EEL 6935: Distributed Computing Spring 2017 Instructor: J. Fortes

Homework 1 Assigned: 1/5/2017 Due: 1/19/2017 @ 5 Pm (use UF E-learning - Canvas)

Topology Management in Peer-to-Peer Systems

Preamble:

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. Tanenbaum's Distributed Systems book and in attached reading supplement 1 (The paper "T-Man: Fast Gossip-based Constructions of Large-Scale Overlay Topologies" by Mark Jelasity and Ozalp Babaoglu). This algorithm is also known as Jelasity and Babaoglu's algorithm. The gist of this algorithm is discussed in the next paragraph (use the above references for the detailed descriptions).

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 old lists as their neighbors and discards all the others.

You will write a sequential program that implements Jelasity and Babaoglu's algorithm so that in each cycle the nodes in the network initiate communication with each of their neighbors one by one. Your program MUST accept N, the total number of nodes in the network, and k, the number of neighbors each node maintains as the input parameters.

Tasks:

There are 3 tasks as part of this homework which are elaborated below:

1. (**Dynamic Ring Topology**): Consider the case where the nodes are identified using integers *(1, 2…N)* and the distance between two nodes *a* and *b* is defined by the Euclidean distance. In a plane with point a at (x1, y1) and point b at (x2, y2), the Euclidean distance between them is defined as

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

   Using this distance will result in a ring.

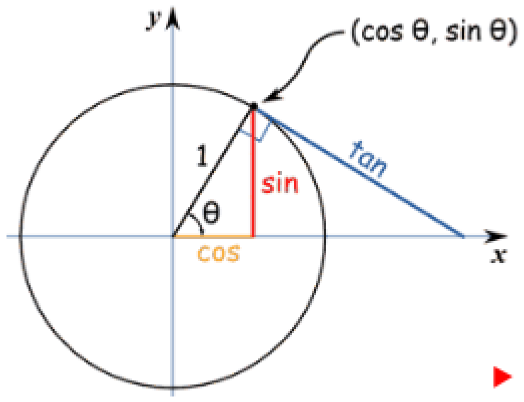
   You will write a program that will accept ***N***, the total number of nodes in the network, ***k,*** the number of neighbors each node maintains, ***n***, the number of radius values, ***r1, r2, r3…,*** the sequence of 'r' values whose total number is specified by the above **n**. The value of **r** 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.

a. 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. The sum of distances between neighboring nodes is defined as

$$\sum_{i=1}^{N} \sum_{node_j \in neighbors(node_i)}^{N} distance(node_i, node_j)$$

where *neighbors*(***node**$_i$*) indicates all the nodes stored in the neighbor list of (***node**$_i$*)

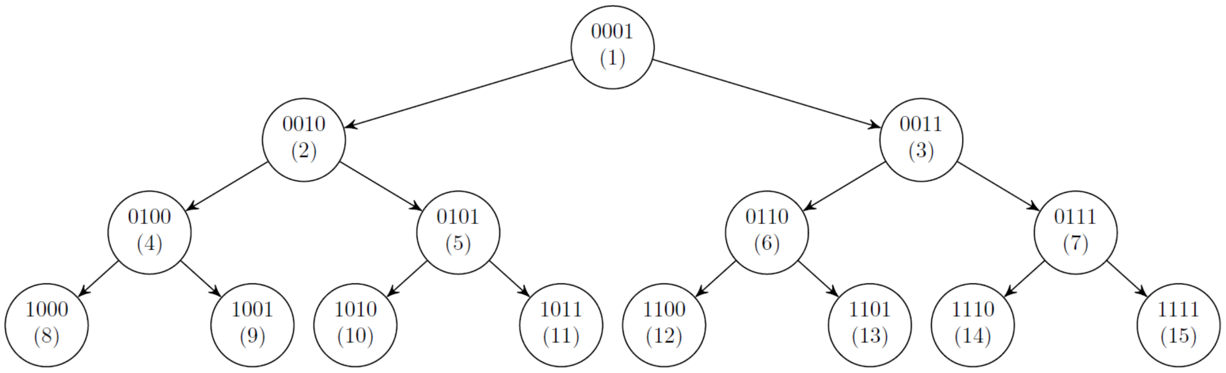
b. For convenience, a node's location is given by (cos θ, sin θ) where θ is the length of the arch or angle from the node to the positive x-axis in radians.



2. **(Binary Tree Topology):** You will modify the program to only accept ***N*** and ***k*** as inputs for this case. The topology you will be constructing is the binary tree. The following is an excerpt from Babaoglu's paper describing how this topology can be obtained.

"*Binary tree: A low diameter topology can be constructed from a binary tree: the profiles are binary strings of length m, excluding the all zero string. Distance is defined as the shortest path length between the two nodes in the following undirected rooted binary tree. The string 0 . . . 01 is the root. Any string 0a$_{2...m}$ has two children a$_2$ . . . a$_m$0 and a$_2$ . . . a$_m$1. Strings starting with 1 are leaves. This topology is of interest because (unlike the previous ones) it has a very short (logarithmic) diameter of 2m.*"

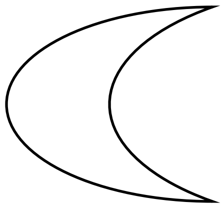
The graphic given below shows you an example of the naming of nodes using binary strings. Here m=4 i.e. nodes are named with binary strings of length 4. The distance between the nodes "5" and "10" is 1. The distance between nodes "5" and "6" is 4. The distance between nodes “5” and “13” is 5. The root node is always considered to be at level 1. In this case the leaves are at level 4. A node whose value is between 2^x and 2^{x+1} -1 will be at level x+1.



