Object-Oriented Programming in Java

Problem Set 3 — Due: Jan 17, 2001

Group Project: Gnutella

Gnutella is a peer-to-peer networked file sharing program. Its functionality is similar to that of Napster from the user's point of view, except that it is not specific to audio files, and the index and search capabilities are not centralized as in Napster, but distributed among all the Gnutella nodes.

Each Gnutella program is both a client that can initiate searches and download files, and a server that can upload local files and perform searches on behalf of clients. The term *servent* is often used to describe this sort of client-server combination (Note that this is different from `servant` in both spelling and connotation).

Each Gnutella servent running on an individual machine is connected via the network to some small set of other servents (also called nodes to emphasize their being part of a network graph). This interconnected set of servents is the Gnutella network. The nodes of the network work together to operate as one large search engine.

The properties that make Gnutella an suitable project for this class are:

- Functional requirements that demand network, stream IO, GUI, and multi-threaded programming techniques,
- Sufficient complexity for a challenging short-term group project, without being overwhelming,
- An operational network architecture,
- A published and well-defined protocol specification to implement,
- A collection of reference clients to test against,
- A number of Web sites devoted to it provided documentation and reference implementations, and
- An active network to connect to once the project is complete.

Gnutella Servent Operation

When the Gnutella servent is started, it is configured with a directory on the users machine containing files to share, a directory to store downloaded files, and the IP address of a Gnutella node to initially connect to. Once the servent program is connected, the user can initiate keyword-based searches. The servent forwards search requests onto the network and displays results as they filter back. Unlike a centralized search engine like Google, a Gnutella node checks an incoming query against the files it is sharing and then forwards the query to the other nodes it is directly connected to. Also unlike Google, search is based only on file name, not content.

When a search is successful, the user can request to download any of the files matching the search. This causes the Gnutella servent to directly connect to the node sharing that file and request a download. If the request is successful, it downloads the file, displaying the progress of the download as it proceeds.

While it is operating, the Gnutella servent is also functioning as a server: matching query requests against the files it is sharing, uploading requested files to remote nodes, and forwarding search requests through the network.

The Gnutella Network

The nodes in the Gnutella network communicate via the Gnutella protocol, which defines the behavior of a Gnutella servent. Because the Gnutella servents are connected via this protocol, the individual servent programs do not have to be the same. Any program that implements the Gnutella protocol can be a legitimate node in the Gnutella network. In fact there are several servents that are available on the Internet, including several for Windows and two written in Java.

The Gnutella network is actively used on the Internet and there are several thousand nodes active at any given time. The distributed nature of the Gnutella network and its search algorithms make it an interesting study in network architecture. It has shown some interesting effects as the network has scaled in size. The architectural issues, however interesting, are beyond the scope of this class (there are several papers on the Web discussing these issues if you are interested).

Resources

A copy of the Gnutella protocol specification is available on our Web site. (Note: This PDF is apparently only viewable from acroread.)

The master site for information on Gnutella is `http://gnutella.wego.com/`. This site lists pointers to other sites with additional information and/or clients. A Web search on "Gnutella" will turn up more information. Use the Web and the attached protocol document as your main sources of information for this project.

The staff will set up a number of Gnutella servents downloaded from the Web on the TA machines. These servents can be used to test against as well as for UI inspiration.

Servent Functionality

The functionality we expect you to support includes:

- Full implementation of the network protocol as both a client and server, sufficient to be a functioning node in the Gnutella network.
- Maintenance of user configuration information, such as node to connect to on startup, directory to share for upload and share, directory in which to download files. This configuration should be kept in a file and be persistent between program executions.
- The ability to connect to a set of nodes on startup.
- A GUI that allows the user to change the configuration info, connect to a node, initiate a search, display the results of a search (a list of files with associated node data), download a file from a search result, and monitor who is uploading what from this node.
- A certain amount of security on the file sharing. In particular, that the servent does not permit access to files outside of the upload directory. (Be careful of paths such as `../../yourfile` which will allow outside access if you are not careful).

Plan

This is a major implementation effort involving several people. It is an exercise is group organization and project management (without an actual manager!) as well as a programming challenge. We highly recommend following these steps.

Step 1: Take a deep breath, stop, and think.

Step 2: Read the protocol spec thoroughly. Read any documentation available on the Internet, and check out the reference clients.

Step 3: In your groups, develop your design from the top down. Decide first on what the major modules or sub-systems are and how they communicate. Define the interfaces between them and what data is shared. Document these decisions.

Step 4: Decide how the project is to be divided among the group members. It could be one sub-system per person, or multiple people working together on a single piece (the proponents of Extreme Programming believe in pair programming; two people working together on the same code, one typing and one watching, to check each other's errors and design decisions).

Step 5: Decide on how the group is going to coordinate and share files, data, and testing.

Step 6: Come up with a plan for testing each sub-system independently and a plan for testing the entire program.

Step 7: Each sub-group should then design their sub-system, again top down. Following the steps from PS2, define the major classes and interfaces, and the methods by which they interact. The design the data representations and any utility classes and methods. Document these decisions. You should have substantial Javadoc written and working before actually implementing any methods.

Step 8: Finally, start implementing bottom-up, testing each new set of methods and each new functionality as it get developed. Consider an implementation strategy that would allow partial sub-system testing and early testing of the integration of these into the overall program in order to validate your design.

Step 9: Complete sub-system testing and combine into the complete program.

Step 10: Test the program. Connect with other nodes in ADU and search for interesting files to download. Host some files on your machine (keep it legal, please) for others to upload.

Step 11: Once you have verified your program against the reference client (and those of other groups), try to connect to the Gnutella network outside of ADU.

Good Luck