



Mobile Information Device Programming (14)

Lecturer: Alireza Mousavi
School of Engineering & Design
www.brunel.ac.uk/~emstaam

Additional **Source**: Y. Feng and J. Zhu (2001), **Basic Network Programming in J2ME MIDP**, provided by **Sams**, <http://www.sampublishing.com>



MIDP Networking & Serial Communications

1. Characteristics of Wireless Data Networks
2. Network Interface
3. Generic Connection Framework
4. HTTPConnection Interface
 1. HTTP connection **States**
 2. Security
 3. Connection Construct
 4. Establishing Connections
 5. Client/Server
5. Socket Connection
6. CommConnection



Facts

- MIDP target devices operate on a wide range of wireless and wired networks with incompatible transport protocols
- Wireless networks have **less bandwidth**, **more latency** and **less stable connections** than wired networks
- MIDP needs to cope with this and allow for future developments
- MIDP expects that **HTTP** and **HTTPS** to be standard protocol for all devices
- MIDP defines optional protocols for datagrams, sockets, secure sockets, and serial communication ports



Facts cont.

- The networking in J2ME needs to be very flexible to cater for a variety of devices (device specific)
- The Generic Connection framework(GCf) is defined in CLDC
- The Generic Connection framework(GCf) provides a general abstract description of the networking and file input/outputs to cover as large range of devices as possible
- The abstractions provided by GCf are defined as Java interfaces
- Based on device capabilities the OEM choose the interfaces they want to be implemented by their devices



