



Sesión 10: Conexiones de red

Índice




- Marco de conexiones genéricas
- Conexión HTTP
- Envío y recepción de datos
- Conexiones a bajo nivel
- Mensajes SMS
- Bluetooth
- Servicios Web

Conexiones de red



- Marco de conexiones genéricas
- Conexión HTTP
- Envío y recepción de datos
- Conexiones a bajo nivel
- Mensajes SMS
- Bluetooth
- Servicios Web

GCF



- GCF = Generic Connection Framework**
 - Marco de conexiones genéricas, en `javax.microedition.io`
 - Permite establecer conexiones de red independientemente del tipo de red del móvil (circuitos virtuales, paquetes, etc)
- Cualquier tipo conexión se establece con un único método genérico**

```
Connection con = Connector.open(url);
```
- Según la URL podemos establecer distintos tipos de conexiones**


| | |
|--|--------------|
| <code>http://jee.ua.es/pdm</code> | HTTP |
| <code>datagram://192.168.0.4:6666</code> | Datagramas |
| <code>socket://192.168.0.4:4444</code> | Sockets |
| <code>comm:0;baudrate=9600</code> | Puerto serie |
| <code>file:/fichero.txt</code> | Ficheros |

Programación de Dispositivos Móviles

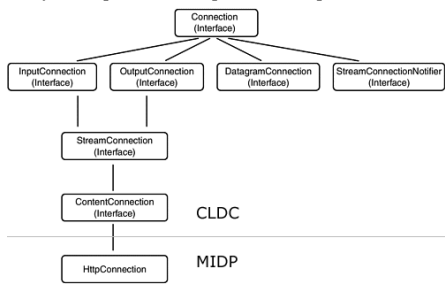
© 2006 Depto. Ciencia Computación e IA

Conexiones de red-4

Tipos de conexiones



- En CLDC se implementan conexiones genéricas
- En MIDP y APIs opcionales se implementan los protocolos concretos



```


graph TD
    C[Connection Interface] --> IC[InputConnection Interface]
    C --> OC[OutputConnection Interface]
    C --> DC[DatagramConnection Interface]
    C --> SCN[StreamConnectionNotifier Interface]
    IC --> SC[StreamConnection Interface]
    OC --> SC
    SC --> CC[ContentConnection Interface]
    CC --> HC[HttpConnection]
  
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-5

Conexiones de red



- Marco de conexiones genéricas
- Conexión HTTP**
- Envío y recepción de datos
- Conexiones a bajo nivel
- Mensajes SMS
- Bluetooth
- Servicios Web

Programación de Dispositivos Móviles

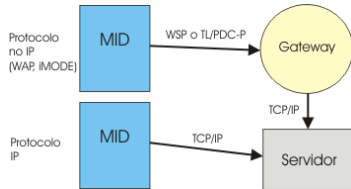
© 2006 Depto. Ciencia Computación e IA

Conexiones de red-6

Conexión HTTP



- El único protocolo que se nos asegura que funcione en todos los móviles es HTTP
 - Funcionará siempre de la misma forma, independientemente del tipo de red que haya por debajo



Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-7

Leer de una URL



- Abrimos una conexión con la URL

```
HttpConnection con = (HttpConnection)Connector.open(
    "http://j2ee.ua.es/index.htm");
```

- Abrimos un flujo de entrada de la conexión

```
InputStream in = con.openInputStream();
```

- Podremos leer el contenido de la URL utilizando este flujo de entrada

- Por ejemplo, en caso de ser un documento HTML, leeremos su código HTML

- Cerramos la conexión

```
in.close();
con.close();
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-8

Mensaje de petición



- Podemos utilizar distintos métodos

```
HttpConnection.GET
HttpConnection.POST
HttpConnection.HEAD
```

- Para establecer el método utilizaremos:

```
con.setRequestMethod(HttpConnection.GET);
```

- Podemos añadir cabeceras HTTP a la petición

```
con.setRequestProperty(nombre, valor);
```

- Por ejemplo:

```
c.setRequestProperty("User-Agent",
    "Profile/MIDP-1.0 Configuration/CLDC-1.0");
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-9

Mensaje de respuesta



- A parte de leer el contenido de la respuesta, podemos obtener

- Código de estado

```
int cod = con.getResponseCode();  
String msg = con.getResponseMessage();
```

- Cabeceras de la respuesta

```
String valor = con.getHeaderField(nombre);
```

- Tenemos métodos específicos para cabeceras estándar

