

**CS 5480/6480: Computer Networks – Fall 2006**  
**Programming Assignment 1**  
**Due by 5:00 PM on Thursday September 28th 2006**

**Important:**

- **No cheating will be tolerated.**
- **Late submissions are docked 10% per day late, unless approved by the professor in advance.**

**Total Points for this homework: 100**

The goal of your programming assignment is to build a very *simple peer-to-peer system* involving three nodes using *sockets*. You need to write programs that run on three nodes and achieve the following tasks.

- Elect a node among the three as the leader. The leader node is a peer but keeps track of the files available on the p2p network together with the IP addresses and port numbers from where they could be obtained.
- Each non-leader node sends information about the files it has and the port numbers from where they could be obtained to the leader. When a peer requires a file, it queries the leader for the file and if the peer has the information about the file, it replies back with the filename, IP address and port number.
- Once a peer finds out the IP address and port number from where a desired file could be obtained it sets up a connection to that IP address and port and obtains the file.

Each node runs *two* programs. The first program carries out the following.

**Leader Election:** Assign identification numbers to each node, setup TCP connections among the three nodes, exchange the identification numbers and choose the node with the lowest identity as the leader. The other two peers should now remove any connections they have between themselves. In setting up the TCP connections you might have to order the *server* and *client* functionality across the three nodes.

**Inform the leader about files that peers are willing to share:** These peers then send the information about the files they want to share to the leader especially identifying the IP address and port number from which the files could be obtained.

**Query the leader for a required file:** One of the two non-leaders sends a query to the leader seeking a file that is actually available at the other peer (but the peer initiating the query does not know that). The leader replies back with the filename, IP address and port number. If the filename is unknown to the leader, it replies with a “file not found” code (you could pick any form of indication from the leader to the peer).

**File transfer:** The peer initiating the query establishes a connection with the IP address (of the other peer) and port number it obtained in the query response and sets up a TCP connection with the other peer and transfers the files and saves it in its local disk.

The second program is a simple server that responds to clients request for a file.

Follow all the guidelines provided in the document on grading policy for programming assignments that is available from the class web page. For this assignment, let each peer have three files that it wishes to share. Print the consolidated list of files and their source IP addresses and ports that are available at the leader after the peers send that information to the leader. Show one unsuccessful and one successful query followed by one successful peer-to-peer file transfer.

You should check the return code of all the socket calls and print error messages when calls fail.

## Grading Policy for Programming Assignments

The program(s) you hand in should work correctly and must be well-documented. You should submit your programming assignment *electronically* using the *handin* command. Your submission should include the following:

1. The entire code containing in-line documentation.
2. A separate document of a page or two (at most) describing the overall program design, a verbal description of “how it works”, and design tradeoffs considered and made. Also describe possible improvements and extensions to your program (and sketch how they might be made).
3. A separate description of the tests you ran on your program to convince yourself that it is indeed correct. Also describe any cases for which your program is known not to work correctly.
4. **A plain text *readme.txt* file describing how to run your program(s).**
5. The executable files of your program(s).
6. A plain text file *output.txt* containing sample output as required by the assignment.

## Grading

### *Program Listing*

works correctly	60 points
in-line documentation	10
quality of design	10

### *Design Document*

description	5
tradeoffs discussion	5
extensions discussion	5

*Thoroughness of test cases* 5

*Total* 100 points

**Note:** A full 10 points for quality of design will only be given to well-designed, thorough programs. A correctly working, documented and tested program will not necessarily receive these 10 points.

## Electronic Submission - Turning in files using handin in the CADE

All programming assignments must be submitted on CADE machines using the *handin* command. To electronically submit files while logged in to a CADE machine, use:

```
% handin cs5480 assignment_name file_1 file_2 ...
```

where `cs5480` is the name of the class account (same for both `cs5480` and `cs6480` students), and `assignment_name` (`PA1`, `PA2`, or `PA3`) is the name of the appropriate subdirectory in the `handin` directory. Use `PA1` for this assignment.

The CADE also provides a web-based facility for turning in files electronically:  
<<https://cgi.eng.utah.edu/webhandin/index.cgi>>

## Some other points from previous offerings of this class

- Every programming assignment of this course needs to be individually done by a student. No teaming or pairing is allowed.
- You need to program in one of the following three languages: C, C++, or Java
- Linux is the preferred OS, although Windows platform is acceptable.
- You could assume that the peers know the IP address and port numbers of other peers for establishing TCP connections. However, for file transfer this information is obtained from the leader node.
- *readme.txt* – This file must contain compilation and running instructions detailing how the TA should go about running your program on three machines (is there any specific machine IP you have hard-coded, is there any specific order in which the programs have to be started, etc).
- Submit ALL your files - This includes data files which you have used for testing the file transfer.