a. Given below is a sample code to find the distance between any two nodes a and b.

```
int dist (int a, int b) {
      int bits = 32;
      int alevel=bits;
      int blevel=bits;
      int commonprefix=0;
      int mask = 1 << bits-1;
      // find the level of node a
      while( (mask & a) == 0 )
      {
            a <<= 1;
            alevel--;
      }
      // find the level of node b
      while( (mask & b) == 0 )
      {
            b <<= 1;
            blevel--;
      }
      int length = Math.min(alevel,blevel);
      while( (mask & ~(a ^ b)) != 0 && length>0)
      {
            b <<= 1;
            a <<= 1;
            commonprefix++;
            length--;
      }
      return alevel - commonprefix + blevel - commonprefix;
}
```

3. **(Crescent Moon Topology):** Change your code and the definition of distance so that a network shaped as a "crescent moon" results from running the code. **Explain your definition of distance for this case.**



4. **Additional Instructions:**

   - In the task 1, dynamic ring topology, 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.

   - In the homework report, you should show the results of testing your program using N=1000 nodes and k=30 neighbors and running it for 50 cycles.
     - For the task 1 – dynamic ring topology, your homework report should show the results of testing your program using N=1000 nodes, k=30 neighbors, n=5, r1=1, r2=4, r3=5, r4=8, r5=10 and running it for 50 cycles.

   - You are also required to draw a two-dimensional plot showing the sum of distances between neighboring nodes after each running cycle of the initialization phase. In the plot, the vertical axis represents the sum of distances and the horizontal axis represents the number of cycles. You can use any tool of your choice to generate the plot.

5. **Questions**

   a. What methods do you use to ensure that there are no separated nodes in the dynamic ring topology and the crescent moon topology?

   b. Can a node's neighbor list show the same node in multiple entries?

<u>Submission (*Thursday, 01/19/2017, 5 Pm*):</u>

1. Your source code containing the main method must be named TMAN.<extension of your program>. The names of the other supporting classes, if any, is up-to your choice.
2. Comment every module/class/function in your code and use descriptive variable names.
3. Your program MUST run using the command line arguments as shown below. It must accept the input arguments ***N***, ***k*** and *topology,* where the topology can be one of **D, B or C** which represents dynamic ring, binary tree and crescent moon respectively. The total number of cycles is fixed at 50 for this homework.

a. For the dynamic ring topology, your program MUST accept additional parameters ***n*** and ***r1,r2,r3….*** The sequence of radii will be comma separated i.e. 1,3,5,6,7.
   Format: **TMAN N k topology n r1,r2,r3,…,rn**

Below is an example for the different languages. The example also illustrates a sample input for the dynamic ring topology. Note that the values of **n** and **r's** will not be provided for the other topologies.

- ***C: gcc TMAN N k topology***
- ***C++: g++ TMAN N k topology***
- ***Java: java TMAN N k topology***
- ***Matlab: matlab TMAN N k topology***

- ***C: gcc TMAN N k D 5 1,3,5,6,7***
- ***C++: g++ TMAN N k D 5 1,3,5,6,7***
- ***Java: java TMAN N k D 5 1,3,5,6,7***
- ***Matlab: matlab TMAN N k D 5 1,3,5,6,7***

4. The output of the program is the sum of distance for each cycle. For ***1, 5, 10 and 15*** 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 - <topology>_N<number of nodes>_k<number of neighbors>.
   e.g. **D_N100_k3**

   Then, the sum of distances file is named <prefix>.txt
   e.g. **D_N100_k3.txt**

   The node graph file is name <prefix>_<current cycle>.<file extension>
   e.g. **D_N100_k3_10.png**

   The neighbor list file is named <prefix>_<current cycle>.<file extension>
   e.g. **D_N100_k3_10.txt**

5. There are two files that you will be uploading as part of the homework submission. A tar file and a PDF file. **Do NOT tar the PDF file**.
   a. Upload the following files within a SINGLE tar file named "yourLastName_hw1.tar":

      - The source code of your program
      - A file named "makefile" to compile the program. The TA will type only "make" to compile the program. Other methods are not allowed.
      - A text file named "readme.txt" that precisely specifies the running environment including the operating system, language written, compiler version and any software needed to run your program. It also describes the program structure such

as files, classes and significant methods. If your source code is in multiple files, describe briefly the content of each file.

- All these files should be tarred into a file i.e. the following command should work to create the tar file.

  *$> tar cvf <yourLastName_hw1>.tar **

b. A PDF document (yourLastName_hw1.pdf) describing the definition of distance for the crescent moon topology and all results. Include the plots and the answers to the questions. Also, include your source code in the PDF. The source code MUST be pasted towards the end of the PDF. Include all your classes/modules/functions in the PDF file.

   NOTE: The PDF file MUST NOT be tarred. You will have two files to submit. A tar that contains your source code, readme file and a make file, and a PDF file that contains plots, answers to questions, definition of distance for the crescent moon and pasted text of your source code.

6. The tar file and the PDF must be uploaded to Canvas.

Submission Policy:

- Do NOT include binary files. Use the file names as specified above. 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 to test your submitted code using the tar file. If untar, make, compile or any other command needed to executed your program do NOT work, your homework grade will be **zero.**