```
getLength()  
getType()  
getLastModified()
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-10

Conexiones de red



- Marco de conexiones genéricas
- Conexión HTTP
- Envío y recepción de datos
- Conexiones a bajo nivel
- Mensajes SMS
- Bluetooth
- Servicios Web

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-11

Enviar datos



- Utilizar parámetros

- GET o POST

- Parejas <nombre, valor>

```
HttpConnection con = (HttpConnection)Connector.open(  
    "http://j2ee.ua.es/registra?nombre=Pedro&edad=23");
```

- No será útil para enviar estructuras complejas de datos

- Añadir los datos al bloque de contenido de la petición

- Debemos decidir la codificación a utilizar

- Por ejemplo, podemos codificar en binario con
DataOutputStream

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-12

Tipos de contenido



- Para enviar datos en el bloque de contenido debemos especificar el tipo MIME de estos datos

➢ Lo establecemos mediante la cabecera `Content-Type`

```
con.setRequestProperty("Content-Type", "text/plain");
```

➢ Por ejemplo, podemos usar los siguientes tipos:

| | |
|--|-----------------|
| <code>application/x-www-form-urlencoded</code> | Formulario POST |
| <code>text/plain</code> | Texto ASCII |
| <code>application/octet-stream</code> | Datos binarios |

Codificación de los datos



- Podemos codificar los datos a enviar en binario

➢ Establecemos el tipo MIME adecuado

```
con.setRequestProperty("Content-Type",  
    "application/octet-stream");
```

➢ Utilizaremos un objeto `DataOutputStream`

```
DataOutputStream dos = con.openDataOutputStream();  
dos.writeUTF(nombre);  
dos.writeInt(edad);  
dos.flush();
```

- Si hemos definido serialización para los objetos, podemos utilizarla para enviarlos por la red

Leer datos de la respuesta



- Contenido de la respuesta HTTP

➢ No sólo se puede utilizar HTML

➢ El servidor puede devolver contenido de cualquier tipo

➢ Por ejemplo, XML, ASCII, binario, etc

- Si el servidor nos devuelve datos binarios, podemos descodificarlos mediante `DataInputStream`

```
DataInputStream dis = con.openDataInputStream();  
String nombre = dis.readUTF();  
int precio = dis.readInt();  
dis.close();
```

- Podría devolver objetos serializados

➢ Deberíamos deserializarlos con el método adecuado

Conexiones de red



- Marco de conexiones genéricas
- Conexión HTTP
- Envío y recepción de datos
- **Conexiones a bajo nivel**
- Mensajes SMS
- Bluetooth
- Servicios Web

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-16

Conexiones a bajo nivel



- A partir de MIDP 2.0 se incorporan a la especificación conexiones de bajo nivel
 - Sockets
 - Datagramas
- Nos permitirán aprovechar las características de las nuevas redes de telefonía móvil
- Podremos acceder a distintos servicios de Internet directamente
 - Por ejemplo correo electrónico
- Su implementación es optativa en los dispositivos MIDP 2.0
 - Depende de cada fabricante

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-17

Sockets



- **Establecer una comunicación por sockets**

```
SocketConnection sc = (SocketConnection)
Connector.open("socket://host:puerto");
```

- **Abrir flujos de E/S para comunicarnos**

```
InputStream in = sc.openInputStream();
OutputStream out = sc.openOutputStream();
```

- **Podemos crear un socket servidor y recibir conexiones entrantes**

```
ServerSocketConnection ssc = (ServerSocketConnection)
Connector.open("socket://:puerto");
SocketConnection sc =
(SocketConnection) ssc.acceptAndOpen();
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-18

Datagramas



- Crear conexión por datagramas

```
DatagramConnection dc = (DatagramConnection)
Connector.open("datagram://host:puerto");
```

- Crear un enviar paquete de datos

```
Datagram dg = dc.newDatagram(datos, datos.length);
dc.send(dg);
```

- Recibir paquete de datos

```
Datagram dg = dc.newDatagram(longitud);
dc.receive(dg);
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-19

Conexiones de red



- Marco de conexiones genéricas
- Conexión HTTP
- Envío y recepción de datos
- Conexiones a bajo nivel
- Mensajes SMS
- Bluetooth
- Servicios Web

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-20

Conexión de mensajes



- Con WMA podremos crear conexiones para enviar y recibir mensajes de texto SMS

- Utilizaremos una URL como

```
sms://telefono:[puerto]
```

- Creamos la conexión

```
MessageConnection mc = (MessageConnection)
Connector.open("sms://+34555000000");
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-21

Envío de mensajes



▪ Componemos el mensaje

