Giao tiếp Bluetooth với JavaME

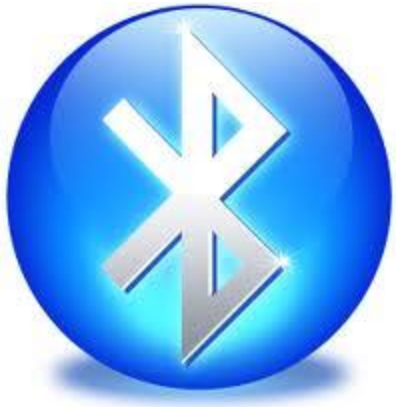
GV: ThS. Phan Nguyệt Minh

minhpn@uit.edu.vn

http://courses.uit.edu.vn

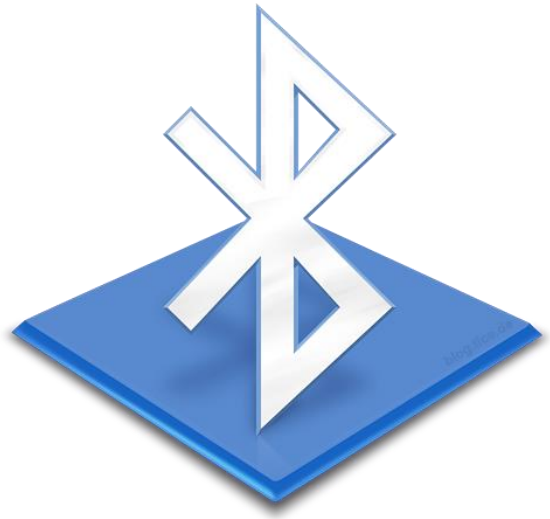
Bluetooth

- Bluetooth là một công nghệ cho phép truyền thông giữa các thiết bị với nhau mà không cần cáp và dây dẫn. Nói cách khác, nó là một chuẩn điện tử.



Bluetooth

- Bluetooth được xây dựng bởi Bluetooth Special Interest Group (SIG).
- Bluetooth được đặt theo tên thân mật của một vị vua Đan Mạch: Harald Bluetooth.



Bluetooth

- Bluetooth là chuẩn kết nối không dây tầm ngắn, thiết kế cho các kết nối thiết bị cá nhân hay mạng cục bộ nhỏ, trong phạm vi băng tần từ 2.4 đến 2.485 GHz.
- Hoạt động trên 79 tần số đơn lẻ.
- Được thiết kế kết nối tầm thấp với 3 lớp khác nhau nhằm có thể cơ động truyền sóng đi xa nhất đến mức có thể. Thông thường, các loại di động hiện tại dùng Bluetooth ở lớp thứ 2, với cường độ 2.5 miliWatt (*mW*) và phạm vi chỉ có 12m trở lại.

Các thế hệ Bluetooth

- Bluetooth 1.0
- Bluetooth 1.1
- Bluetooth 1.2
- Bluetooth 2.0+ERD
- Bluetooth 2.1+ERD
- Bluetooth 3.0+HS
- Bluetooth 4.0
- Bluetooth 4.1

Ưu nhược điểm của Bluetooth

Ưu điểm:

-Truyền dữ liệu miễn phí.

-Thiết lập kết nối dễ dàng mà không cần Ascess Point.

-Sử dụng được bất cứ ở đâu.

-Sử dụng cùng một chuẩn giao thức nên mọi thiết bị bluetooth có thể kết nối với nhau.

