
hci0:   Type: USB
        BD Address: 00:A0:96:1F:B6:93 ACL MTU: 128:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:113 acl:0 sco:0 events:12 errors:0
        TX bytes:35 acl:0 sco:0 commands:9 errors:0
        Features: 0xff 0xff 0x05 0x00 0x00 0x00 0x00 0x00
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy:
        Link mode: SLAVE ACCEPT
        Name: 'bluez'
        Class: 0x000000
        Service Classes: Unspecified
        Device Class: Miscellaneous,
        HCI Ver: 1.1 (0x1) HCI Rev: 0x72 LMP Ver: 1.1 (0x1) LMP Subver: 0x72
        Manufacturer: Cambridge Silicon Radio (10)
```

Outils en ligne de commande pour :

- gestion périphériques (hciconfig, hcitool)
- gestion sdp (sdptool)
- gestion RFCOMM (rfcomm)
  - création serveur
  - connexion
  - association de ports virtuels

Fichiers de configurations pour stockage

- Nom de périphérique
- Pairing (code PIN)
- Cryptage des connexions

Programmation HCI utilitaire

```
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>

void main (int argc, char **argv)
{
        int dev_id, num_rsp, length, flags;
        inquiry_info *info = NULL;
        bdaddr_t bdaddr;
        int i;
        dev_id = 0; /* device hci 0 */
        length = 8; /* 10.24 seconds */
        num_rsp = 10;
        flags = 0 ;
        num_rsp = hci_inquiry (dev_id, length, num_rsp, NULL, &info, flags);
        for (i=0 ; i<num_rsp ; i++ ) {
                baswap (&bdaddr, &(info+i)->bdaddr);
                printf ("\t%s\n", batostr(&bdaddr)) ;
        }
        free (info);
}
```

Programmation HCI Raw

```
int sock, cmd_len;
unsigned char buf[256];
unsigned char cmd[256];
struct sockaddr_hci addr;
sock = socket(AF_BLUETOOTH, SOCK_RAW, BTPROTO_HCI)
addr.hci_family = AF_BLUETOOTH;
addr.hci_dev = 0;
bind(sock, (struct sockaddr *)&addr, sizeof(addr));
...
send (sock, cmd, cmd_len, 0);
recv (sock, buf, 256, 0);
...
close (sock);
```

Accès direct au trames de la HCI

- pour implémenter une recherche de périphériques peronnalisée
- pour tester une pile bluetooth personnalisée
- protocole interne H:4 (rappel)

Programmation SDP : Déclarations

```
static sdp_session_t *session;
static sdp_record_t *record;

#define TIMESERVER_UUID "2c3ef0c8-4367-4369-b1df-b7315e76332a"
static uint128_t* timeserver_uuid_data = NULL;
static uuid_t timeserver_uuid;

static uint16_t timeserver_psm = 0xccdd;
uint16_t version = 0x0100;
```

Programmation SDP : Enregistrement

```
sdp_list_t *svclass, *pfseq, *apseq, *root, *aproto;
uuid_t root_uuid, l2cap;
sdp_profile_desc_t profile[1];
sdp_list_t *proto;

session = sdp_connect(BDADDR_ANY, BDADDR_LOCAL, 0);
record = sdp_record_alloc();
sdp_uuid16_create(&root_uuid, PUBLIC_BROWSE_GROUP);
root = sdp_list_append(NULL, &root_uuid);
sdp_set_browse_groups(record, root);

sdp_uuid16_create(&l2cap, L2CAP_UUID);
proto = sdp_list_append(NULL, &l2cap);
proto = sdp_list_append(proto, sdp_data_alloc(SDP_UINT16, &timeserver_psm));
apseq = sdp_list_append(NULL, proto);
aproto  = sdp_list_append(NULL, apseq);
sdp_set_access_protos(record, aproto);

svclass = sdp_list_append(NULL, &timeserver_uuid);
sdp_set_service_classes(record, svclass);

sdp_uuid128_create(&profile[0].uuid, timeserver_uuid_data);
profile[0].version = 0x0100;
pfseq = sdp_list_append(NULL, &profile[0]);
sdp_set_profile_descs(record, pfseq);

sdp_set_info_attr(record, "Time server example", NULL, NULL);

sdp_record_register(session, record, 0);
```

Recherche SDP :

