CS/ECE 757 – Computer Networks
Fall 2003

Project
A lookup service for Delaunay triangulation overlays

Instructions:

- This is a group project. You can work in groups of 3 or 4 students. (No groups of 1 or 2 student(s)).
- The grade for this project is weighted as follows:
  - 50% Report (Design, Write-up, Experiments)
  - 30% Code and User manual
  - 10% Test and Demo
  - 10% Intermediate Reports and updates
  - Extra credit: up to 20% (for features of the system)

Objective: Design and implement a lookup service for the HyperCast overlay sockets.

Description:
The basic operation of the lookup service is the insertion, lookup, and deletion of (key, value) pairs. Each file is associated with a (key, value) pair, where the key is a (x,y) coordinate and value is the location of the file (e.g., a URL). (Key, value) pairs are stored at servers, called nodes. Each node has an (x,y) coordinate. The coordinates of nodes are selected randomly from a 10000x10000 grid. The key of a file is obtained by using a hash operation on the file name, the result of the hash function is an (x,y) coordinate in a 10000x1000 grid. In Figure 1, nodes are indicated by circles and files are indicated by rectangles.

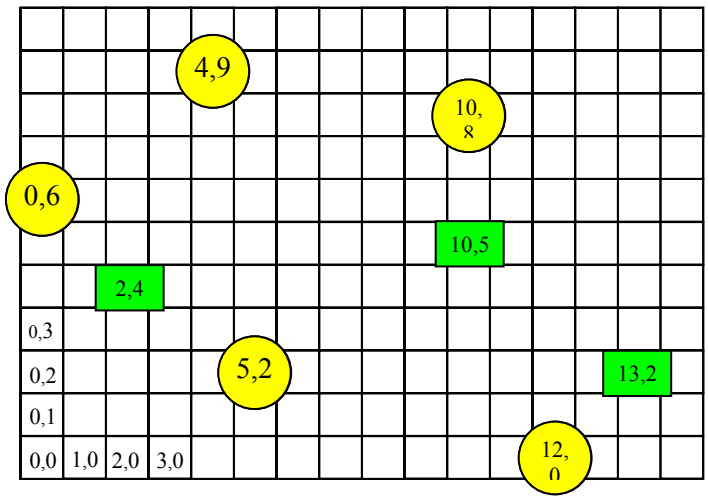

Figure 1.

The nodes establish a Delaunay triangulation overlay network. A (key, value) pair is stored at a node if the (x,y) value of the key lies in the Voronoi region associated with the nodes. Recall from the lecture on Delaunay Triangulation overlays that Voronoi regions and the Delaunay triangulation are closely related (http://www.cs.virginia.edu/~cs757/slidespdf/757-11-DT.pdf).

The Voronoi region for a node is the region of the plane that is closer to the node than any other node. To create a Delaunay triangulation from Voronoi regions, connect nodes that have neighboring Voronoi regions by an edge. To create Voronoi regions from a Delaunay triangulation, draw the perpendicular bisection for each edge of the Delaunay triangulation. Figure 2 shows the Delaunay triangulation and the Voronoi regions for the nodes in Figure 2.

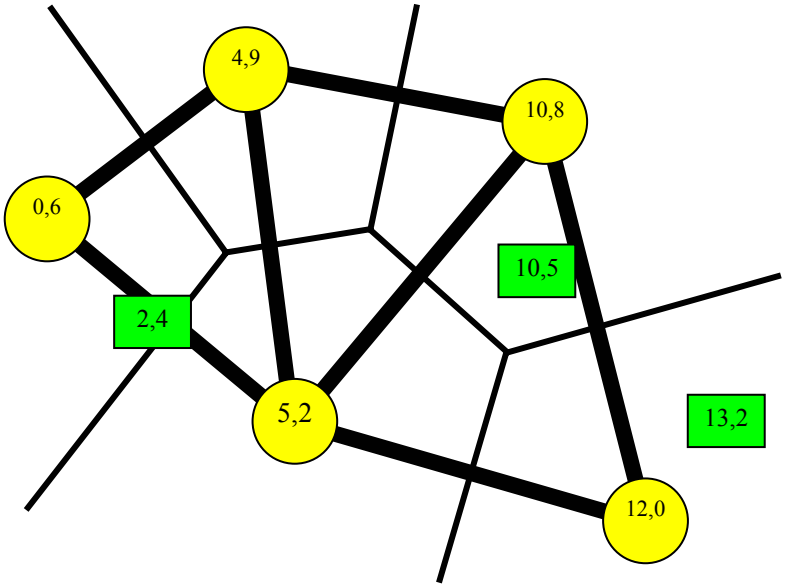

Figure 2.

