Distributed Computing

CS5320

Programming Assignment-I

CS19MTECH11009

KAMMARI NAVEEN KUMAR

Assignment on Termination Detection

Approach:

We need to find the termination of all the processes in the distributed system. A distributed system is a collection of computers or components (processes) that are located at different locations and they communicate with each other through message passing to perform some particular task. As we don't have availability of distributed systems , we are asked to create a distributed environment using threads and processes. The distributed environment should contain the systems/processes which are connected to each other in the form of a connected graph topology. Each node communicates with every other node through messages.

In this distributed environment each process will maintain two threads. One thread for receiving messages and another thread will be responsible for internal computation of a process. In this system, each process will have its own unique identification number , color and socket descriptor as its attributes. Each process is considered to be running on a different system. We are having 'n' no.of processes which are communicating with each other in the distributed manner. The socket programming is used to make communication between the processes. As mentioned in the assignment description, each process will maintain color. Initially all processes will contain white as their color. If the process color is red then it is active. The process will be passive if it has blue color. If all the processes are blue then termination is occured. So we need to find the termination of processes in the given distributed system.

At the starting of the program , I have read input file for the information required for the distributed environment and information related to complete the required task. I have created a thread for each process. Each process is assigned a port number , which will be used in socket programming for communication. After creating **'n'** processes , as a part of computation I have chosen some random processes which will turn their color to red and continue further communication with other processes. Each process will make communication by sending a message to its neighbours.

During the computation in the distributed system environment, one process will initiate termination detection algorithm by introducing some random processes that own blue color. We call such a node as initiator. This initiator node will detect the termination of all processes.

Dijkstra & Scholten Termination Detection Algorithm:

The computation initiated by a single initiator and spreading over to several other nodes in the network.There are two kinds of messages in the network: **signals** propagate along the direction of the edges, and **acks** propagate in the opposite direction. The initiator node will send a **signal** message towards its children to activate termination detection. If a child is idle then it sends **ack** message to its parent. So this process will be continued to know termination in the network.

Internal node which receives a signal message may propagate this message to its neighbours. The node will send an **acknowledgement** message to its parent only if all its children send him/her **ack** message and the node is idle. This **Dijkstra and Scholten** algorithm uses two **variables** called **C** and **D.** I have used the concept explained in the book by **A. Kshemkalyani and M. Singhal** .(**7.5 A Spanning-Tree-Based Termination Detection Algorithm).**

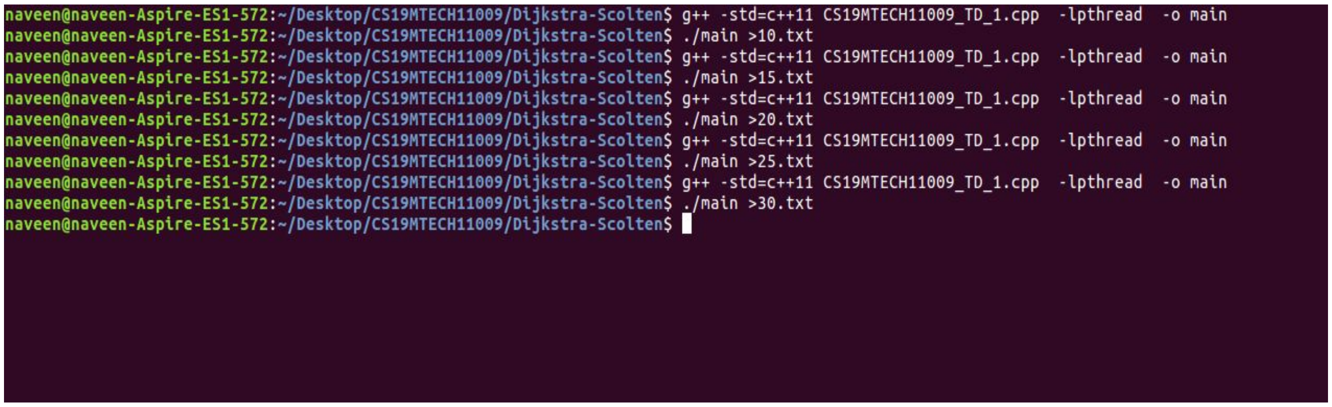
Note: So that I'm not writing the explanation of the Spanning - Tree algorithm

Programming Approach:

