

COMP 4704 – Spring 2009

HW#4

Instructor:

Prof. Matt Rutherford – mjr@cs.du.edu

Due by:

5pm on April May 7, 2009

Please choose **ONE** of following two options. The assignments should be emailed to me at mjr@cs.du.edu before the due date.

Option #1: Decentralized P2P Implementation

The goal of this programming assignment is to implement a simple P2P program that is capable of using the Push, Pull, and Push/Pull algorithms for building up random connections between peers.

You should create a single program that takes the following parameters (feel free to have defaults for each option):

- **Algorithm** – a flag indicating either the *push*, *pull*, *push/pull* algorithm should be used for swapping information with peers.
- **Peer Count** – an integer representing the number of peers to keep track of.
- **Timeout** – an integer representing the average delay between each iteration of the algorithm. This can be in seconds.
- **Address/Port** – the host/port combination that the node will listen on for incoming connections. This will also be used as the unique identifier of the node.
- **First Peer** – the host/port of the first peer that the node knows about.

You can assume that all peers that are connected to each other are running with the same algorithm and peer count.

The program will basically perform two tasks:

1. Listen for incoming requests and process them according to the algorithm selected.
2. Initiating connections with randomly selected peers from the internal list and interacting with the remote host as dictated by the algorithm selected.

Note: You will need to decide on a message format / protocol that is used between instance of the peers. I would strongly advise using an ASCII encoded, newline-delimited scheme. This will be relatively easy to implement, and you can also interact with a program through telnet (by specifying the port to connect to explicitly) or similar programs.

Note2: The timeout parameter should be the “average timeout”, meaning that you could randomly sample around that number for each delay so that you don’t have all of your peer programs in lockstep.

Deliverables

- A ZIP file containing only source code, no binary or compiled files. This should include:
 - A `README.txt` file that describes the command-line interface to your program and how to compile / run if it is not obvious.
 - The source code for your program.
 - A simple test script / program that will start 5 instances of the program and run to completion.

Please try not to send elaborate directory structures with your code, if everything exists in the same directory (i.e., the top-level one) it makes it much easier to grade.

Option #2: Textbook Problems

- Chapter 5, Problem #6
- Chapter 5, Problem #9
- Chapter 5, Problem #13
- Chapter 5, Problem #17
- Chapter 5, Problem #18

Deliverables:

- A document containing detailed (one to two paragraph) answers for each of the questions.