

Devices

Bluetooth Programming

- Refer :
<http://people.csail.mit.edu/albert/bluez-intro/>
- Book: <http://www.btessentials.com>
- Similar to TCP/IP programming

Basics

- The different parts of network programming can be separated into several components
- Choosing a device with which to communicate
- Figuring out how to communicate with it
- Making an outgoing connection
- Accepting an incoming connection
- Sending data
- Receiving data

- Bluetooth devices have 48 bit address. Like MAC address

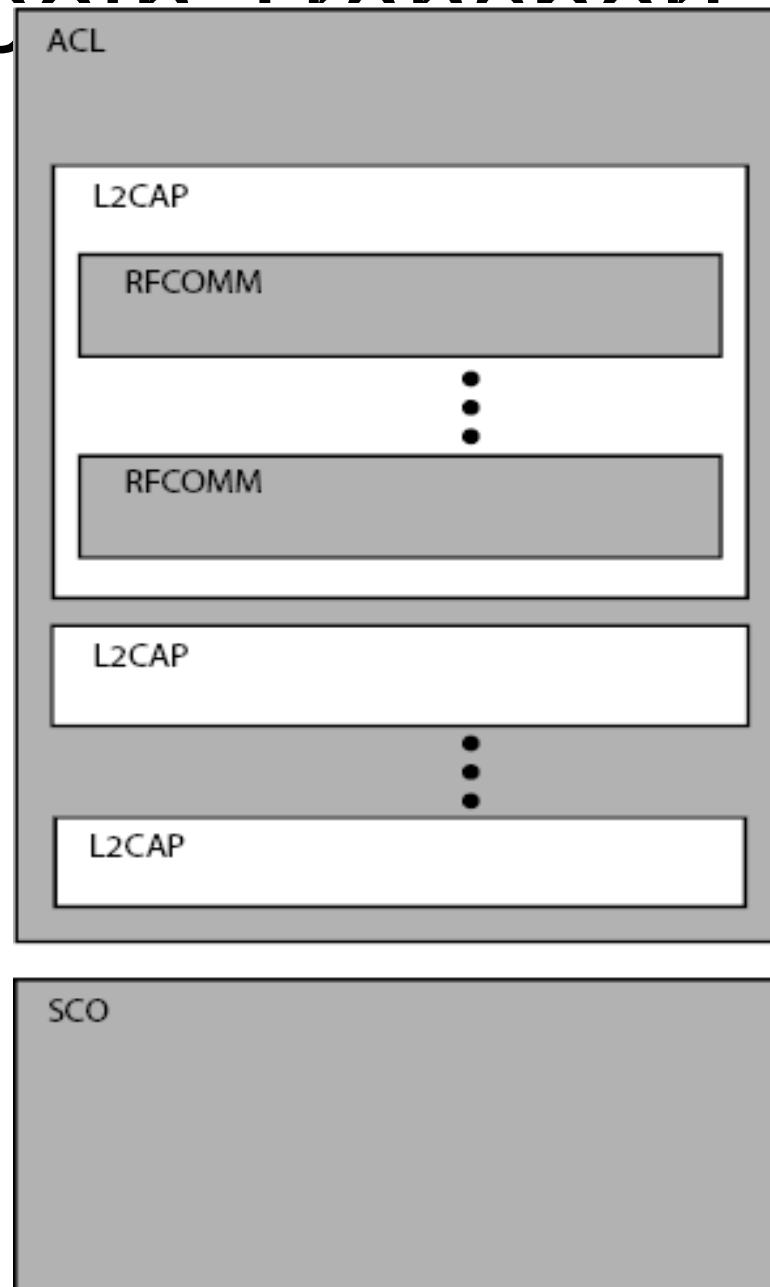
How to communicate

Requirements	Internet	Bluetooth	Ports
Reliable, Stream-based	TCP	RFCOMM	30
Reliable, Datagram	TCP	RFCOMM or L2CAP	
Best effort, datagram	UDP	L2CAP	Odd # 1-32765

- Small number of port prompt for Service Discovery Protocol (SDP) in RFCOMM.
- Universally Unique Identifier (UUID) are assigned to different services in the standard.

