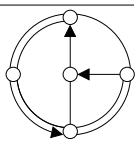
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



SS 2002 Roger Wattenhofer, Keno Albrecht, Fabian Kuhn, Aaron Zollinger

Mobile Computing
Exercise 2

Assigned: April 18, 2002
Due: April 25, 2002

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.

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 less significant byte of an address is placed *before* the more 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://distcomp.ethz.ch/mobicomp/`):

```
public interface Packet {

    /**
     * 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 void setData(String str);
    public byte[] getData();
    public String getStringData();

    /**
     * Get data length
     */
    public int getDataLength();

}

public interface SHSocket {

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

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

    /**
     * Receive a packet in non-blocking or blocking way
     */
    public boolean receiveIfAvailable(Packet p) throws IOException;
    public void receive(Packet p) throws IOException;

    /**
     * Factory methods to create packets
     */
    public Packet createPacket(int receiver);
    public Packet createBroadcastPacket();

}
```

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 4567. Choose your own ad hoc addresses at random ("ad hoc" ;-). How can you learn about someone else's address without special agreements?