Distributed Systems

IN5020

Programming Assignment 3

shielak, praveema, kathabar

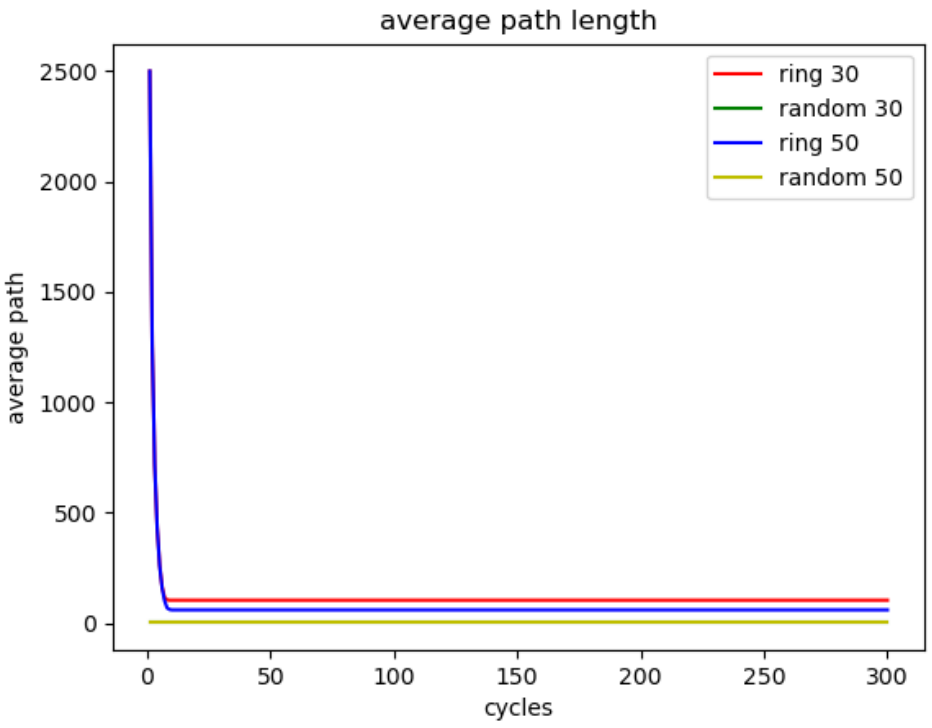
1 User guide to compiling and running the distributed application

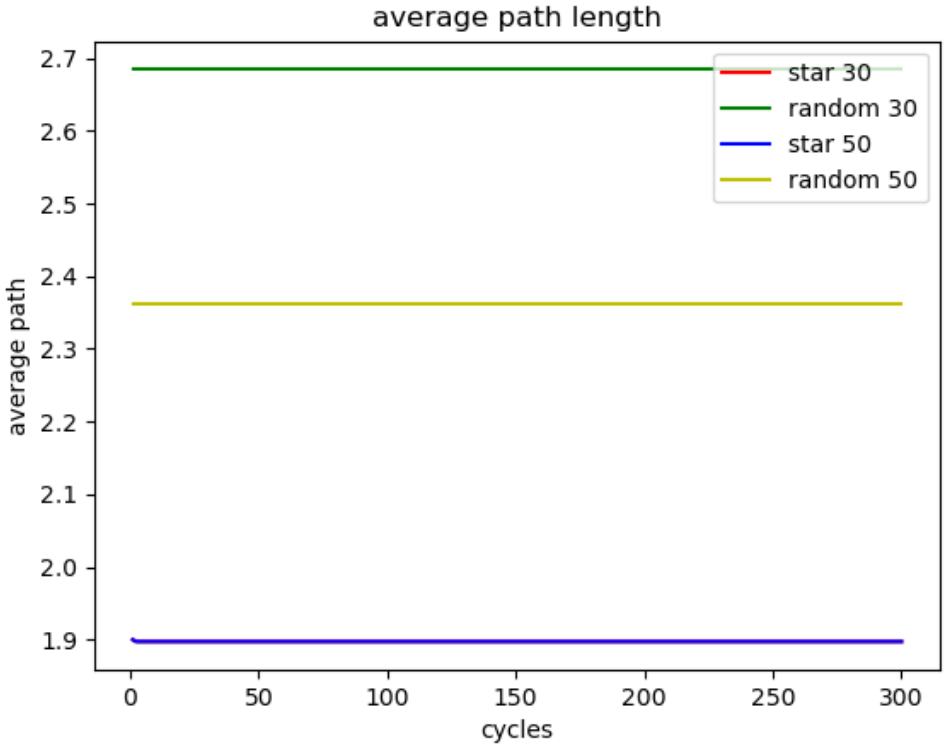
steps to execute the jar file:
1) The first step is to compile the java files.