- Radio Frequency Communications (RFCOMM) protocol is a reliable streams-based protocol. It provides roughly the same service and reliability guarantees as TCP.
- The Logical Link Control and Adaption Protocol (L2CAP) is a packet-based protocol that can be configured with varying levels of reliability.
 - L2CAP can be compared with UDP (best effort)

Bluetooth Transport Protocol



Programming

- Python is considered very good. (interpreted)
(though I haven't tried programming using python)
- C programs using Bluez library.
 - Fast and direct interact device drivers. No middle layer

Scan Program

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>
int main(int argc, char **argv)
{
    inquiry_info *ii = NULL;
    int max_rsp, num_rsp;
    int dev_id, sock, len, flags;
    int i;
    char addr[19] = { 0 };
    char name[248] = { 0 };
    dev_id = hci_get_route(NULL);
    sock = hci_open_dev( dev_id );
    if (dev_id < 0 || sock < 0) {
        perror("opening socket");
        exit(1);
    }
    len = 8;
    max_rsp = 255;
    flags = IREQ_CACHE_FLUSH;
    ii = (inquiry_info*)malloc(max_rsp * sizeof(inquiry_info));

    num_rsp = hci_inquiry(dev_id, len, max_rsp, NULL, &ii,
flags);
    if( num_rsp < 0 ) perror("hci_inquiry");
    for (i = 0; i < num_rsp; i++) {
        ba2str(&(ii+i)->bdaddr, addr);
        memset(name, 0, sizeof(name));
        if (hci_read_remote_name(sock, &(ii+i)->bdaddr, sizeof(name),
name, 0) < 0)
            strcpy(name, "[unknown]");
        printf("%s %s\n", addr, name);
    }
    free( ii );
    close( sock );
    return 0;
}
```

Select Bluetooth interface

Scan for 8*len time
max_rsp = # of devices
ii stores devices info

Server

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>
int main(int argc, char **argv)
{
    struct sockaddr_rc loc_addr = { 0 }, rem_addr = { 0 };
    char buf[1024] = { 0 };
    int s, client, bytes_read;
    socklen_t opt = sizeof(rem_addr);
    // allocate socket
    s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
    // bind socket to port 1 of the first available
    // local bluetooth adapter
    loc_addr.rc_family = AF_BLUETOOTH;
    loc_addr.rc_bdaddr = *BDADDR_ANY;
    loc_addr.rc_channel = (uint8_t) 1;
    bind(s, (struct sockaddr *)&loc_addr, sizeof(loc_addr));
    // put socket into listening mode
    listen(s, 1);
    // accept one connection
    client = accept(s, (struct sockaddr *)&rem_addr, &opt);
    ba2str( &rem_addr.rc_bdaddr, buf );
    fprintf(stderr, "accepted connection from %s\n", buf);
    memset(buf, 0, sizeof(buf));
    // read data from the client
    bytes_read = read(client, buf, sizeof(buf));
    if( bytes_read > 0 ) {
        printf("received [%s]\n", buf);
    }
    // close connection
    close(client);
    close(s);
    return 0;
}
```