```
aid 0x0000 (SrvRecHndl)
        uint 0x805d898
aid 0x0001 (SrvClassIDList)
        uuid-128 0x2c3ef0c8
aid 0x0004 (ProtocolDescList)
        uuid-16 0x0100 (L2CAP) uint 0xcc33
aid 0x0005 (BrwGrpList)
        uuid-16 0x1002 (PubBrwsGrp)
aid 0x0009 (BTProfileDescList)
        uuid-128 0x2c3ef0c8 uint 0x100
aid 0x0100 (SrvName)
        str "Time server example"
```

Programmation L2CAP

Application serveur

```
int sock, client;
int addrlen;
struct sockaddr_l2 addr;
struct hci_dev_info di;

addr.l2_family = AF_BLUETOOTH;
bacpy(&addr.l2_bdaddr, BDADDR_ANY);
addr.l2_psm    = htobs(PSM);

sock = socket(AF_BLUETOOTH, SOCK_SEQPACKET, BTPROTO_L2CAP);
bind(sock, (struct sockaddr *)&addr, sizeof(addr));
listen(sock, 10);
addrlen = sizeof(addr);
client = accept (sock, (struct sockaddr *)&addr, &addrlen);
/* read & write */
close (sock);
```

Programmation L2CAP

Application client

```
int serveur;
struct sockaddr_l2 laddr, raddr;
struct hci_dev_info di;
struct l2cap_options opts;
int optsize;

hci_devinfo(0, &di);
laddr.l2_family = AF_BLUETOOTH;
laddr.l2_bdaddr = di.bdaddr;

optsize = sizeof(opts);
getsockopt(s, SOL_L2CAP, L2CAP_OPTIONS, &opts, &opt);
opts.omtu = OMTU; opts.imtu = IMTU;
setsockopt(s, SOL_L2CAP, L2CAP_OPTIONS, &opts, opt);

raddr.L2_family = AF_BLUETOOTH;
str2ba(BTADDR_SERVEUR,&raddr.l2_bdaddr);
raddr.l2_psm = htobs(PSM);

serveur = socket(AF_BLUETOOTH, SOCK_SEQPACKET, BTPROTO_L2CAP);
bind(serveur, (struct sockaddr *)&laddr, sizeof(laddr));
connect(serveur, (struct sockaddr *)&raddr, sizeof(raddr));
/* read & write */
close (serveur);
```

```
root@tclinux:/home/xavier/devel_c/Bluetooth/timeserver# hcitool inq
Inquiring ...
< 01 01 04 05 33 8B 9E 08 0A
> 04 0F 04 00 01 01 04
> 04 02 0F 01 B7 4C 1F 96 A0 00 01 00 00 00 00 00 5B 3D
> 04 01 01 00
        00:A0:96:1F:4C:B7       clock offset: 0x3d5b    class: 0x000000
root@tclinux:/home/xavier/devel_c/Bluetooth/timeserver# ./timeserverclient 00:A0:96:1F:4C:B7
< 01 05 04 0D B7 4C 1F 96 A0 00 18 CC 01 00 5B BD 01
> 04 0F 04 00 01 05 04
> 04 03 0B 00 28 00 B7 4C 1F 96 A0 00 01 00
< 02 28 20 0C 00 08 00 01 00 02 01 04 00 33 CC 40 00
> 04 13 05 01 28 00 01 00
> 04 1B 03 28 00 05
> 02 28 20 10 00 0C 00 01 00 03 01 08 00 50 00 40 00 00 00 00 00
< 02 28 20 0C 00 08 00 01 00 04 02 04 00 50 00 00 00
> 04 13 05 01 28 00 01 00
> 02 28 20 11 00 1A 00 01 00 05 02 0A 00 40 00 00 00 00 00 01 02 40
> 02 28 10 0D 00 00 04 01 08 00 40 00 00 00 01 02 40 00
< 02 28 20 0E 00 0A 00 01 00 05 01 06 00 50 00 00 00 00 00
< 02 28 20 05 00 01 00 50 00 03
Sent: 03
> 04 13 05 01 28 00 01 00
> 04 13 05 01 28 00 01 00
> 02 28 20 09 00 05 00 40 00 02 11 0C 0D 2C
Received: 02:11:0c:0d:2c
Checksum OK
It's: 17:12:13
< 02 28 20 0C 00 08 00 01 00 06 03 04 00 50 00 40 00
End
> 04 13 05 01 28 00 01 00
> 02 28 20 0C 00 08 00 01 00 07 03 04 00 50 00 40 00
< 01 06 04 03 28 00 13
> 04 0F 04 00 01 06 04
> 04 05 04 00 28 00 16
```

Programmation RFCOMM

Application serveur

