Assignment #3: Wake-On-LAN Server

A way to save energy is to suspend or shutdown a computer when not in use. Many computers can be woken up if a special packet is received (see http://en.wikipedia.org/wiki/Wake-on-LAN). The so-called magic packets needed to wake-up a computer on your own LAN can be UDP datagrams. This is the choice we will use for this assignment.

This assignment is about you implementing a Wake-On-LAN server that accepts wake-up scheduling instructions through the network. Such an instruction consist of three values: the MAC address of the computer to wake up, the IP broadcast address of the network and the number of seconds before sending the wake-up message. All this data is encoded in the data field of a UDP datagram sent to port 2323 of the server. Server will return a UDP datagram containing the string "Ok" if previous request was accepted (i.e. no format errors) or "Error" if anything was wrong in the client request.

To make it easy for development and testing, requests will be encoded using ASCII text. Requests have the following syntax: <*MAC_Address*><sp><*IP_Address*><sp><*seconds_to_wakeup*>. Where *MAC_Address* format is hh:hh:hh:hh:hh:hh and hh is a two-digit lowercase hexadecimal number and *IP_Address* is the regular decimal dotted notation ddd.ddd.ddd.ddd (ddd may be one, two or three digits long). The *seconds_to_wakeup* field will be an unsigned integer number between 0 and 2^16. This way you can use `nc` shell command as a client. For example: "`nc -u localhost 2323`" and then "`12:34:56:78:ab:ff 192.168.1.255 600`" for waking up that computer in ten minutes time.

Server may accept an unlimited number of wake-up scheduling requests. A pending request can be cancelled if it is repeated with time value equals zero (same MAC and IP addresses and time till wake-up is 0).

Once the time for a wake-up has elapsed, the server will send the proper wake-up datagram to the designated broadcast address. Data format for magic packet for the example above is {0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x12, 0x34, 0x56, 0x78, 0xab, 0xff, ... repeat last 6 bytes 15 times more}. And that data is sent to destination address 192.168.1.255 and to port number 7. No answer is expected from the computer being woken up.

Your Code

Your code has to conduct the following tasks:

1. Server listening on port 2323 has to be accepting schedule-request datagrams and returning the corresponding response (Error or Ok).
2. Server has to manage a list that contains all the pending wake-up operations. Once a wake-up has been performed, the corresponding entry is removed from the list. If a new request is received and time is not zero, then it is added to the list of pending requests.
3. If a second request is received for the same MAC address and the requested time is not zero, then it will overwrite the previous wake-up time.
4. If the server receives a scheduling request with a value of zero for the time parameter, then the pending wake-up in the list for this MAC address is removed.
5. Server never ends.
6. The server will print out to the standard output the wake-up schedule list when it receives a

datagram whose data length is less than 10 bytes (no format-check will be done for these).

7. At any moment, the server may receive a new scheduling request without missing a single wake-up operation.

Some tips

- You can use a single DatagramSocket at port 2323.
- A sample wakeOnLan method is provided. Feel free to use it.
- I used a HashMap for my implementation, you may want to have look at it (list management).
- Please remember you can try this at home even with only one computer. Just open several windows and use localhost as the destination address for nc command.
- You cannot wake-up your own computer with your code. If you want to do real-life tests you will need a second computer. Please note that most laptops cannot be woken up by wifi traffic.
- For a computer to be able to respond to a wake-up request several conditions need to be met: wake-on-lan needs to be enabled (some computers will need BIOS setup, others need configuration software, like `ethtool` in Linux) and the computer has to be shutdown or suspended properly for wake-up to be possible.
- Whether you use two threads or a single one is your implementation choice, but make sure that you do not miss incoming datagrams nor wake-up operations.
- Server printing the pending wake-up operations can be very handy for you to check the management of it is done properly.
- Please note that all the requests **must** use the same broadcast IP address.

Due date **misan@disca.upv.es**

Your Java source code has to be submitted by email by **March 25th, 2011**. Main class name MUST BE **WolServer**. Please send me the java source code file(s) by email to the email account shown here with EXACTLY the subject: **Assignment#3**

Please include a MANDATORY **comment on the source code with your name**. If you have any doubt about the assignment I suggest you to stop by my office during office hours (posted on the web).

Assessment

Assignment work only gets a pass/fail mark.

If your code works (in my computer) then you pass the assignment. Early submissions (before deadline) can benefit of an early warning: If your code does not performs as expected, I'll report that back to you so you can fix it before the deadline. Please remember assignments are optional. If you prefer to use other programming language different than Java, please talk to me first. Assignment work is to be performed individually without sharing code with other students.

Appendix: Sample Code

```
public static void wakeOnLan(String mac, InetAddress ip) {
        DatagramSocket ds = null;
        try { ds = new DatagramSocket(); } catch(SocketException e) {System.err.println("Socket error"); }
        String[] addr=mac.split(":");
        byte[] buffer = new byte[102];
        for(int i=0;i<6; i++) {
                buffer[i]=-1; // -1 == 0xff
                for(int j=0;j<16; j++) buffer[(j+1)*6+i]=(byte) ((Character.digit(addr[i].charAt(0), 16) << 4)
                + Character.digit(addr[i].charAt(1), 16));
        }
        DatagramPacket dp = new DatagramPacket(buffer,102,ip,7);
        try { ds.send(dp); } catch (IOException e) {System.err.println("Cannot send magic packet"); }
}
```

Appendix: Sample client dialog

```
$ nc -u localhost 2323
12:34:55:12:12:12 192.168.123.15 500
Ok
11:22:33:FF:5B:49 192.168.123.255 500
Error
11:22:33:ff:5b:49 192.168.123.255 500
Ok
11:22:33:55:58:49 192.168.255 500
Error
```