The lookup service supports the following operations. These operations should be provided as an application programming interface (API):

Insert operation: The Insert(key, value) operation can be started from any node. The local node tests if the key is in the Voronoi region of the node. If no, the (key, value) pair is forwarded to a neighbor, whose (x,y) coordinate is closer than that of the local node. If yes, the (key, value) pair is inserted locally.

Delete operation: The Delete(key) operation can be started from any node. The local node tests if the key is in the Voronoi region of the node. If no, the (key, value) pair is forwarded to a neighbor, whose (x,y) coordinate is closer than that of the local node. If yes, the node deletes the local (key, value) pair (assuming it exists).

Query operation: The Query(key) operation can start at any node. The local node tests if the key is in the Voronoi region of the node. If no, the (key, value) pair is forwarded to a neighbor, whose (x,y) coordinate is closer than that of the local node. If yes, the node looks up the local (key, value) value and returns the (key, value) node to the node where the query was started.

Further reading:

- There is a lot of information on Voronoi regions available at www.voronoi.com.
- Information on the HyperCast software is available from www.cs.virginia.edu/hypercast.

Your Task:

1. Design a lookup service for Delaunay triangulation overlays as described above. Your design should be able to deal with:
    a. Multiple nodes joining and leaving at the same time. The lookup system should not loose (key, value) pairs if a node is leaving.
    b. The system must be able to deal with deletions, insertions, and queries running at different nodes at the same time.
    c. Robustness to failure of nodes. Even when some nodes fail (temporarily or permanently), queries should be handled successfully. (Note: This may require to store the same (key, value) pair at multiple nodes.)
    d. High Performance. The lookup system should be able to deal with a very large number of (key, value) pairs with a very large number of simultaneous queries. An important performance measure is the time it takes to complete a query.
    e. Scalability: The time to complete an insert, deletion, or lookup operation should be reasonable even when the system is large.
    f. Usability: The system should have an intuitive application programming interface.
    g. Extra features: Design, test, and evaluate additional features of your system. For example, you can provide access to actual files, such as a file sharing system with a graphical user interface.

2. Implement the lookup service as an application program using HyperCast overlay sockets. Prepare a user manual that discusses how to use your program.

3. Evaluate your system in measurement experiments. Provide graphs that show the results of your performance measurements. Conduct experiments that measure the robustness, performance, and scalability of the implemented system. (Example experiments can include: How many (key, value) pairs can be stored at a single server? How does the time to complete a query change with the number of (key, value) pairs and the number of nodes?).

The experiments must be run on multiple machines (Avoid the departmental interactive servers: mamba, viper, cobra). You can obtain access to the PlanetLab testbed (www.planet-lab.org) to run wide-area experiments (You need to talk to the instructor if you need to gain access).

4. Write a report, that describes your design, your implementation, and your experiments. The report should be written like a research paper (see the CAN, CHORD, and other papers on the web site). The length of the report should be 15-25 pages.

5. Demonstrate your working system to the instructor and discuss your report.

Schedule:

- You need to send weekly updates on the progress of your system to cs757@cs.virginia.edu. Include an updated schedule for completing your project. The update should have a minimum length of 1 paragraph. The updates are due on Fridays (5pm).
- You need to have at least 2 meetings with the instructor about your project. One meeting should be scheduled during the design phase, and one meeting should be scheduled during the experimentation phase. You are responsible for scheduling the meetings. Schedule the meetings by sending email to jorg@cs.virigina.edu.
- Milestones:
    - **Oct 28:** Complete forming of teams (send names of group members to cs757@cs.virginia.edu).
    - **Nov 7:** Submit a design document (by email) that describes your design, your evaluation plan and the planned extra features.
    - **Nov 21:** (**Checkpoint only**) By this time, most of the implementations should be completed. Please report the status of your implementation in your weekly update.
    - **Nov 28**: (**Checkpoint only**) Experiments should be completed. Please report the status of your experiments in your weekly update.
    - **Dec 5:** Due date for project. **Please submit the complete software and report as a CD to Prof. Liebeherr's mailbox.**
    - **Dec 8-10**: Schedule a 30 min demonstration of your system.