```
int sock, client;
struct sockaddr_rc addr;
int alen;

sock = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
addr.rc_family = AF_BLUETOOTH;
bacpy(&addr.rc_bdaddr, BDADDR_ANY);
addr.rc_channel = htobs(CHANNEL);
bind(sock, (struct sockaddr *)&addr, sizeof(addr));
listen(sock,QUEUE);
alen = sizeof(addr);
client = accept(sock, (struct sockaddr *)&addr, &alen);
/* read & write */
close (client);
close (sock);
```

Programmation RFCOMM

Application client

```
int serveur;
struct sockaddr_rc laddr, raddr;
struct hci_dev_info di;

hci_devinfo(0, &di);
laddr.rc_family = AF_BLUETOOTH;
laddr.rc_bdaddr = di.bdaddr;
laddr.rc_channel = 0;

raddr.rc_family = AF_BLUETOOTH;
str2ba(BTADDR_SERVEUR,&raddr.rc_bdaddr);
raddr.rc_channel = htobs(CHANNEL);

serveur = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
bind(serveur, (struct sockaddr *)&laddr, sizeof(laddr));
connect(serveur, (struct sockaddr *)&raddr, sizeof(raddr));
/* read & write */
close (serveur);
```

Pourquoi utiliser linux comme station de développement ?

- Libre !
- Outil hcidump pour analyse des trames
- Liberté d'accès à tous les niveaux de la pile bluetooth
- Interface simple
- Nombreux exemples sous forme de code source
- Outils efficaces et étudiables
- Liberté offerte par les différentes interfaces

En revanche :

- interface (Api) relativement compliquée et peu documentée
- peu utilisé par les utilisateurs lambda
- actuellement dédié :
  - aux applications embarquées
  - aux universitaires
  - en période de recherche et développement, prototypage en entreprise

Support Bluetooth via pile intégrée à partir de XP SP2
- API pour HCI, RFCOMM
- API socket

Pile Widcomm largement répandue
- SDK confidentiel
  - Coût à l'achat ~1500$
  - Pas de royalties ensuite
- API pour SDP, L2CAP, RFCOMM, Profils
- API claire, simple et documentée

Diverses autre initiatives (FreeBT, ...)

Nokia:
http://forum.nokia.com
SDK pour J2ME et Symbian (Series40, Series 60, Series 80 et Series 90)
Interface de gestion de périphériques interfaçable
API Bluetooth pour L2CAP, RFCOMM, SDP
Langage C++ ou Java selon téléphones

SonyEriccson
http://developer.sonyericsson.com/
SDK pour J2ME, Mophun, Symbian UIQ2.0 et UIQ2.1
Interface de gestion de périphériques interfaçable
API Bluetooth pour L2CAP, RFCOMM, SDP
Langage C++ ou Java

Symbian
http://www.symbian.com/
Nombreux outils supplémentaires, SDKs pour SymbianOS

Création d'un Active Object
Envoi de commandes - Affectation de statuts
Gestion des évènements et lecture : méthode RunL de l'AO

