

CS/ECE 757 – Computer Networks Fall 2004

Assignment 4: Programming with HyperCast Overlay Sockets

Instructions:

- Complete the assignment individually. You may discuss the homework assignment with anyone in class, but you must submit your own program!
- The only help you are allowed are the material available from www.cs.virginia.edu/hypercast, the online documentation at www.javasoft.com or www.sun.com, the material on the web which is listed on the CS 757 website, and a book on the Java programming language.
- The grade for this project is weighted as follows:
 - 50% Implementation
 - 20% Functional Tests
 - 30% Write-up

Due Date: November 11, at the beginning of class.

Objective: Implement an inverse multicast (many-to-one) application.

Background and Preparation:

- This homework requires programming with the HyperCast overlay socket.
- Before starting with the assignment you must download the HyperCast 2.0 software.
 - Go to: <http://www.cs.virginia.edu/~mngroup/hypercast/download.html>
 - Download the user manual and follow the installation guide.
 - Run the Whiteboard demo application described in Section 2.3 of the user manual. (Downloading, installing and running the whiteboard application should take about 15 minutes.)
 - Follow the instructions in Section 5 of the user manual to compile and execute the “HelloWorld” program.
- **You may also download a more recent version of HyperCast (version 3.0), however, the available documentation for this version is limited. Please contact the instructor if you are interested in working with a more recent version.**

Hints:

- HyperCast 2.0 requires the Java Development Kit (JDK) version 1.2 or higher.
- Use the “HelloWorld” program as the basis for your programming assignment.
- The following parts of the Hypercast website will be particularly helpful:
 - a. Javadoc of the API: <http://www.cs.virginia.edu/~mngroup/hcast/javadoc/index.html>

(Specifically:

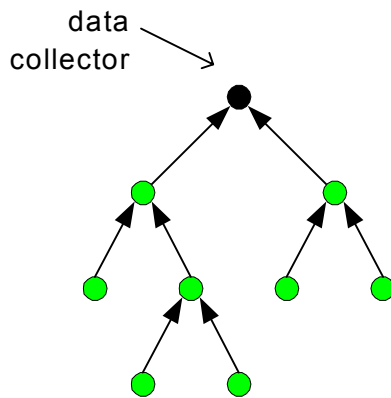
http://www.cs.virginia.edu/~mngroup/hcast/javadoc/edu/virginia/cs/mng/hypercast/I_OverlaySocket.html)

b. Description of the API:

<http://www.cs.virginia.edu/~mngroup/hypercast/designdoc/API+Example/api.htm>

Problem Description:

- Create a Delaunay triangulation overlay network with N nodes.
- One node in the overlay network is the **data collector**, and the other N-1 nodes are **data reporters**.
- The logical address of the data collector is (12345, 67890) and the logical address of the data reporters are selected randomly. This is done by setting the attributes in the properties file (“hypercast.prop”) to `DT2-0.Coords = 12345,67890` and `DT2-0.Coords = RANDOM10000`, respectively.
- The data reporters send messages to the data collectors. The messages are sent upstream in a tree that has the data collector as the root.
- Refer to the figure below. The figure shows a tree. The root node is the data collector and the other nodes are the data reporters. Data reporters are leaf nodes in the tree if they do not have children. Otherwise, the data reporters are intermediate nodes.



- Each data reporter sends one message every second to its parent node in the tree with the data collector as the root (see figure). This is done by sending an overlay message
`Socket.SendToParent(msg, LACollector),`
where **msg** is the message, and **LACollector** is the logical address of the data collector.
- The payload of message **msg** contains the logical address of the data reporter and a timestamp as plain text. For example, a message could be of the form:
10023,8222 10:23.45

Here, (10023,8222) is the coordinate and 10:23.45 is the timestamp.
The timestamp should have a granularity of a second.

- The data collector displays each received message. Each message is displayed in a new line.
- **(Optional extension):** In this extension, the intermediate nodes can merge multiple messages. Here, each node is allowed to send at most 1 message per second. If an intermediate nodes receives messages from its child nodes, it buffers the payload of these messages. When the node wants to send its own message (after the second is up), it concatenates these messages into a single

message and sends them upstream. When the data collector receives a message that has been concatenated, it displays all coordinates and timestamps from that message in a single line. For example:

```
10023,8222    10:23.45 | 3242,9483    10:24.83 | 917,2342    10:22.83
```

Requirements:

- The data reporters and data collector must be able to run on different machines on the Internet.
- The application must be able to run with at least $N=100$ nodes.
- The application must permit that data reporters join and leave while the application is running.
- You receive 20% extra credit if you complete the optional extension.

Your Task:

1. Implement the data collector and the data reporters using the HyperCast 2.0 overlay sockets.
2. Test your client implementation with the server specified in the requirements.
3. Deliver a max 2 pages write-up which describes your implementation approach, a description of your tests, and known bug list.

Submission Guidelines:

- Your submission includes the write-up, the source code files of your program, and max. 2 pages of output from the data collector. (The output should be from a test of the application with 10 clients.)
- You will submit your homework electronically using the guidelines posted on the website.
- You will submit also submit a hardcopy of your write-up and your source code.