Client

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>
int main(int argc, char **argv)
{
    struct sockaddr_rc addr = { 0 };
    int s, status;
    char dest[18] = "01:23:45:67:89:AB";
    // allocate a socket
    s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
    // set the connection parameters (who to connect to)
    addr.rc_family = AF_BLUETOOTH;
    addr.rc_channel = (uint8_t) 1;
    str2ba( dest, &addr.rc_bdaddr );
    // connect to server
    status = connect(s, (struct sockaddr *)&addr, sizeof(addr));
    // send a message
    if( status == 0 ) {
        status = write(s, "hello!", 6);
    }
    if( status < 0 ) perror("uh oh");
    close(s);
    return 0;
}
```

Basics about N8x0

- Runs debian variant called Maemo
- Has a very rich repository collection.
- Most of the source code is available

programming N8x0

- Nokia's N series tablets run on ARM processor
- Need for ARM compiled code
- Need for a cross-compiler (traditional)
- Nokia uses virtual machine SDK – Scratchbox
- Scratchbox -

SCRATCHBOX

- Features
 - Black box environment
 - Runs over qemu (QEMU is a generic and open source machine emulator and visualizer)
 - Tedious installations (Ubuntu host preferred)
 - GCC
 - N8X0 emulator with screen visualization
 - Apt-get !

Friend finder code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>

int main(int argc, char **argv)
{
    inquiry_info *ii = NULL;
    int max_rsp, num_rsp;
    int dev_id=0, sock, len, flags=0;
    int i, max_count;
    char addr[20][19] = { 0 };
    char name[20][25] = { 0 };
    char addr_tmp[19];
    char msg[50];
    int choice;
    FILE *fout;
    int k;
    sock = hci_open_dev( dev_id );
    if (dev_id < 0 || sock < 0) {
        perror("opening socket");
        exit(1);
    }
    if ((fout=fopen(".bluetoothfriends","r"))==NULL)
    {
        printf("Error: cannot open .bluetoothfriends \n");
        exit(1);
    }
    i=0;
    while (!feof(fout) && i < 25)
    {
        fscanf(fout,"%s %s\n",addr[i],name[i]);
        // printf("%s %s\n",addr[i],name[i]);
        i++;
    }
```

```

}
max_count=i;

while(1)
{
    len = 10;
    max_rsp = 10;
    flags = IREQ_CACHE_FLUSH;
    ii = (inquiry_info*)malloc(max_rsp * sizeof(inquiry_info));
    //printf("DEBUG: before hci_inquiry . \n");
    num_rsp = hci_inquiry(dev_id, len, max_rsp, NULL, &ii, flags);
    //printf("DEBUG: After hci_inquiry . number of device found :%d\n",num_rsp);
    if( num_rsp < 0 )
    {
        perror("hci_inquiry");
        sleep(60);
        continue;
    }

    for (k=0;k<num_rsp;k++)
    {
        ba2str(&(ii+k)->bdaddr, addr_tmp);
        //printf("Address found %s\n",addr_tmp);
        i=0;
        for(i=0;i<max_count;i++)
        {
            if(strcmp(addr_tmp,addr[i])==0)
            {
                printf("Attention %s is near\n",name[i]);
                strcpy(msg,"flite -t \"Attention !!");
                strcat(msg,name[i]);
                strcat(msg,"is near\n");
                system(msg);
                sleep(3);
            }
        }
    }
    sleep(60);
}
free( ii );
close( sock );
return 0;
}

```

Message Digest

- `void md5_str(char* password, unsigned char* str_digest)`
- `{`
-
- `unsigned char digest[16];`
- `memset(digest, '\0', 16);`
- `unsigned char temp[4];`
- `unsigned int len = strlen (password);`
- `MD5_CTX context;`
- `int i;`
- `MD5_Init(&context);`
- `MD5_Update (&context, password, len);`
- `MD5_Final(digest, &context);`
-

- `//printf("\n:");`
- `for (i = 0; i < 16; i++)`
- `{`
- `memset(temp,'\0',4);`
- `//printf ("%02x", digest[i]);`
- `sprintf (temp,"%02x", digest[i]);`
- `strcat(str_digest,temp);`
- `}`
- `//printf("\n");`
- `return ;`
-
- `} //end of md5_str()`
-

Scanning Script

```
while [ true ]  
do  
  `echo --Date-- >> /media/mmc1/data.txt`  
  `date >> /media/mmc1/data.txt`  
  `echo --BlueTooth-- >> /media/mmc1/data.txt`  
  `btsearch -t >> /media/mmc1/data.txt`  
  `echo --WLAN-- >> /media/mmc1/data.txt`  
  `iwlist wlan0 scan >> /media/mmc1/data.txt`  
  sleep 60  
done
```