```
enum TBTEngineState
        {
        ENotConnected, EConnectFailed,
        EConnecting, ERetrievingInfoR, ERetrievingInfoW, EIconOff, EReady,
        ESyncTime, ESyncTimeW, ESyncTimeE,
        EMonitor, EMonitorW, EMonitorR, EMonitorE,
        ERetrieveTimedOut, EDisconnecting
        };
class CBTEngine : public Cactive
{
CBTEngine(CNotifier*, CModel*);
void ConstructL();
void RunL();

public:
static CFirstP800AppBTEngine* NewL(CNotifier*, CModel*);
~CfirstP800AppBTEngine();
...
};
```

Initialisation

```
CactiveScheduler::Add(this);

m_SocketServ.Connect();
TProtocolDesc pInfo;
m_SocketServ.FindProtocol(_L("L2CAP"), pInfo);

TUid uid;
uid.iUid = m_PSM;
m_BTService.SetUid(uid);
m_BTService.SetChannelID(m_PSM);
m_BTService.SetProtocolID(KSolBtL2CAP);
m_BTService.SetAuthentication(EFalse);
m_BTService.SetAuthorisation(EFalse);
m_BTService.SetEncryption(EFalse);

RBTMan btman;
User::LeaveIfError(btman.Connect());

RBTSecuritySettings secset;
User::LeaveIfError(secset.Open(btman));

TRequestStatus requestStatus;
secset.RegisterService(m_BTService, requestStatus);
User::WaitForRequest(requestStatus);

secset.Close();
btman.Close();
```

UI de recherche & connexion

```
CQBTUISelectDialog* dialog
      = new (ELeave) CQBTUISelectDialog(m_Addr,
                                        m_Name,
                                        m_DevClass,
                                        CQBTUISelectDialog::EQBTDeviceFilterAll,
                                        serviceFilter);
dialog->LaunchSingleSelectDialogLD();

m_RemoteSockAddr.SetPort(m_PSM);
m_RemoteSockAddr.SetBTAddr(m_Addr);
m_BTSocket.Open(m_SocketServ, KBTAddrFamily, KSockSeqPacket, KL2CAP);

iStatus = KRequestPending;
m_BTSocket.Connect(m_RemoteSockAddr, iStatus);
SetActive();
```

Déconnexion

```
TLMDisconnectACLBuf aclConnBuf;
aclConnBuf().iDevAddr = m_Addr;
aclConnBuf().iReason = 0x13;
TRequestStatus stat;
m_BTSocket.Ioctl(KLMDisconnectACLIoctl, stat, &aclConnBuf, KSolBtLM);
User::WaitForRequest(stat);
```

Envoi de données :

```
void CBTEngine::Send()
{
	TPtr8 aBuffer = m_OutBuffer->Des();
	aBuffer.SetLength(3);
	aBuffer[0] = 0x08;
	aBuffer[1] = icon_option_off | icon_sms;
	aBuffer[2] = 0x00;

	iStatus = KRequestPending;
	m_BTSocket.Write(*m_OutBuffer, iStatus);
	SetActive();
}
```

Lecture de données :

```
void CBTEngine::Recv()
{
	m_PtrInBuffer.Set(m_InBuffer->Des());
	if (!IsActive())
		{
		iStatus = KRequestPending;
		m_BTSocket.Read(m_PtrInBuffer, iStatus);
		SetActive();
		}
}
```

Changements de statuts de l'Active object

