EEL6871 Autonomic Computing Fall 2013 **Homework 1**
Assigned: Thursday, Aug-22-2013
Due: Monday, Sep-2-2013 5pm

Topology Management in Peer-to-Peer Systems

Requirements

In this homework, you are required to write a C/C++/Java/Matlab program to implement the method for Topology Management of Overlay Networks described in Section 2.2.2 of Andrew S. Tanenbaums Distributed Systems book (reading supplement 1). This algorithm is also known as Jelasity and Babaoglus algorithm. In this algorithm, every node in the network maintains a list of neighbors. During the network-initialization phase, each node randomly selects k neighbors and places them into its neighbor list. During the network-evolution phase, in each cycle of the iterative algorithm, every node randomly selects one of its neighbors, and then sends a list consisting of the identifiers of its neighbors and of itself to that neighbor. The selected neighbor also sends its neighbors list back to the node which initiated the action. Upon receiving the new neighbor list, the nodes select the nearest k nodes from both the new and the old lists as their neighbors and discard all the others.

This homework includes two tasks. First you will write a program to generate a fixed-radius ring topology (radius=1) using the above-mentioned algorithm with the value of the radius r being an input integer. Second, change the program so that r is reread every 5 iterations. Your program should adapt (i.e. changing the old value of r to the new value of r) in order to generate the ring whose radius is the last input value of r. However, this adaptation can only be done by incrementing the radius by one in every 3 iterations.

Using your implementation of the algorithm, you must report the sum of distances of neighboring nodes during the initialization phase and after each running cycle. Also you should also generate the node graphs for 0(the initial phase), 5,10,15 and 20 cycles (the nodes are placed along a circle and the neighbors are linked by lines) as is done in the reading supplement page 50. You can use a graph visualizer/plotter program of your choice (e.g. the Matlab toolbox RNB but better ones may exist.).

Also refer to the paper T-Man: Fast Gossip-based Constructions of Large- Scale Overlay Topologies by Mark Jelasity and Ozalp Babaoglu (reading supplement 2) for a more detailed description of the Jelasity and Babaoglus algorithm.

Details of the homework implementation are explained as follows.

Please read the following notes very carefully:

1. You will write a sequential program so that in each cycle, every node in the network initiates communication with one randomly chosen neighbor.

2. Your program should accept the following parameters.

    - x, the task number.
    - N, the total number of nodes in the network.
    - k, the number of neighbors each node maintains.
    - n, the number of r values.
    - r1,r2,r3... The sequence of r values whose total number is specified by the above n. For example in java , the following command will run task 1 with 1000 nodes, 3 neighbors for each node and the ring radius set to 5.

        ```
        java hw1 1 1000 3  1 5
        ```

        For task 1, the value of r is asked only once (n=1). For task 2, the sequence of r is read one by one every 5 iterations. Other parameters remain the same.

    - The output of the program is the sum of distances for each cycle. For 0, 5, 10, 15 and 20 cycles the program produces the node graph files (png, jpg, gif ...) and a file that contains the neighbor list for each node. Any output file name should have a prefix <task_number>_N <number of nodes>_k<number of neighbors>.

        ```
        e.g. 1_N100_k3
        ```

        Then, the sum of distances file is named <prefix>.txt

        ```
        e.g. 1_N100_k3.txt
        ```

        The node graph file is named <prefix>_<current cycle>.<file extension>.

        ```
        e.g. 1_N100_k3_15.png
        ```

        The neighbor list file is named <prefix>_<current cycle>.<file extension>.

        ```
        e.g. 1_N100_k3_15.txt
        ```

3. The nodes are identified using integers (1, 2 ... N). The distance between two nodes a and b is defined by the Euclidean distance. In a plane with point a at (x_1, y_1) and point b at(x_2, y_2), the Euclidean distance between them is defined as

    $$\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2} \tag{1}$$

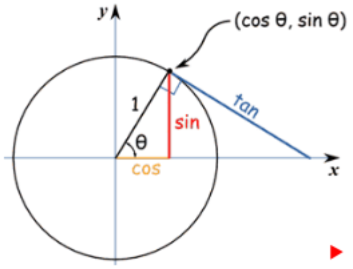
    Using this distance will result in a ring.

4. The sum of distances between neighboring nodes is defined as,

    $$\sum_{j}^{N} \sum_{node_i \in neighbors(node_j)}^{N} distance(node_i, node_j) \tag{2}$$

    where neighbors($node_j$) indicates all the nodes stored in the neighbor list of $node_j$.

5. For convenience, a node location is given by $(cos\theta, sin\theta)$ where θ is the length of the arch or angle from the node to the positive x-axis in radians.



6. In the second task, when the ring radius increases, your program will map all nodes on the original circle to a larger circle. The connections between nodes should not change, that is the neighbor lists of each node remain the same, only the node location is modified.

Deliverables

1. A file named as *yourname*_hw1.tar containing:

   - The source code of your program. Remember to use good software design and development practices.
     (a) Comment your code to describe what your code does.
     (b) Use variable names that are descriptive of their function.
   - A text file that explains how to compile and run your program. This should contain the exact command line to be used to run your program with all necessary command line arguments. If your source code is in multiple files, describe briefly the content of each file.
   - Do NOT send binary files. Use the file names as specified above. All these files should be inside a directory named your name, i.e. the following command should work to create the tar file:

     ```
     tar cvf yourname_hw1.tar *
     ```

2. A PDF document named *yourname*_hw1.pdf describing your results and containing the plots.

Submission Policy

– Follow strictly to the format specified in every homework. Incorrect submission formats will lead to a grade reduction.

– All submissions are expected by the deadline specified in the homework assignment. Grade is automatically reduced by 25% for every late day.

– Make sure you test your submitted code using the tar file you submitted before submission. If untar, make, compile or any other command needed to execute your program do not work, your homework grade will be zero.