Programming project 3:
Worm Propagation Simulation

Name: Megha Darshan Uma Ranganatha
UCF Id: 4481996

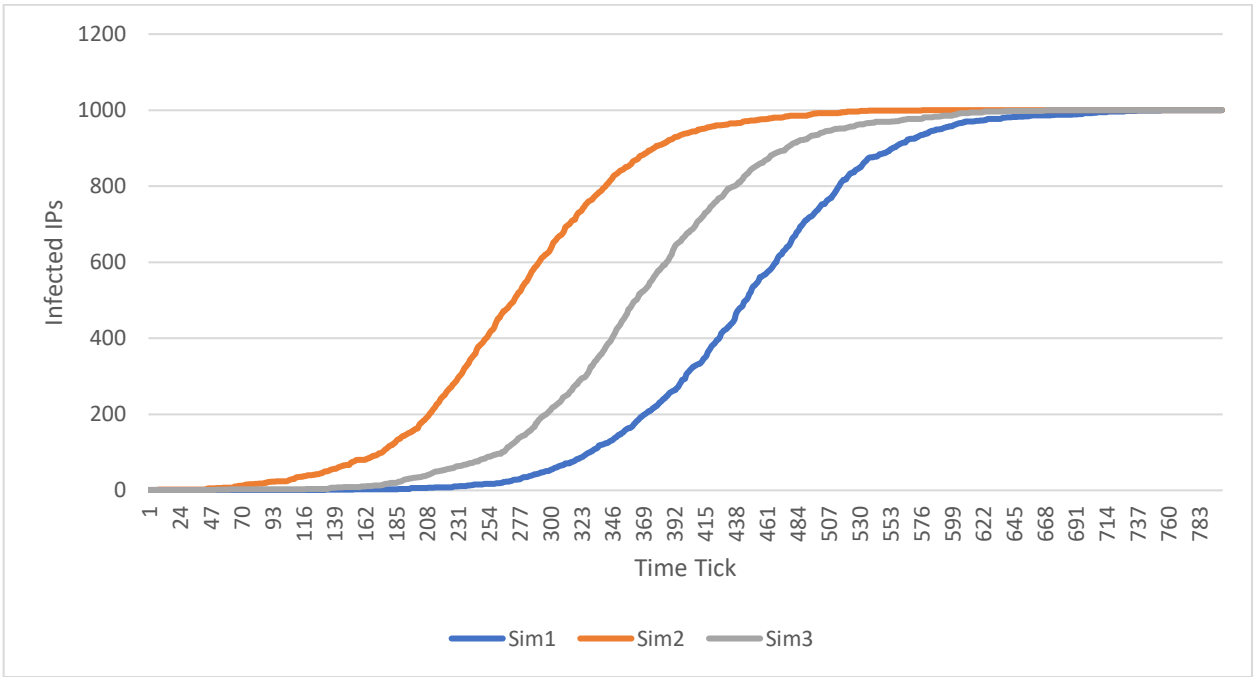
Design of the Simulation:
The worm propagation simulation is designed using Java.

1. WormR.java
2. WormL.java

Steps followed to design Simulation model:

1. Random Scanning worm propagation (wormR.java).
for (x<simulN)

- Initialize the vectors node status
  If (index is between 1-10)
  { then susceptible}
  If (index is between 11-1000)
  { then empty}
- Increment Time tick: k ++;
  For system's entity IP addresses (i= 1,2…10)
  {node status is updated as infected}
- Simulate for node i in the last time step (k-1 -> k) based on all
  nodes status at time tick k-1. Then, update the state of node i if there is any change. Finally, output time tick k's system's states.