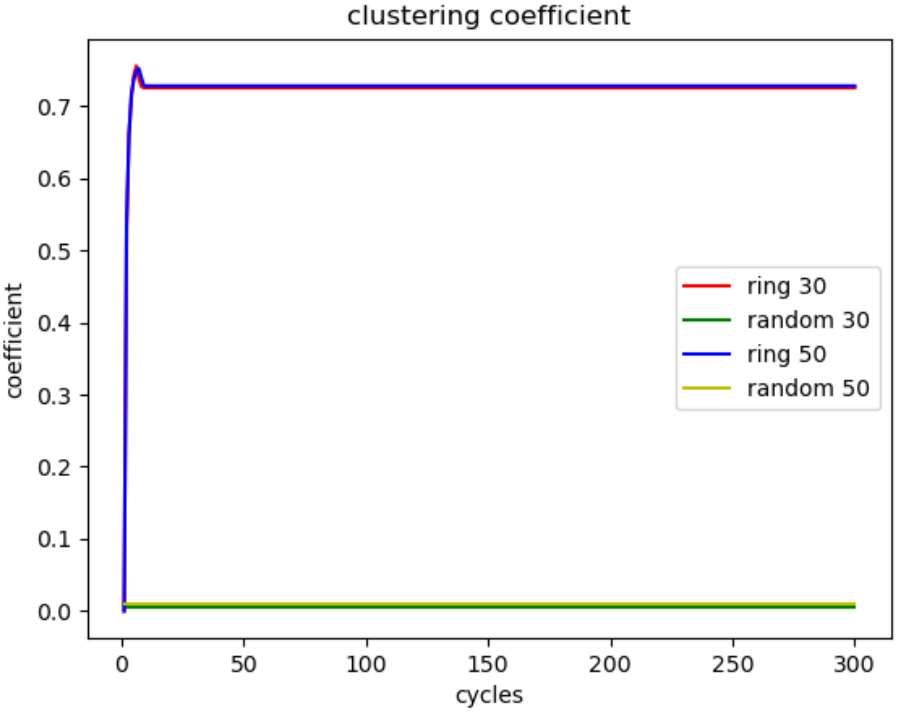
```
java -jar assignment3.jar scripts/<filename>
```

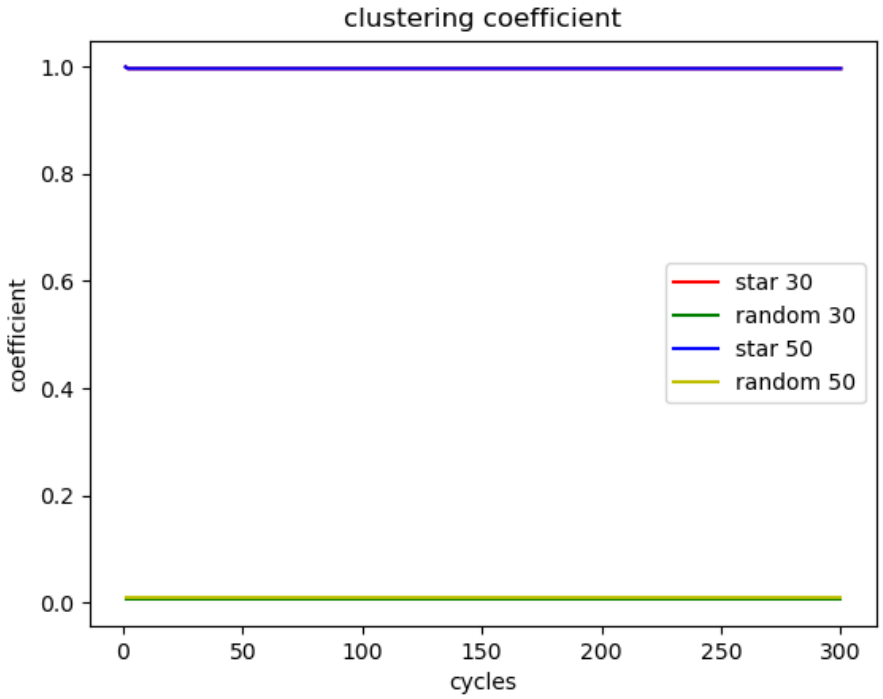
filename - ring30.txt, ring50.txt, star30.txt, star50.txt, random30.txt, random50.txt