Ưu nhược điểm của Bluetooth

- *Nhược điểm:*

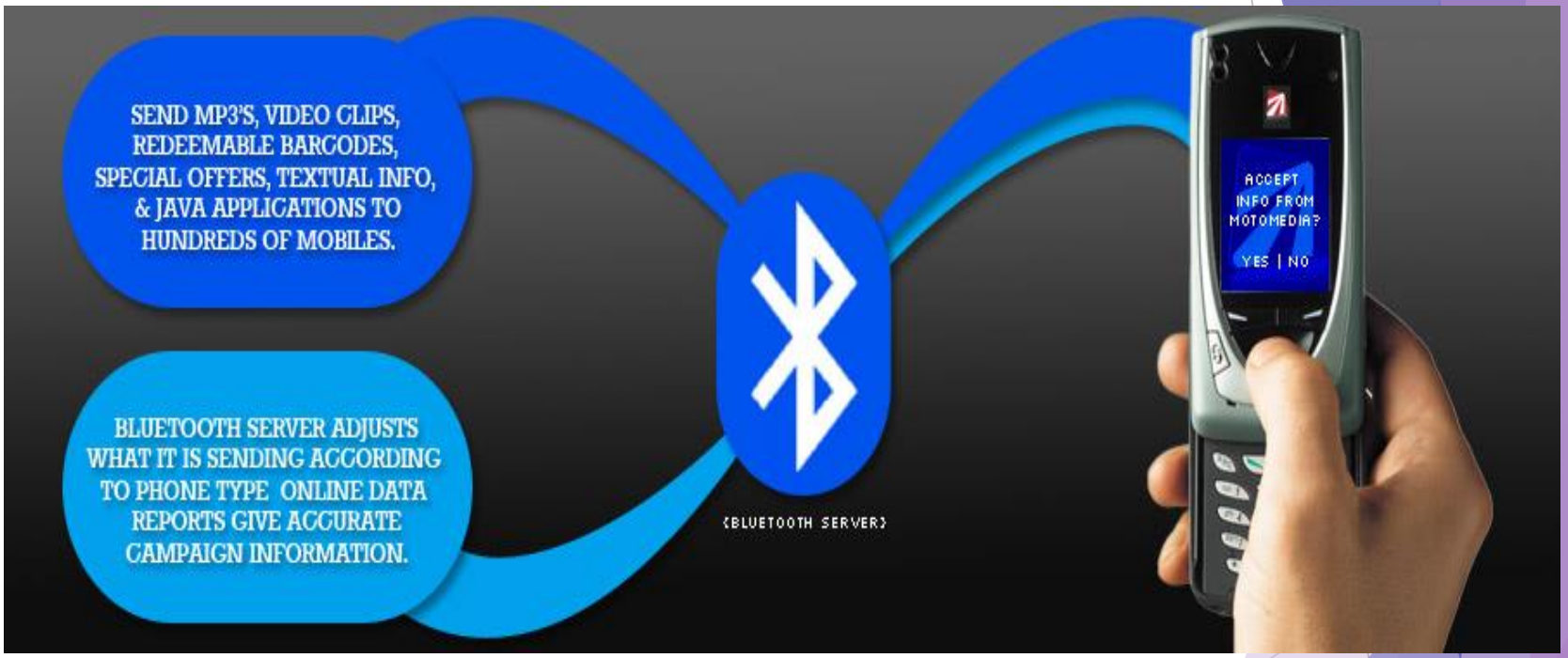
-Khoảng cách kết nối ngắn.

-Bảo mật thấp.

-Tốc độ truyền không cao.

-Bị nhiễu bởi một số thiết bị sử dụng sóng Radio khác.

Bluetooth với J2ME



Bluetooth với J2ME

- *Có 3 loại giao tiếp trong công nghệ Bluetooth:*

  -OBEX: Viết tắt bởi “Object Exchange” giao thức giao tiếp được sử dụng để trao đổi dữ liệu vật lý như các tập tin, hình ảnh, và kể cả các định dạng nhị phân.

  -L2CAP: Viết tắt bởi “Logical Link Control and Adaptation Protocol” được sử dụng để gửi các gói dữ liệu giữa máy chủ và máy khách.

  -RFCOMM: Viết tắt bởi ” Radio Frequency Communication” được sử dụng cho luồng dữ liệu đơn giản.

Mô hình hoạt động của ứng dụng Client và Server trong J2ME

Bước	Server	Client
1	Quản lý thiết bị, thiết lập chế độ nhận biết	Quản lý thiết bị, thiết lập chế độ nhận biết
2	Mở kết nối Bluetooth, kết nối dịch vụ	Nhận biết các thiết bị Bluetooth xung quanh
3	Chờ Client kết nối	Nhận biết dịch vụ thiết bị Bluetooth đã nhận biết ở trên
4		Nếu tìm thấy dịch vụ cần thiết, tiến hành kết nối tới dịch vụ đó
5	Mở các dòng nhập xuất dữ liệu để trao đổi thông tin	Mở các dòng nhập xuất dữ liệu để trao đổi thông tin

Tạo Server

-*Tìm kiếm các thiết bị khác*:

Sử dụng 2 hàm:

+LocalDevice.getLocalDevice ()

+setDiscoverable(DiscoveryAgent.GIAC)

-*Mở kết nối Bluetooth*

Xây dựng một chuỗi URL Bluetooth:

URL = “btspp://localhost:” + UUID + “;name=rfcommtest;authorize=true”;

Chuỗi URL Bluetooth

Hàm Connector.open (URL)

StreamConnectionNotifier

Hàm acceptAndOpen()