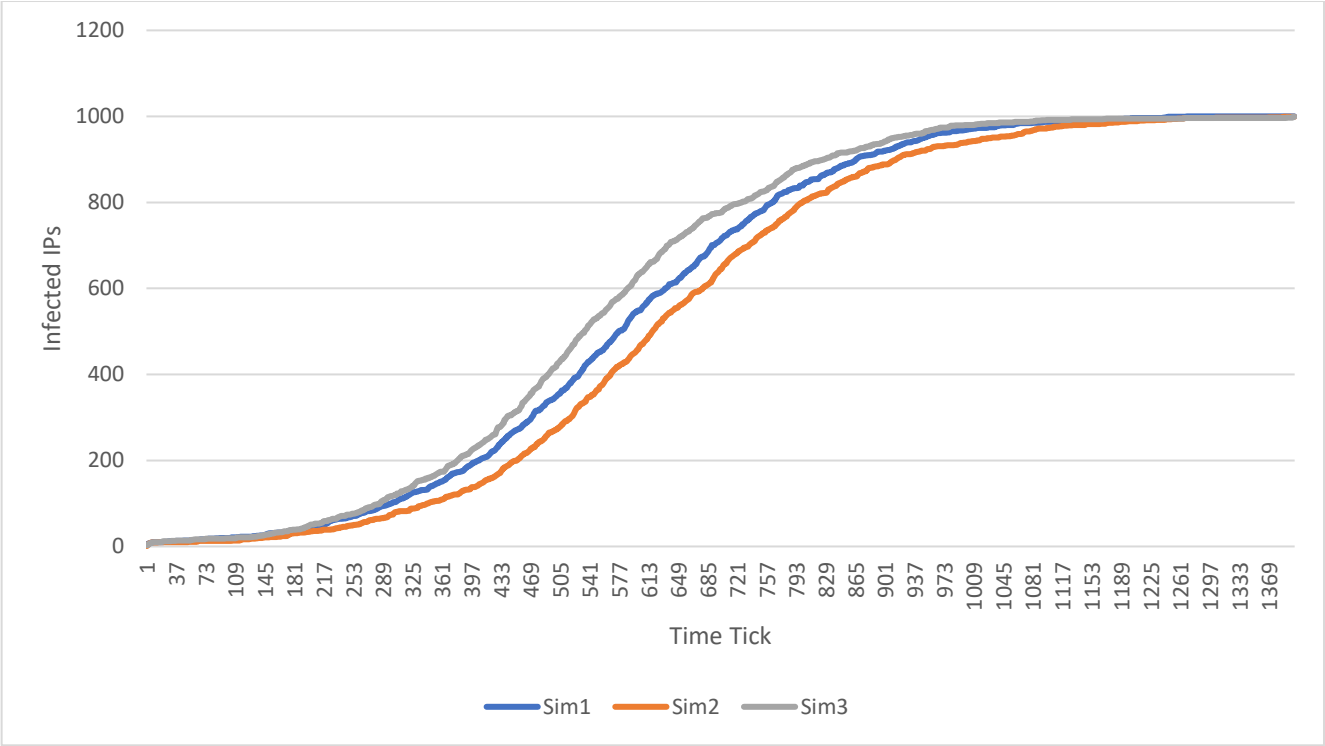


Graph(a) represents random scanning simulations performed 3 times. It is seen from the above graph that plots each of data points Time tick and Infection IPs till when the infection count reaches 1000. the different coloured curves in the graph represents the plot of the different simulations. As shown in the

graph the rate of propagation remains exponential for an extent of time and it turned constant as the number of infections reaches 1000.

2. Working of local preference scanning worm propagation (wormL.java):

- Increment Time tick: k ++;
- for (x<simulN) Initialize the vectors node status
  If (index is between 1-10)
  { then susceptible}
  If (index is between 11-1000)
  { then empty}
- Generate random values between 0 and 1. If the values generated is between 0 and 0.6
  {then Select the current node between (currentnode-10) and (current node+10) }
- For system's entity IP addresses (i=1,2…10). Then update the node status as infected
  Simulate what could happen for node i during the last time step (k-1 -> k) based on time tick at k-1.
  Update the state of node i if there is any change. Then, Output time tick k's system's states.