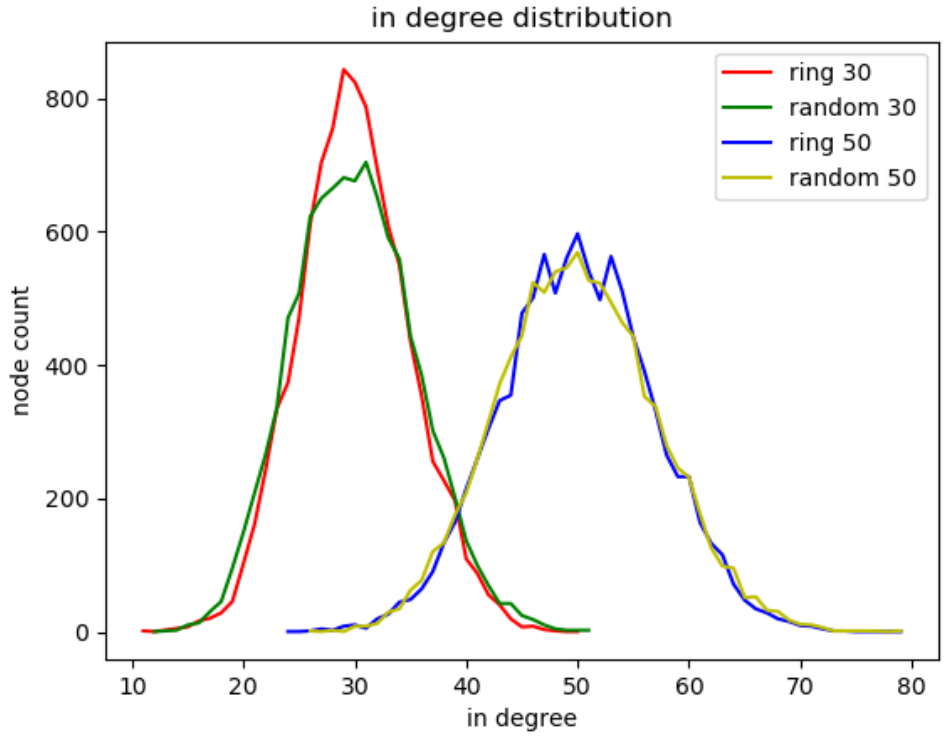
2 Graphs

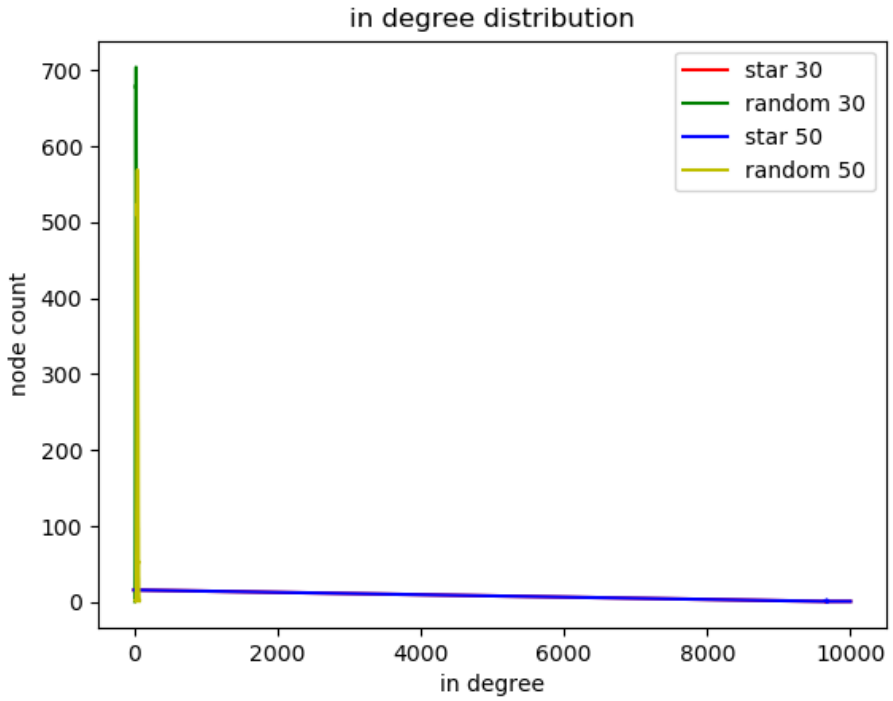



average path length
star 30
random 30
star 50
random 50
average path
2.7
2.6
2.5
2.4
2.3
2.2
2.1
2.0
1.9
0
50
100
150
200
250
300
cycles


clustering coefficient
ring 30
random 30
ring 50
random 50
coefficient
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
50
100
150
200
250
300
cycles


clustering coefficient
1.0
0.8
0.6
0.4
0.2
0.0
coefficient
0
50
100
150
200
250
300
cycles
star 30
random 30
star 50
random 50


in degree distribution
800
600
400
200
0
node count
10
20
30
40
50
60
70
80
in degree
ring 30
random 30
ring 50
random 50



3 Design Choices

3.1 Components used from the peersim

1. Protocol and Control interfaces - the existing template provided in the assignment used the different variations of protocol and control interfaces.
    a. There are 2 types of protocol variation: one is cycle based and the other one is event based. The class which implements the protocol interface runs the simulation.
    b. There are multiple variations of control interface. In-degree observer implemented the control interface. The clustering coefficient parameter and the shortest path length parameter observers are based on GraphObserver class which is provided by the peersim.
2. Script file for the simulator class - The script is used to pass the input to the simulator. All the network configuration, simulation configuration and the observer configurations are passed in this text file.

3.2 Basic Shuffle Protocol

1. The algorithm is provided in the template. The only complexity was finding out how to use the peer sim functions to implement the algorithm. The cache contains the list of edges for each node since it inherited the protocol interface the simulator directly configure the topology on top of the neighbor manipulation functions provided in the class. All the neighbor manipulation functions uses the cache to add/remove neighbors.