Characteristics of Wireless Data Networks

MIDP 2.0 covers two types of wireless networks

1. Circuit-Switched data (CSD) → GSM (10kbps)
2. Packet-Switched data (PSD) → GPRS (170kbps)



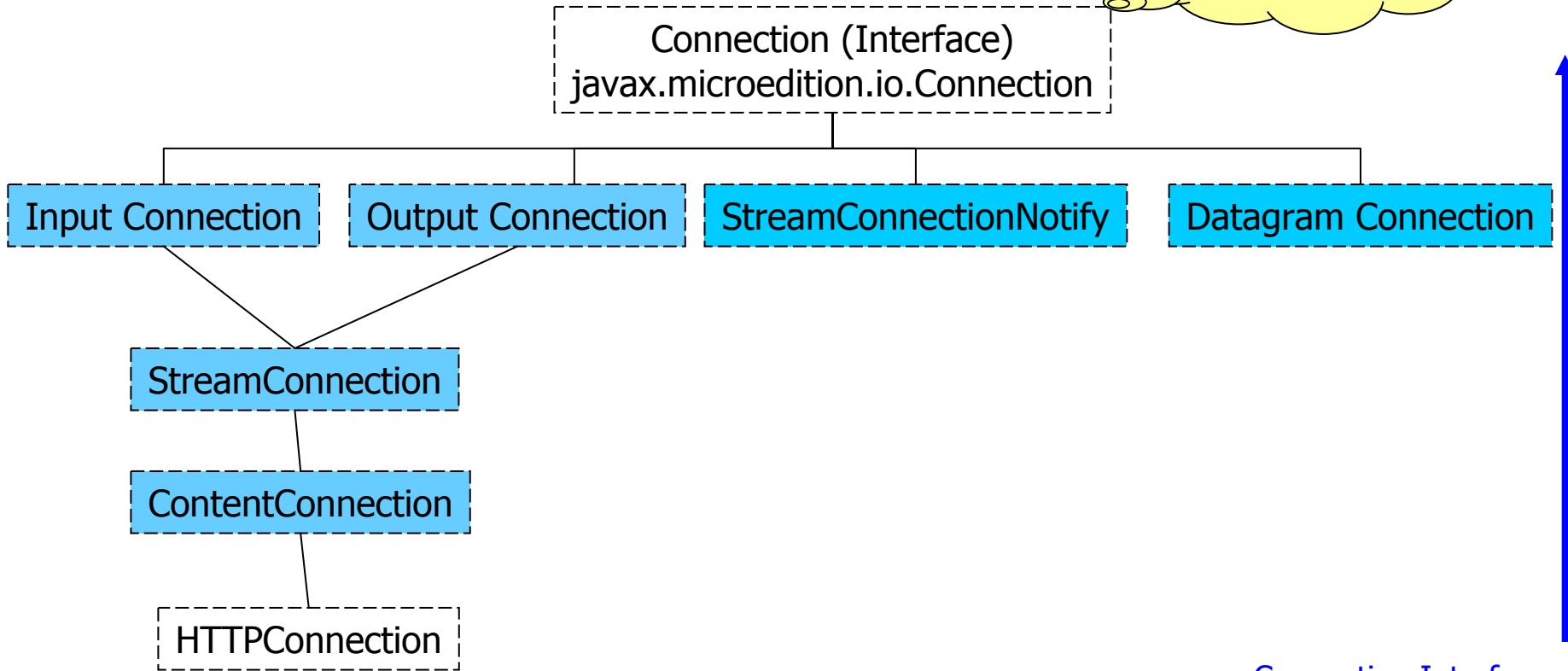
Network Interface

- MIDP devices should support HTTP (Why ?)
- A device can implement HTTP using:
 - IP protocols such as TCP/IP, or
 - Non-IP protocols such as WAP or i-Mode (in this case a specific gateway is required)
- MIDP 2.0 contains required APIs to maintain data comms. These APIs are:
 - ServerSocketConnection
 - UDPDatagramConnection
 - CommConnection
- It enables MIDlets using the protocols to run properly across implementation platforms



Generic Connection framework

Top Layer
Generic
Abstract type of connections



Connection Interface
Hierarchy

Increase in level
Of abstraction



Generic Connection framework cont.

- **InputConnection:**
 - This interface describes a specific connection's input stream data(*InputStream*)
 - The data is in a stream of bytes
 - See [slide 14](#) for the two methods associated with this interface
- **OutputConnection**
 - Another subclass of *Connection* interface
 - Designed to handle output streams
 - Use this interface for writing byte-oriented data to destination
 - See [slide 14](#) for the two methods associated with this interface

“By using the combination of these two interfaces you will be able to handle the input or output stream to or from the device”



Generic Connection framework cont.

- **StreamConnection:**

- Extends InputConnection and OutputConnection interfaces
- Inherits the methods from both [\(four methods\)](#)
- It represents a connection as a stream of data
- It serves as a place holder representing any type of connection, where the data can be represented as a stream of bytes
- The StreamConnection interface holds detail information about:
 - connection mechanism
 - The protocol used in implementing a connection
 - The syntax and semantics

in an abstract form.

- It offers both send and receive capabilities for socket-based communications



Generic Connection framework cont.

- **URLConnection:**

- This interface provides the platform for describing and providing content instead of bytes of raw data
- The idea behind this interface is to describe the content in a higher level defined by the protocol
- It provides the abstract to represent the content
- It conforms to set of protocols that describe the data they transport
- The interface contains methods:
 - *String getEncoding()*
 - *Long getLength()*
 - *String getType()*

"For a full list of HttpURLConnection Interface Methods please see: Piroumian, V. (2002), Wireless J2ME Platform Programming, Sun Micro Systems Press"



Generic Connection framework cont.

- **DatagramConnection**

- This interface is used to create a datagram for a UDP communication (for further information see R. Riggs et al 2003)
- There are several methods in this interface ([see next slide](#))
- An application opens a UDP datagram connection using *Connector.open* method
- The application provides a URI string that starts with “*datagram:// hostname, port no.*”
- What is a [Datagram](#)?



DatagramConnection Methods

Method	What it does
<i>int getMaxLength()</i>	Returns the maximum allowed length for a datagram packet
<i>int getNominalLength()</i>	Returns the nominal length for a datagram packet
<i>Datagram newDatagram(byte[] buf, int size)</i>	Creates a new datagram object, <i>buf</i> is the placeholder, and <i>size</i> is the length of the buffer to be allocated
<i>Datagram newDatagram(byte[] buf, int size, String location)</i>	Creates new datagram object, parameter <i>location</i> points to the destination of the datagram message.
<i>Datagram newDatagram(int size)</i>	Creates a new datagram object with an automatically allocated buffer size
<i>Datagram newDatagram(int size, String loc)</i>	Creates a new datagram object and <i>loc</i> specifies the destination of this datagram message
<i>void receive(Datagram rxdgram)</i>	Receives datagram object <i>rxdgram</i> from a remote host
<i>void send(Datagram sxdgram)</i>	Sends a datagram object <i>sxdgram</i> to the remote host
For further information consult: www.sampublishing.com/articles	