StreamConnection

Hàm openOutputStream() hoặc openInputStream()

```
 m_strUrl= "btspp://localhost:" + RFCOMM_UUID + ";
name=rfcommtest;authorize=true";
// m_StrmConn = BTFACADE.waitForClient(SERVICE_NBR);
Try
{
m_LclDevice = LocalDevice.getLocalDevice();
m_LclDevice.setDiscoverable(DiscoveryAgent.GIAC);
m_StrmNotf =
    (StreamConnectionNotifier)Connector.open(m_strUrl);
//Now it will start waiting for the client connection
m_StrmConn = m_StrmNotf.acceptAndOpen();
m_bInitServer = true;
m_Output = m_StrmConn.openOutputStream();
m_Input  = m_StrmConn.openInputStream();
}
```

Tạo Client

- Trước tiên, ta cần tìm kiếm các thiết bị Bluetooth xung quanh.

```
public void SearchAvailDevices()
{
try
{
//First, get the local device and obtain the discovery agent.
  m_LclDevice = LocalDevice.getLocalDevice();
 m_DscrAgent=  m_LclDevice.getDiscoveryAgent();
 m_DscrAgent.startInquiry(DiscoveryAgent.GIAC,this);
 }
catch (BluetoothStateException ex)
{
System.out.println("Problem in searching the Bluetooth devices");
 ex.printStackTrace();
 } }
```

Tạo Client

Để tạo một Client, ta cần thực thi một giao diện DiscoveryListener.

Giao diện này có các hàm chính sau:

- void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
- void servicesDiscovered(int transID, ServiceRecord[] records)
- void serviceSearchCompleted(int transID, int respCode)
- Connector.open(m_strUrl)

Gửi và nhận dữ liệu

- Khi client và server bắt đầu làm việc, quá trình giao tiếp được thực hiện thông qua 2 hàm: sendMessages() và receiveMessages().

```
public void sendMessages(String v_strData) {
if((m_bInitClient) || (m_bInitServer) )
 {
    try
         {
           m_Output.write(v_strData.length());
           m_Output.write(v_strData.getBytes());
         }
   catch (IOException ex)
 {
   ex.printStackTrace();
  }    } }
```

```
public String  recieveMessages()
 {
byte[] data = null;
    try
  {
 int length = m_Input.read();
      data       = new
   byte[length];
      length     = 0;
     while (length !=
   data.length)
     {
 int ch = m_Input.read(data,
   length, data.length -
   length);
         if (ch == -1)
         {
      throw new
   IOException("Can't read
   data");
       }
   length += ch;
    }
  }
catch (IOException e)
  {
System.err.println(e);
   }
 return new String(data);
}
```