```
String texto =  
    "Este es un mensaje corto de texto";  
TextMessage msg = mc.newMessage(mc.TEXT_MESSAGE);  
msg.setPayloadText(texto);
```

▪ El mensaje no deberá pasar de 140 bytes

- Si se excede, podría ser fraccionado
- Si no puede ser fraccionado, obtendremos un error

▪ Enviamos el mensaje

```
mc.send(msg);
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-22

Recepción de mensajes



▪ Creamos conexión de mensajes entrantes

```
MessageConnection mc = (MessageConnection)  
    Connector.open("sms://:4444");
```

▪ Recibimos el mensaje

```
Message msg = mc.receive();
```

▪ Esto nos bloqueará hasta la recepción

- Para evitar estar bloqueados, podemos utilizar un listener
- Con un `MessageListener` se nos notificará de la llegada de mensajes

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-23

Conexiones de red



▪ Marco de conexiones genéricas

▪ Conexión HTTP

▪ Envío y recepción de datos

▪ Conexiones a bajo nivel

▪ Mensajes SMS

▪ Bluetooth

▪ Servicios Web

Programación de Dispositivos Móviles

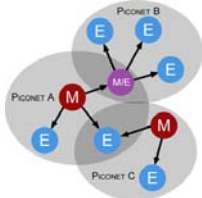
© 2006 Depto. Ciencia Computación e IA

Conexiones de red-24

Redes bluetooth



- Las redes bluetooth son redes “ad hoc”
 - La red se crea dinámicamente
 - Tenemos la capacidad de “descubrir” dispositivos
 - Conecta pequeños dispositivos (sustituye al cable)
- Topología



Piconet: 1 maestro con 7 esclavos como máximo

Scatternet: Conexión de varias piconets

Programación de Dispositivos Móviles

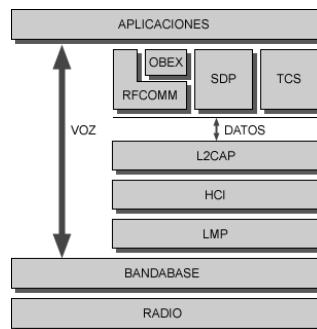
© 2006 Depto. Ciencia Computación e IA

Conexiones de red-25

Protocolos bluetooth



- **L2CAP**
 - Protocolo a bajo nivel
 - Transmisión de paquetes
 - Sin control de flujo
- **RFCOMM**
 - Puerto serie sobre radio
 - Realiza control de flujo
- **SDP**
 - Descubrimiento de dispositivos



Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-26

Servicios



- Los servicios se identifican mediante un **UUID**
 - *Universal Unique Identifier*
 - Clave de 128 bits, única en el tiempo y en el espacio
 - Se puede generar con herramientas como `uuidgen`
- Podemos buscar dispositivos y explorar los servicios que ofrecen
 - Los servicios se buscarán mediante su UUID
 - Tipos de búsqueda de dispositivos:
 - GIAC:** General.
 - Encuentra tanto dispositivos descubribles GIAC como LIAC.
 - LIAC:** Limitada. Para búsquedas acotadas.
 - Sólo encuentra dispositivos descubribles LIAC.

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-27

Publicar un servicio (servidor)



- Generar UUID para nuestro servicio

```
public final static String UUID =  
    "000000000000010008000123456789ab";
```

- Hacemos nuestro dispositivo local descubrible

```
LocalDevice ld = LocalDevice.getLocalDevice();  
ld.setDiscoverable(DiscoveryAgent.GIAC);
```

- Crear servicio

```
StreamConnectionNotifier scn =  
    (StreamConnectionNotifier)  
    Connector.open("btspp://localhost:" + UUID );
```

- Aceptar conexiones entrantes

```
StreamConnection sc =  
    (StreamConnection)scn.acceptAndOpen();  
InputStream is = sc.openInputStream();  
OutputStream os = sc.openOutputStream();
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-28

Descubrir dispositivos y servicios (cliente)



- Obtener agente de descubrimiento

```
LocalDevice ld = LocalDevice.getLocalDevice();  
DiscoveryAgent da = ld.getDiscoveryAgent();
```

- Crear un objeto DiscoveryListener

```
deviceDiscovered(RemoteDevice rd, DeviceClass dc);  
inquiryCompleted(int tipo);  
servicesDiscovered(int transID, ServiceRecord[] srvs);  
serviceSearchCompleted(int transID, int estado);
```

- Comenzar búsqueda de dispositivos

