Carleton University
Department of Systems and Computer Engineering
SYSC 3303 - Real-Time Concurrent Systems - Summer 2013

Assignment 1

Background

The goal of this assignment is to build a very basic 3 part system consisting of a client, an intermediate host, and a server. Clients send requests to the intermediate host, which sends them on to the server. The server sends responses to the intermediate host, which sends them on to the client. From the client's point of view, the intermediate host appears to be the server. From the server's point of view, the intermediate host appears to be the client. In this assignment, the intermediate host will not change the packets, it will just send them on. The intermediate host could be updated to change packets and thus become an error simulator for the system.

Note that this assignment requires knowledge of the material presented on UDP/IP and Java's DatagramPacket and DatagramSocket classes, as well as conversion between Strings and arrays of bytes. It includes some very basic knowledge of UML Class and Collaboration Diagrams and basic UCMs, but does not require any knowledge of Java threads or TFTP.

Specification

The client algorithm is:

- the client creates a DatagramSocket to use to both send and receive
- repeat the following 10 times:
    - the client creates a DatagramPacket
        - the packet is either a "read request" or a "write request" or an invalid request (suggestion: alternate between read and write requests with at least one invalid request):
            - read request format:
                - first two bytes are 0 and 1 (these are binary, **not** text)
                - then there's a filename converted from a string to bytes (e.g. test.txt)
                - then a 0 byte
                - then a mode (netascii or octet, any mix of cases, e.g. ocTEt) converted from a string to bytes
                - finally another 0 byte (and nothing else after that!)
            - write request format:
                - just like a read request, except it starts with 0 2 instead of 0 1
    - the client prints out the information it has put in the packet (print the request both as a String and as bytes)
    - the client sends the packet to a well-known port: 68 on the intermediate host
    - the client waits on its DatagramSocket
    - when it receives a DatagramPacket from the intermediate host, it prints out the information received, including the byte array

The intermediate host algorithm is:

- the host creates a DatagramSocket to use to receive (port 68)

- the host creates a DatagramSocket to use to send and receive
- repeat the following "forever":
    - the host waits to receive a request
    - the host prints out the information it has received (print the request both as a String and as bytes)
    - the host forms a packet to send containing exactly what it received
    - the host prints out this information
    - the host sends this packet on its send/receive socket to port 69
    - it waits to receive a response
    - it prints out the information received
    - it forms a packet to send back to the host sending the request
    - it creates a DatagramSocket to use to send this request
    - it prints out the information being sent
    - it sends the request

The server algorithm is:

- the server creates a DatagramSocket to use to receive (port 69)
- repeat the following "forever":
    - the server waits to receive a request
        - the packet should be either a "read request" or a "write request" (see details above)
        - the server should parse the packet to confirm that the format is
            - 0 1 or 0 2
            - some text
            - 0
            - some text
            - 0
            - nothing else after that!
    - the server prints out the information it has received (print the request both as a String and as bytes)
    - if the packet is not valid (as per the above), the server sends back 0 5 (exactly two bytes)
    - if the packet is a valid read request it sends back 0 3 0 1 (exactly four bytes)
    - if the packet is a valid write request it sends back 0 4 0 0 (exactly four bytes)
    - the server prints out the response packet information
    - it then creates a DatagramSocket to use just for this response
    - and sends the packet via the new socket to the port it received the request from
    - and closes the socket it just created

The Echo Client-Server example discussed in class, as well as the APIs for the DatagramSocket, DatagramPacket, and String classes, as well as information on Java arrays which are available through the http://java.sun.com help facility, may prove useful.

Work Products

1. A “README.txt” file explaining the names of your files, IDE used, set up instructions, etc.
2. One UCM showing the client, intermediate host, and server.
3. Three UML Collaboration diagrams, one each for the client, intermediate host, and server.
4. One or more UML Class diagrams showing your system.
5. The source code for all three parts of the system, as well as any files required to run these files in the Java IDE chosen. (You may submit test classes if you wrote them, but be sure to explain the files you have submitted – see #1 above.) Your code should demonstrate good programming style, and be well

documented, etc. For examples of "industrial quality" Java code, have a look at Sun's Java coding conventions, which can be found on our Java resources Web site. (This site can be reached by a link from the SYSC 3303 Web site.) For parts 2 and 3, hand-drawn scanned diagrams are acceptable, as long as they are neatly drawn and your handwriting is legible, and the software required to view them is present in the lab.

Hints

For this assignment and for the project you need to be able to run multiple main programs (projects) concurrently. Ensure that your IDE is configured correctly. See the course reference material for more information.

Don't get mixed up between text and bytes. For example, the requests start with a 0 byte, **not** the character 0 converted to a byte.

Reminder

The TAs will mark your assignments in the lab. It is your responsibility to ensure that your code works in that environment, and that any software required for viewing any text/diagrams is also present in that lab. You may use any Java IDE available in the lab (e.g. JCreator or Eclipse).

Submitting Assignments

Assignments are to be submitted electronically using the assignment “submit” program. Emailed submissions will not be accepted. See the course outline for the procedure to follow if illness causes you to miss the deadline.

Due: Saturday, May 11th at 8pm SHARP!