



Mobile Computing

Exercise 2

Assigned: November 7, 2005

Due: November 21, 2005

1 Mobile “Hello World”

In this exercise you will implement two little Java programs to send a message—such as “Hello World”—from one device and receive it on another mobile device.

For this purpose we use the `java.net.MulticastSocket` class to send and receive `java.net.DatagramPackets`. If the system is configured correctly according to Exercise 1, the messages will automatically be sent and received via the Wireless LAN cards by the Java communication layers. Read the Java documentation of the above two classes for information on how to send/receive packets. Make sure you use the same group (multicast address) and port on both the sender and receiver side.

Note that the message will only be transferred if the receiver device is located within the transmission range of the sender. Packets are *not* automatically relayed by other mobile devices.

Also use the ad-hoc emulator to develop and test your programs.

2 Abstraction Layer

The goal of this exercise is to build an abstraction layer which handles all low-level details you had to consider in the above exercise. This abstraction layer will provide a simple send/receive interface to other layers or applications to be implemented in later exercises.

In addition to abstraction, we also introduce an addressing scheme. We use 16 bit numbers to distinguish different devices. Every packet begins with a header consisting of the sender followed by the receiver address. In the packet, the more significant byte of an address is placed *before* the less significant byte. If the receiver address is 0, the packet is to be broadcast, that is, all devices within transmission range should receive the packet.

Use what you have learned so far to implement the following two Java Interfaces, which can be downloaded from the lecture website (<http://dcg.ethz.ch/mobicomp/>):

```
public interface Message {  
  
    /**  
     * Set/get the sender address of the packet  
     */  
    public void setSender(int addr);  
    public int getSender();  
  
    /**  
     * Set/get the receiver address of the packet  
     */  
}
```

```

public void setReceiver(int addr);
public int getReceiver();

/**
 * Set/get the data part of the packet
 */
public void setData(byte[] data);
public byte[] getData();
}

public interface SHSocket {

/**
 * This address and port must be used for all communication
 * on this layer. Only clients using the same values will
 * be able to communicate with each other.
 */
public static final String GROUP_ADDRESS = "239.0.0.1";
public static final int COMMUNICATION_PORT = 32224; // 'dcg' in sms language

/**
 * All clients receiving a message with this value
 * in the receiver address field will handle the message.
 */
public static final int BROADCAST_ADDRESS = 0;

/**
 * Set/get our own address
 */
public void setOwnAddress(int addr);
public int getOwnAddress();

/**
 * Send a packet
 */
public void send(Message p) throws IOException;

/**
 * Receive a packet in a blocking way
 */
public Message receive() throws IOException;
}

```

You can now adjust the above mobile “Hello World” application to use these interfaces. In order to allow communication with other students’ solutions, use the multicast group 239.0.0.1 and port 32224. Implement the application such that you can set your own ad-hoc address at program start. How can you learn about someone else’s address without special agreements?