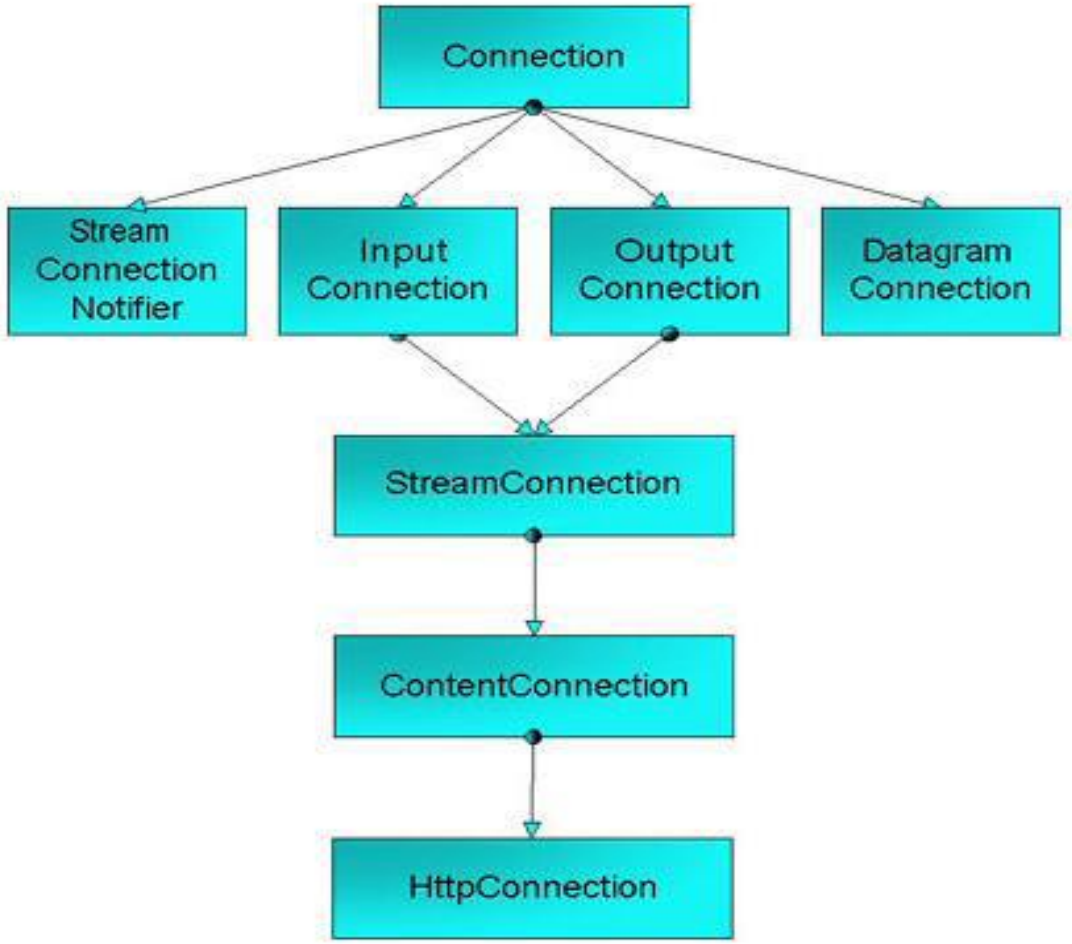
Kết nối mạng với J2ME

- Nội dung:
  - Generic Connection Framework
  - Kết nối HTTP
  - Kết nối Socket
  - Kết nối Datagram

Generic Connection Framework

- GCF là một tập hợp các lớp và giao diện được thiết kế nhằm tạo thuận tiện cho việc truy xuất đến các hệ thống lưu trữ và kết nối mạng.
- Mục tiêu của GCF không phải là tạo ra một tập các lớp mới hoàn toàn mà nó cung cấp một tập con của J2SE một cách có chọn lọc. Tập con này được giới hạn và tối ưu để phù hợp với những ràng buộc và khác biệt của những thiết bị di động.

Cây phân cấp connection



Cây phân cấp Connection - Khung kết nối chung

Giao diện *GCF*	Chú thích
Connection	Là kiểu kết nối cơ bản trong GCF. Và là cha của các kiểu kết nối còn lại
ContentConnection	Quản lý một kết nối (http,..) Cung cấp các phương thức cơ bản: độ dài nội dung, mã hóa và kiểu của nội dung
DatagramConnection	Quản lý kết nối Datagram
InputConnection	Quản lý kết nối vào
OutputConnection	Quản lý kết nối ra
StreamConnection	Quản lý một dòng. Là sự tổng hợp của InputConnection và OutputConnection
StreamConnectionNotifier	Lắng nghe trên một cổng và tạo ra một StreamConnection ngay sau khi nhận thấy có hoạt động trên cổng.

- Trong *GCF* thì *URLs* có vai trò quan trọng, nó được dùng trong Internet. *URLs* đưa ra vị trí , cách thức truy cập đến nguồn tài nguyên

- Dạng của một *URLs* trong *GCF*

protocol://host:port/url-path

Trong đó

protocol: Là phương thức xử lý hay giao thức (http, ftp…)

host: Là tên hoặc địa chỉ IP của máy tính nơi đặt tài nguyên

port: Cổng hoạt động của giao thức(tùy chọn), phụ thuộc vào scheme

url-path: Là đường dẫn đến tài nguyên

Một số protocol và loại connection

Protocol	Kết nối	Kiểu kết nối GCF	Định nghĩa bởi
bt12cap	Bluetooth	L2CAPConnection	JSR82
datagram	Datagram	DatagramConnection	Mọi CLDC , CDC, MIDP
File	File Access	FileConnection, InputConnection	JSR75

http	HyperText TransferProtocol	HttpConnection	MIDP 1.0, MIDP 2.0, Foundation Profile, J2SE (JSR 197). Support is required
https	Secure HTTP	HttpsConnection	MIDP 2.0
comm	Serial I/O	ComConnection	MIDP 2.0

sms mms cbs	Short messaging service Multimedia Messaging Service Cell Broadcast SMS	MessageConnection	JSR 120, JSR 205
comm	Serial I/O	ComConnection	MIDP 2.0
Socket ServerSocket	Socket	SocketConnection, ServerSocketConnection	JSR118 (MIDP 2.0)
datagram	UDP Datagram	UDPDatagramConnection	JSR118 (MIDP 2.0)

Kết nối HTTP

HTTP là giao thức duy nhất chắc chắn được hỗ trợ bởi MIDP 1.0. Chúng ta có thể giao tiếp với máy chủ hay bất kỳ thiết bị từ xa nào có hỗ trợ giao thức này nhờ vào lớp HttpConnection. Lớp Connector cung cấp cho người dùng bảy phương thức để tạo kết nối tới máy chủ .