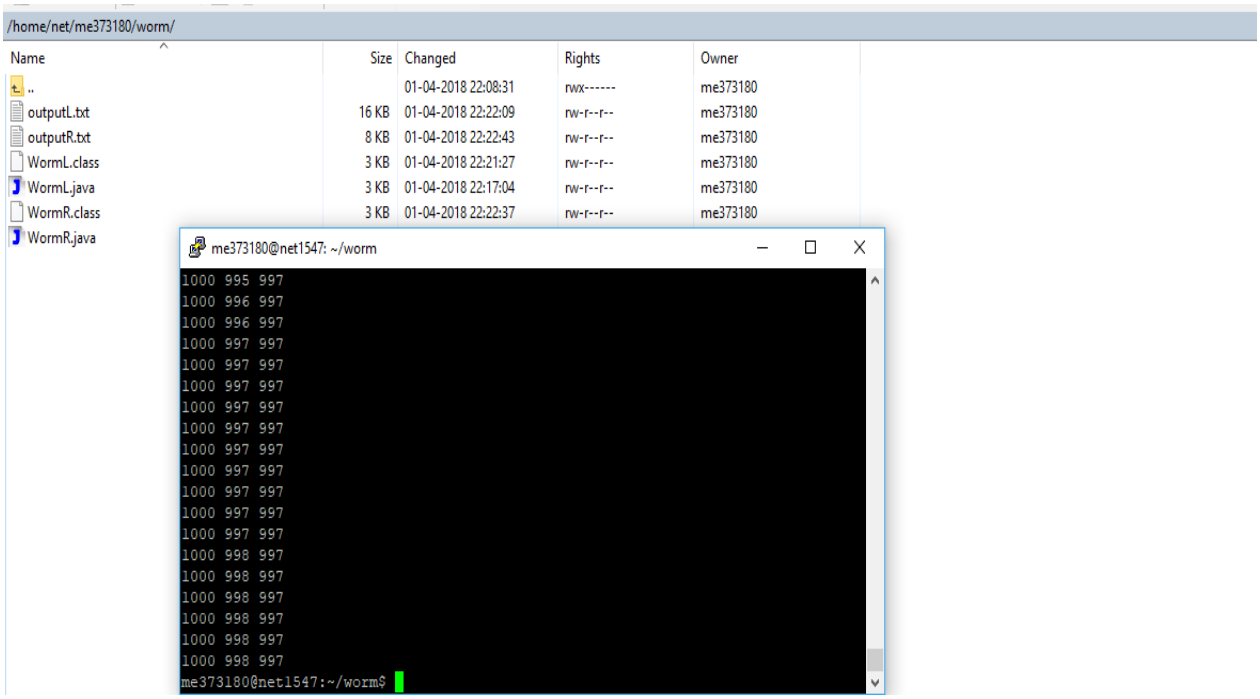
The graph above represents 3 random scanning simulations performed. We plot each of the data points Time Tick and infection Ips until the infection count reaches 1000. It can be seen from the graph that the rate of propagation of worm is initially low and increases rapidly as the time tick reaches about 250. The rate of propagation became constant as the number of infections reaches 1000.

Steps to execute the WormL and WormR files in Eustis machine:

- Open putty client
- Navigate to the code path: home/net/me373180/worm/
- Excecute below command
  1. Javac WormL.java
     Java WormL
  2. Javac WormR.java
     Java WormR

- Check for the two txt files created:
  1. OutputL.txt
  2. OutputR.txt

Below is the screenshot of the WormL and WormR code run in Eustis:



Screenshot shows the two text files successfully created OutputL.txt and OutputR.txt after I ran the files WormL and WormR.

Below are the important variables used in the code:

- **int scanRate = 2** //Initialize the Scan rate;
- **int simulRun= 3** //Define max no of runs 3 times for which the simulation runs,
- **int omega = 100000** //Initialize the IP address space
- **int simulTime = 1400** // Here we are assigning the maximum time tick value as 1400.
- **int [][] infNode = new int[simulRun][simulTime]** //Create a 2-D matrix to store the number of infected nodes. The dimension is simulRun x simulTime;
- **int idxSeed = ThreadLocalRandom.current().nextInt(0, k) *1000+ThreadLocalRandom.current().nextInt(0,10);** //Using idxSeed to store current index of the IP addresses that are susceptible.
- **int infectComputer** //infected is used to keep the count of the infected IP addresses after each simulation run.
- **int IP** // Store the IP address in the range.

Code Explanation:

Code to generate values with probability p=0.6, it picks value y such that x-10<=y <=x+10:

```
for(int t=0; t<simulTime; t++){
        for(int j=0; j<infectComputer; j++){
            for(int i=0; i<scanRate;i++){
                double randomNo = ThreadLocalRandom.current().nextDouble(0, 1);
            int temp =idxSeed;
            int current_index=temp;
```

```
        if(randomNo<0.6)
        {
           IP=ThreadLocalRandom.current().nextInt(current_index-10,current_index+10);
        }
        else
        {
           IP = ThreadLocalRandom.current().nextInt(0,100000);
        }
                    if(status[IP] !=null && !status[IP]){
                      status[IP] = true;
                    infectComputer++;
                    }
         }
    }
```

Code to initialize the first node in both local and random:

```
int idxSeed = ThreadLocalRandom.current().nextInt(0, k) * 1000
+ThreadLocalRandom.current().nextInt(0,10);
status[idxSeed] = true;
infect++;
System.out.println("Initial infected Comp set at IP" + seed);
```

This code generates the susceptible value between 1 to 100000 and then infects the first node.

Output files generated:

1. The output file of Random scanning worm propagation is ‘outputR.txt’.
2. The output file of Local scanning worm propagation is ‘outputL.txt’.