Connector.open() **method**

The GCf supports the following form of connections by calling the *Connector.open()* method in the **Connector Class**

1. HTTP → *Connector.Open("http://www.brunel.ac.uk/midp_lab");*
2. Sockets → *Connector.Open("socket :// localhost: 3322");*
3. Datagrams → *Connector.Open("datagram:// http://www.midplab.com: 9000");*
4. Serial Port → *Connector.Open("comm: 0; baudrate = 9600");*
5. File → *Connector.Open("file: /mydata/info.dat*



Establishing Connections (the *Connector* Class)

The seven methods that the *Connector* class provides to establish a connection (all *static* methods)

Method	What it does
<i>static Connection</i> open (<i>String name</i>)	Create Connection in Read_Write CLDC Connector mode
<i>static Connection</i> open (<i>String name, int mode</i>)	Create Connection in a specific CLDC Connector mode
<i>static Connection</i> open (<i>String name, int mode, boolean timeouts</i>)	Create Connection in a specific CLDC Connector mode and handles time out exceptions
<i>static InputStream</i> openInputStream (<i>String name</i>)	Create and open connection input stream
<i>static OutputStream</i> openOutputStream (<i>String name</i>)	Create and open connection output stream
<i>static DataInputStream</i> openDataInputStream (<i>String name</i>)	Create and open connection input data stream (subclass of <i>InputStream</i>)
<i>static DataOutputStream</i> openDataOutputStream (<i>String name</i>)	Create and open connection output data stream

Source: Machow, J. (2002), Sun Micro Systems



Connecting to Communication ports

- The CLDC method *javax.microedition.io.Connector.open* are used by Applications to HTTP connections
- The method takes the URL (String) parameter in the URI defined by [IETF](#)
- Opening the connection may interfere(block) the application thread until the server responds
- This line of code opens connection to the University's home page:

```
HttpConnection myconnection =  
    (HttpConnection)Connector.open("http://www.brunel.ac.uk");
```



An advice for **YOU** application developers

- When designing applications notice the networking and communications constraints
- Calling APIs can be time consuming → Block network activity
- **Example:** CommandListener functions



HttpConnection Interface

MIDP HTTP API interface is:

javax.microedition.io.HttpConnection

It provides:

- Additional field and methods that parse URLs
- Sets request headers
- Parse response headers



HTTP connection states

There are three states:

- *Setup*
- *Connected*
- *Closed*



Example

// Open connection and input stream

```
HttpConnection Con = (HttpConnection)Connector.open( ... );
```

```
InputStream InStrm = Con.openInputStream( );
```

```
...
```

// Close the connection

```
Con.close( );
```

```
While ((int x = InStrm.read( ) != -1){
```

```
...
```

```
}
```

// Close the input stream, InputStream methods cannot be called after this

```
InStrm.close( );
```

Http Security



- Pending device setup access to HTTP may be restricted
- Only trusted MIDlets can request access defined by:
javax.microedition.io.Connector.http
permissions.
- It is described in the "*Trusted MIDlet Suite Security Model*"
- When a MIDlet tries to open connection the device will attempt to authorise. If fails to do so
java.lang.SecurityException is thrown



Exercise 14.1

- Download information from `www`.
- Create a Textbox to enter the URL
- Provide the command "*retrieve*" to retrieve the information from the target source
- Provide an "*Exit*" command to exit application



Next Session

We will discuss:

- Connections Continued
 - Applications that use Socket Connections
 - Applications that use UDP
- MIDP suites security issues



Datagrams further explanation

- There are cases where your application cannot rely on point-to-point channels provided by TCP
- In these instances your application will rely on deliveries of packages of information whose arrival and order is not guaranteed
- UDP standards provide a mode of network comms. Where applications send packet of data called **datagrams**
- **Note** that many IS security systems e.g. firewalls and routers are set to prevent UDP packets, so if there is a problem with connecting to the server by client you need to set the configuration to accept UDP packets.