Phương thức	Mô tả
static Connection open(String name)	Tạo một kết nối có chế độ READ_WRITE
static Connection open(String name, int mode)	Tạo một kết nối với chế độ được chỉ định
static Connection open(String name, int mode, boolean timeouts)	Tạo một kết nối với chế độ được chỉ định, thêm ngoại lệ time out

static InputStream openInputStream(String name)	Tạo kết nối luồng nhập
static OutputStream openOutputStream(String name)	Tạo kết nối luồng xuất
static DataInputStream openDataInputStream(String name)	Tạo kết nối luồng nhập kiểu DataInputStream
static DataOutputStream openDataOutputStream(String name)	Tạo kết nối luồng xuất kiểu DataOutputStream

Generic Connection Framework

Tìm hiểu kết nối qua Socket

- Socket có thể thực thi giao tiếp giữa hai hệ thống và cho phép kết nối mạng được coi như là một dòng (*stream*).
- Một socket tạo thành một dòng có thể đọc hoặc ghi.
- *GCF* cung cấp hai giao diện để làm việc với dòng:
    * *StreamConnectionNotifier*
    * *StreamConnection*
- *StreamConnectionNotifier* được dùng cho máy chủ quản lý cổng mà máy khách kết nối đến.

Generic Connection Framework

Hoạt động của Socket:

- Máy chủ lắng nghe các kết nối từ máy khách
- Máy khách tạo kết nối đến máy chủ
- Khi đã kết nối thành công thì trao đổi dữ liệu

Generic Connection Framework

Hành động có thể làm trên Socket

a. Viết từ Socket: Khi đã kết nối tốt, một dòng ra được sinh ra từ StreamConnection dùng phương thức openOutputStream() hoặc openDataOutputStream() và dùng một vài phương thức của OutputStream để viết dữ liệu . Lớp liên quan: OutputStream, DataOutput, DataOutputStream…

b. Đọc từ Socket: API dùng để đọc từ socket cũng tương tự như các APIs output nhưng dùng để đọc. Khi đã có dòng vào có một số lớp có thể giúp lấy dữ liệu từ một dòng. Các lớp liên quan: InputStream, DataInput, DataInputStream, Reader, InputStreamReader.

Generic Connection Framework

Khi nào dùng Socket

- Khi yếu tố tốc độ được đặt lên hàng đầu

Generic Connection Framework

Tìm hiểu Datagram

- Datagram được thiết kế để gửi gói dữ liệu trên một mạng.
- Datagram cho phép gửi dữ liệu cho dù máy phục vụ có lắng nghe hay không.
- Datagram là cách gửi dữ liệu không chắc chắn.
- Datagram gói dữ liệu theo gói và gửi không cần biết thứ tự

Generic Connection Framework

Datagram trong J2ME

- Trong GCF Datagram tồn tại trong hai lớp là *DatagramConnection* và *Datagram*.

- *Datagram* không giống như một dòng.

- Để gửi dữ liệu sử dụng J2ME API bạn cần ba thứ: địa chỉ gửi gói dữ liệu, cổng nơi hệ thống nhận dữ liệu và dữ liệu.

- Dữ liệu trong Datagram ở dạng byte

Ví dụ

- Máy khách kết nối

```
DatagramConnection conn=
    (DatagramConnection)Connector.open("datagram://127.0.
    0.1:88",Connector.READ_WRITE);
```

- Tạo đối tượng Datagram để gửi thông báo

```
Datagram  datagr= conn.newDatagram(100);
```

- Chuyển dữ liệu sang dạng byte

```
byte [] data="Message from client".getBytes();
```

- Đặt dữ liệu vào Datagram

```
datagr.setData(data,0,data.length);
```

- Gửi dữ liệu

```
conn.send(datagr);
```

Tài liệu tham khảo

- Kim Topley, *J2ME in a Nutshell*, O'Reilly, 2002
- Roger Riggs, *Programming Wireless Devices with the Java™ 2 Platform Micro Edition Second Edition*, Addison Wesley, 2003
- http://java.sun.com
- http://www.javavietnam.org

Q/A