```
void CBTEngine::RunL()
{
	if (iStatus == KErrDisconnected) {
		m_Model->Reset();
		Cancel();
	if (GetState() != ENotConnected) DoDisconnect();
		m_Notifier->NotifyError(R_FP800A_INFO_DISCONNECTED,
							iStatus.Int());
	}
	switch (m_State) {
		case EConnecting:
			if (iStatus == KErrNone) {
				m_Notifier->NotifyProgress(1);
				Recv();
			} else {
				m_State = EConnectFailed;
				m_Notifier->NotifyError(R_FP800A_INFO_CONNECT_FAILED,
								iStatus.Int());
			}
			break;
		case EDisconnecting:
			m_State = ENotConnected;
			break;

...
```

Libération des ressources

```
RBTMan btman;
TInt err = btman.Connect();

RBTSecuritySettings secset;
err = secset.Open(btman);

TRequestStatus requestStatus;
secset.UnregisterService(m_BTService, requestStatus);
User::WaitForRequest(requestStatus);

secset.Close();
btman.Close();

m_BTSocket.Close();
m_SocketServ.Close();
```

PDA et téléphones

http://www.palmsource.com

Interface de gestion de périphériques interfaçable

API Bluetooth pour L2CAP, RFCOMM, SDP

Langage C

Envoi de commandes et trames

Evènements et lecture via callbacks

Initialisation
- BtLibOpen
- BtLibRegisterManagementNotification

UI de recherche
- BtLibDiscoverSingleDeviceLien ACL
- BtLibLinkConnect

Lien L2CAP
- BtLibSocketCreate (assignation du callback de gestion de la socket)
- BtLibSocketConnect

Lecture/Ecriture
- Appel callback et MemMove depuis event_data
- BtLibSocketSend

Déconnexion
- BtLibSocketClose
- BtLibLinkDisconnect

Clotûre
- BtLibUnregisterManagementNotification
- BtLibClose

Java APIs for Bluetooth Wireless Tehnology

Standardisée pour J2ME

Composant optionnel de la CLDC
(Connected Limited Device Configuration)
Classes et apis de base

MIDP
(Mobile Information Device Profile)
Classes et apis de plus haut niveau

JSR (Java Specification Request) 82

Pensé pour les périphériques mobiles à faibles ressources

Utilise une pile bluetooth existante

Accès aux couches SDP, L2CAP, RFCOMM, OBEX, Link Manager

Points d'entrée pour profils GAP, SDAP, SPP, GOEP

Configuration via un BCC (Bluetooth Control Center)

Packages:

- javax.bluetooth
- javax.obex

dépendent de javax.microedition.io

Recherche de périphériques

```
LocalDevice local = LocalDevice.getLocalDevice();
DiscoveryAgent agent = loca.getDiscoveryAgent;
agent.startInquiry(DiscoveryAgent.GIAC, this);
Implémentation des méthodes de l'interface DiscoveryListener
        public void deviceDiscovered (RemoteDevice dev, DeviceClass cod);
        public void inquiryCompleted (nit type);
```

Connexion

```
L2CAPConnection conn;
String url = "btl2cap://00a0961f3c64:52275; authenticate=false;"
                    + "encrypt=false;master=false;"
                    + "receiveMTU=672;transmitMTU=672";
try {
        conn = Connector.open(url);
} catch (BluetoothConnectionException e) {
        ...
}
```

Envoi et réception

```
conn.send("TEST".getBytes());

byte[] ibuf = new byte[conn.getReceiveMTU()];
int bytesRead;
bytesRead = conn.receive(ibuf);
```

Déconnexion

```
conn.close();
```

Références :

- Bluetooth Application Developer's Guide
- Symbian OS C++ for Mobile Phones
- Wireless Java: Developping with J2ME
- Bluetooth Application Programming with the JAVA APIs

- http://www.bluetooth.org/
- http://www.bluez.org/ (et ses mailing listes user et devel)
- http://www.holtmann.org/
- http://msdn.microsoft.com/library/default.asp?url=
  /library/en-us/bluetooth/bluetooth/bluetooth_start_page.asp
- http://forum.nokia.com/
- http://developer.sonyericsson.com/
- http://www.symbian.com/