```
da.startInquiry(DiscoveryAgent.GIAC, mListener);
```

- Buscar servicios de un dispositivo

```
da.searchServices(null, new UUID[]{new UUID(UUID,false)},  
    (RemoteDevice)obtenerDispositivoRemoto(),mListener);
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

Conexiones de red-29

Conectar a un servicio (cliente)



- Una vez descubiertos los servicios de nuestro entorno, habremos obtenido varios objetos `ServiceRecord` con cada uno de estos servicios

```
ServiceRecord rs =  
    (ServiceRecord)obtenerServicioRemoto();
```

- Obtener URL de conexión al servicio

```
String url = rs.getConnectionURL(  
    ServiceRecord.NOAUTHENTICATE_NOENCRYPT, true);
```

- Establecer la conexión

```
StreamConnection sc =  
    (StreamConnection)Connector.open(url);  
InputStream is = sc.openInputStream();  
OutputStream os = sc.openOutputStream();
```

Programación de Dispositivos Móviles

© 2006 Depto. Ciencia Computación e IA

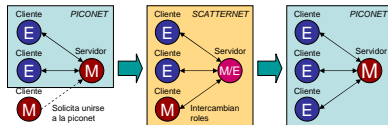
Conexiones de red-30

Roles maestro/esclavo



- Quien realiza la conexión (cliente) actuará como maestro
 - Indicando `;"master=true"` en la URL de quien publica el servicio (servidor) podemos forzar a que sea éste quien se comporte como maestro

```
Connector.open("btspp://localhost:" + UUID + ";" + "master=true");
```

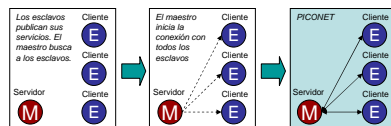


- Esto no funciona en dispositivos que no soporten el intercambio de roles maestro/esclavo

Conexión punto-a-multipunto



- Para poder hacer conexiones punto-a-multipunto en estos casos deberemos:
 - Abrir varios dispositivos que publiquen servicios.
 - En este caso, éstos serán los “clientes”.
 - Iniciar la conexión a todos estos clientes desde un único maestro
 - Este maestro será el “servidor”, ya que es quien coordinará las múltiples conexiones de los clientes.



Seguridad en bluetooth



- Podemos forzar la utilización de diferentes tipos de seguridad en las conexiones bluetooth:
 - Autenticación (`;"authenticate=true"`)
 - Los usuarios de los móviles deben conocerse
 - Se resuelve mediante “emparejamiento” (*pairing*)
 - Los usuarios de los móviles que se conectan deben introducir un mismo código secreto en sendos dispositivos
 - Autorización (`;"authorize=true"`)
 - Quien solicita la conexión a un servicio debe tener autorización
 - Si no está en una lista de dispositivos de confianza, se preguntará al usuario si acepta la conexión
 - Encriptación (`;"encrypt=true"`)
 - Los datos se transmiten encriptados
 - Requiere estar autenticado

Conexiones de red

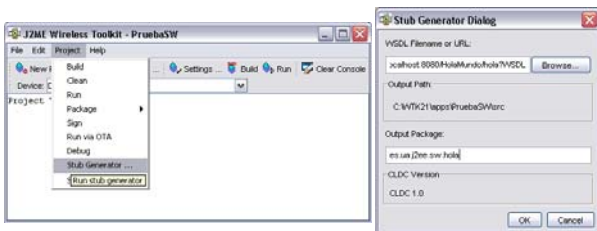


- Marco de conexiones genéricas
- Conexión HTTP
- Envío y recepción de datos
- Conexiones a bajo nivel
- Mensajes SMS
- Bluetooth
- Servicios Web

Servicios web



- Podemos acceder a servicios web desde dispositivos mediante Web Services API
 - Los servicios deben ser de tipo `document/literal`
- Podemos generar un stub mediante WTK 2.2



Invocación de servicios



- Utilizamos el stub para acceder al servicio

```
HolaMundoIF hola = new HolaMundoIF_Stub();
try {
    String saludo = hola.saluda("Miguel");
} catch (RemoteException re) { // Error }
```

- Los servicios web requieren
 - Gran cantidad de memoria
 - Gran cantidad de procesamiento
 - Gran cantidad de tráfico por la red (XML)
- Esto los hace poco adecuados para los dispositivos actuales