EE4210 Computer Communication Networks II

Project (30% towards module grade)

Due on 21 Apr 2013 (Sunday), 2359 hrs

Learning Goals:

1) Hands-on exercise with socket programming, and multithreaded programming.
2) Experience the design and implementation of a peer-to-peer application.

Grading Criteria:

Implementation (50%)
Functional tests (25%)
Written report (25%)

Problem Description:

You are required to implement a simple peer-to-peer file synchronization application. The general idea is as follows. When a peer establishes a TCP connection with another peer, both of them will compare the lists of files they possess (you may assume that these shared files are stored in a designated folder in each peer, hereafter called the "SharedFiles" folder), and proceed to exchange files so that they both have exactly the same files. A TCP connection must remain open until a user manually terminates the peer process at either end of the connection. During the time when the connection remains open, any new file addition to the "SharedFiles" folder of any peer must be propagated to the other connected peers in the overlay network.

Requirements:

(i) A peer will only send to another peer those files that the latter does not have, so that no resources are wasted on transmitting duplicate files.
(ii) A peer must be able to connect to multiple peers simultaneously; all connected peers in the overlay network must eventually have the same files. For example, if A, B and C form an overlay network with a chain topology A-B-C, then A, B and C must end up having exactly the same files.
(iii) A peer must provide an interface (either text or graphical user interface) that allows the user to specify the IP addresses of other peers that it should initiate connections with. For example, for the chain topology A-B-C, one possible way to create this topology is to let A initiate a connection with B, and let B initiate a connection with C. Since a TCP connection is two-way, there is no need for B to initiate a connection with A, nor the need for C to initiate a connection with B.
(iv) A peer process must not crash or hang if a neighboring peer in the overlay network is terminated abruptly.

Assumptions:

To simplify the problem, you may assume the following:
(i) Files with the same filename have identical contents; there is no need for you to verify that they are indeed the same file.
(ii) The users will not initiate connections in such a manner that would result in a "looping" problem within the overlay network.
(iii) The IP addresses are known beforehand, so there is no need for an index server.
(iv) A user will not delete any file from the "SharedFiles" folder of a peer as long as the peer process is running.

Application-layer Protocol Design:
You are required to design your own application-layer protocol, such as the types of messages exchanged (e.g., request, response), the message syntax (i.e., the fields in each message type), the message semantics (i.e., how the information in each field is to be interpreted), and the rules for sending and responding to messages, etc. Make sure that you describe your design in detail in your written report.

Project Submission Deadline:
Due on **21 Apr 2013 (Sunday)**, **2359 hrs**.
Note that 10% marks (i.e., 3% from your module grade) will be deducted for every day it is late. Hence, you are strongly encouraged to start early (instead of a "slow start").

Deliverables:
Zip all files into a single file, name it using your matric number, e.g., U0123456A.zip, and upload it into IVLE Workbin under "Project Submission" anytime before the deadline. You must include the following:
(i) Program source codes (not binary executables). The codes must be commented properly.
(ii) A written report, in PDF format. You may use your own discretion for the number of pages. Your report must describe your application protocol design in detail. You should also describe the challenges that you'd faced in implementing the project, and any bugs that you cannot solve. If you have implemented any extra features that are noteworthy, please highlight them in a separate section.
(iii) A brief "readme" file (.txt or .doc/.docx) that explains how your code can be compiled, as well as the commands and arguments needed for running your programs. You should also provide some command line examples.

Optional: You may also do a screen video recording with voice narration to demo how your application works. Convert your video into a file format such as .flv which is known to result in smaller file size. Zip your video file together with the rest of your files into a single .zip file as described above. Note that the maximum upload size for IVLE is 200MB. Also, note that no extra credit will be awarded for creating such a video; the intention is merely to allow the opportunity for you to explain your application more clearly.

Academic Integrity:
Plagiarism will not be tolerated. You are **not** supposed to share any code with any classmate. You **may** discuss the project requirements or your solution strategies at a high-level, without sharing details at the code level. We do **not** distinguish between those who copy others' work, and those who allow their work to be copied. If you are involved in plagiarism, you will be given 0 mark for the project, and referred to the University for disciplinary action.

Programming Language:
You can use Java, Python, C, or C++ for this project.

Some Helpful Resources:

(i) Some code examples in the IVLE Workbin, under "Project Materials".

(ii) http://www.schaik.com/download/sockets.html, "Web Programming with Sockets". This website provides code examples for both Visual C and Java. You will see how a simple Web application (client + server) can be implemented.

(iii) http://netbeans.org/index.html, "NetBeans Integrated Development Environment". It is a free, open-source Integrated Development Environment for developing software. It supports Java, C/C++, and more. You are recommended to make use of this tool.

(iv) http://www.netbeans.org/kb/index.html, "Documentation for NetBeans IDE".

(v) http://java.sun.com/javase/downloads/index.jsp, "Java SE Development Kit".

(vi) http://docs.oracle.com/javase/tutorial/index.html, "The Java Tutorials".

(vii) http://java.sun.com/docs/books/tutorial/getStarted/cupojava/index.html, "Creating "Hello World!" with Java".

(viii) http://www.python.org/about/gettingstarted, "Getting Started with Python".

(ix) http://docs.python.org/3/tutorial, "The Python Tutorial".

Good Luck & Work Hard!