CS 6378: Advanced Operating Systems
Programming Project 2

Instructor: Ravi Prakash

Assigned on: July 9, 2023
Due on: July 30, 2023

You are required to work on this project alone. Sharing of code with fellow students, or using code from other sources without permission of the instructor and/or TA is strictly prohibited. You are expected to demonstrate the operation of your project to the instructor or the TA.

Project Description

Peer-to-Peer Systems

A peer-to-peer (P2P) system, for the purpose of this assignment, consists of a set of autonomous computers that act as a distributed repository of files, and allow participating computers to search for and download files from each other.

The model to consider for this assignment is the following:

1. Each computer, C_i, maintains a file, F_i that contains a list of files (and associated keywords) that are being made visible to the outside world. Only the files in F_i can be downloaded from C_i by other computers.
2. When a computer, C_i, wishes to join the P2P system it is assumed that it knows of at least one other computer, C_j, that is already part of the P2P system. If C_i is the node that initiates the P2P system then C_i knows that, too.
3. In order to join the P2P system, computer C_i sends a join request to C_j. Both C_i and C_j add each other to their list of neighbors and establish a TCP connection between themselves.

File Sharing Procedure

You have to implement the following simple search approach:

Simple Search Approach: If computer C_k is looking for a file, it can issue the search request using either a file name or a keyword, and a *hop-count* which is initially set to 1.

1. The search request is flooded into the overlay network (along the links represented by the list of neighbors) to other computers that are no more than *hop-count* away from the computer issuing the search request. Having initiated the search request, C_k starts a timer which is set to expire after t_{hop_count} seconds.
2. You must ensure that a computer forwards a search request issued by another computer at most once, *i.e.*, your implementation should be able to detect duplicate requests.
3. Let there be a computer C_x that has a file, listed in the corresponding F_x, whose name matches the specified file name, or whose keyword matches the specified keyword. Then, C_x sends a response to the neighbor from which it received the first copy of the search request. The response contains C_x, and the keyword and file name of the matched entry.

4. If a computer that initiated the search request (C_k) receives a reply, it *consumes* the reply.

5. When a computer that did not initiate the search request (C_l) receives a reply, it forwards the reply to the neighbor from which it received the first copy of the corresponding search request.

6. As part of *consuming* a reply, the search initiator, C_k, collects all the replies received until the expiry of the timer, and then displays them as a list of tuples of the form *(kwd, file_name, comp_name)*. Replies received after the expiry of the timer are ignored.

7. Users, on C_k, that made the request can select which response tuple they are interested in.

8. If the selected tuple corresponds to a file located at computer C_m then C_k establishes a TCP session with C_m and copies the file from C_m to its own directory, and updates F_k accordingly to indicate that the newly copied file is available for sharing. Once the file has been copied from C_m, the TCP session established with C_m for the purpose of obtaining the file is terminated.

9. If the search terminated without success (no reply received before the timer expired), then C_m should double the hop-count and reinitiate the process. This should be repeated until there is success, or the hop-count exceeds sixteen (whichever happens earlier). Note that the timer value is a monotonically increasing function of hop-count.

Your Assignment

Your task is to implement the P2P system described above.

1. Your implementation should make sure that state information about a search in progress is not maintained at the participating nodes for an unduly long period of time.

2. When a computer wishes to join the P2P system it should randomly select one of the computers that is already part of the P2P system as its neighbor and issue a join request.

3. Your P2P system should start with one node, and slowly grow to fifteen nodes. You must choose from the machines set aside for network programming (dc01 - dc45). No two computers that you simulate in this exercise should execute on the same machine.

4. Having grown to fifteen nodes, you should initiate departure of nodes, one at a time. If a departing node has only one neighbor then all that node has to do before departing is to terminate its TCP session with its neighbor and all ongoing TCP file transfer sessions. If a departing node has two or more neighbors then before departing it should arbitrarily select one of its neighbors, n_1, and make it a neighbor of all other neighbors (provided the two nodes are not already neighbors of each other). Then, it should terminate its TCP sessions with all its neighbors as well the ongoing TCP file transfer sessions.

5. For the sake of this exercise you can assume that two or more nodes do not decide to depart concurrently.

Data Collection

For each successful search report the following:

1. The chosen hop-count that resulted in at least one successful reply being received by the initiating computer.

2. The elapsed time between the initiation of the search and the reception of each of the replies by the initiator.

Submission Information

Please submit your files (source code, makefile, log files) as one "zipped" or "tarred" archive using eLearning. The report should contain your measurements for the values in the configuration file provided to you.