2. I have implemented 4 different functions (or) data structures to implement the shuffle

algorithm apart from the code examples provided in the template.

a. removeNeighbor function- removes the entry from the cache based on the index provided as input
b. The flag isWaitingForReply is used by the node to make sure that it won't flood the network with requests
c. getNeighborsSubsetFromCache function - returns the subset from the cache based on the shuffle length and the cache size
d. ShuffleCache - follows the algorithm provided in the template

3.3 Observers

3. The in-degree observer is provided in the template
4. The clustering coefficient and the average path observers are provided in the GraphStats class
5. All the 3 observers write the output values to files. The data from the file is passed to custom python program to plot the graphs. The plot function used the matplotlib library to plot the graph.

4 Analysis

1. Average path - star topology performs better than ring
2. Clustering coefficient - ring performs better than star
3. in-degree distribution - ring performs better than star
4. overall observation - it seems like star topology is not a good choice based on the graphs. The random topology shows a reasonable performance in 3 parameters.

5 Group Work

We have spent a whole day to understand the peersim simulator and the different components it contains. The main difficulty we have faced was we couldn't find a lot of tutorials in the internet for the peer sim. So we spent the whole day to set up our working environment and set up the peer simulator code base. Then after that we split the tasks into 3 different parts and we were not sure how complex each tasks would be since we couldn't estimate how much dependency each task has on peer sim components. Then we tried to understand the components again and see how to reuse the functions provided in the peersim. Finally we have implemented the logic and created the graph.