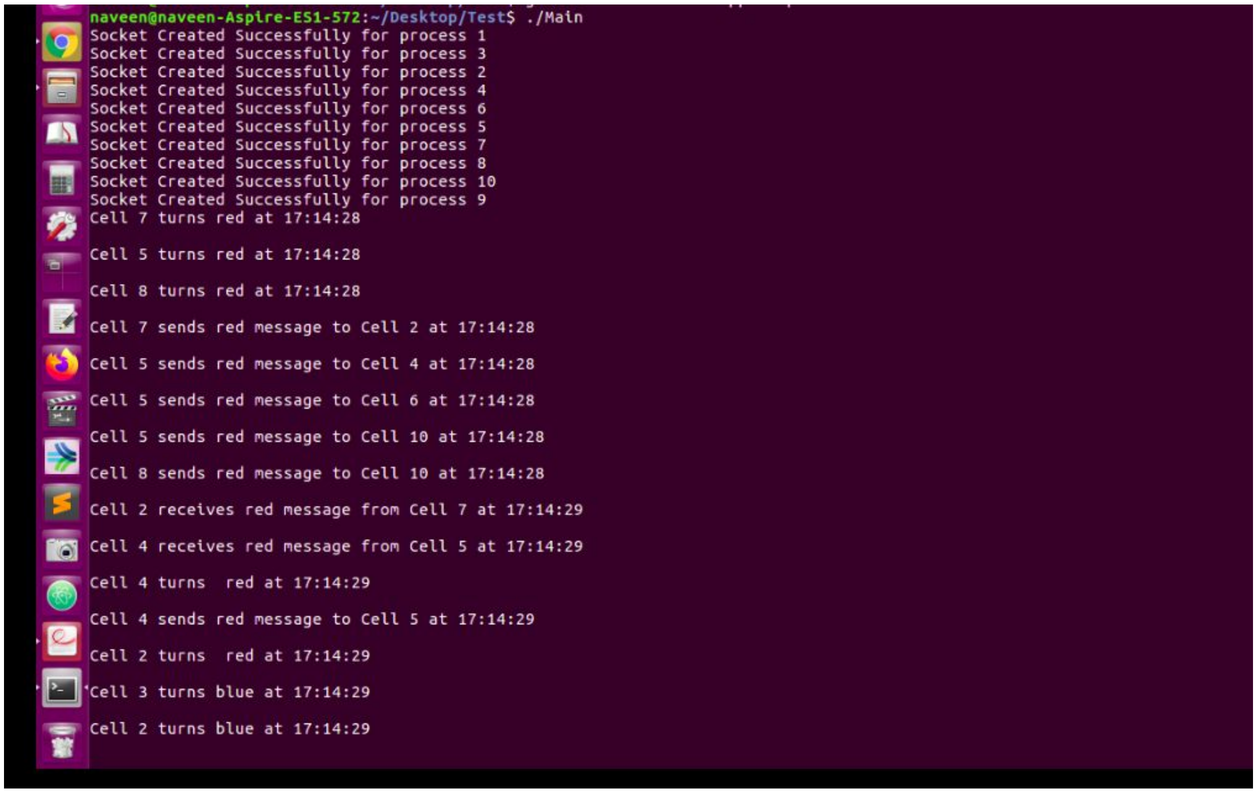
I'm solving this assignment using cpp language. In this assignment I have created a Class which is responsible for send , receive, socket creation and internal computation of the process. Each process will have all these functions. Sending and receiving the messages is the way in which processes will communicate with each other. I performed some computations in each of the send and receive functions mentioned in Process class inorder to get information to detect the termination.

I also created a main file, called CS19MTECH11009_TD_1.cpp, in which we start with the main function. In the main function , I have taken the text file as input to read all the required data to move forward with the computations. After reading a text file, I have created **n** processes using thread, during the process creation the sockets and constructor of Process class will be created for each process. After creating all the processes, I have chosen the red process as mentioned in the assignment description and started the computation.The communication will be happening between the processes , we will determine the termination if and only if all the processes are in blue color.

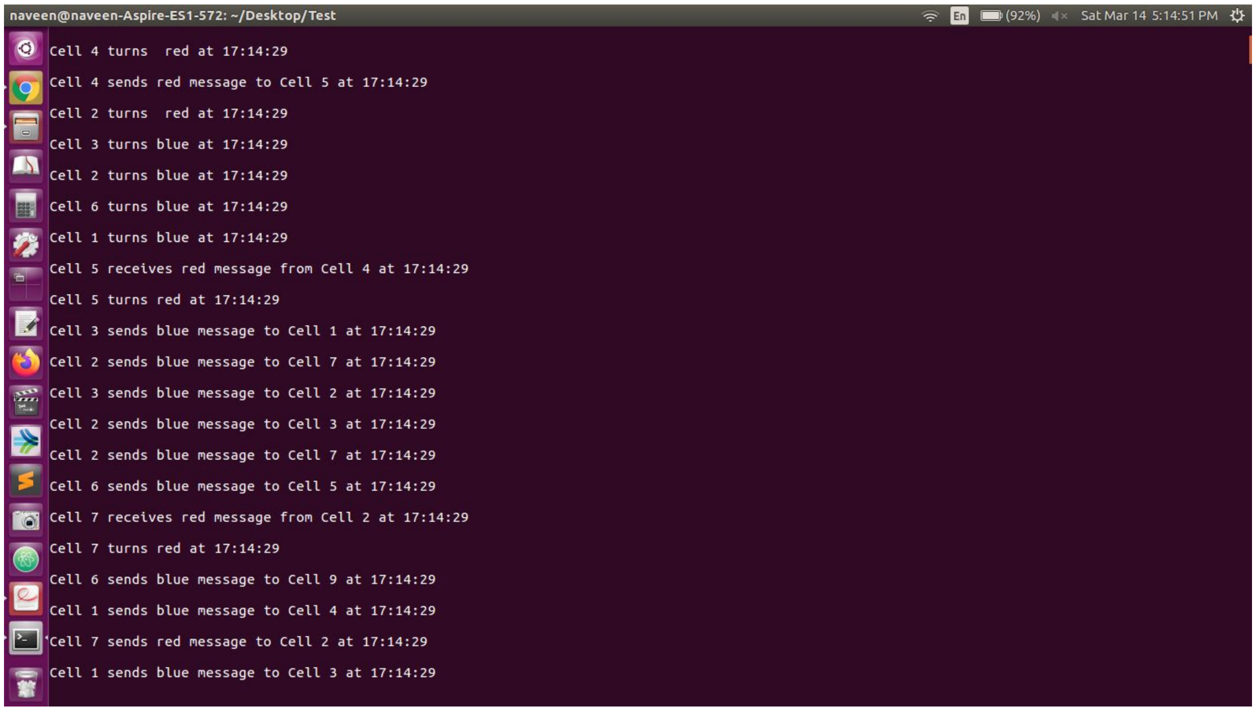
Below images show the running the code for multiple times and the output sample for the code.



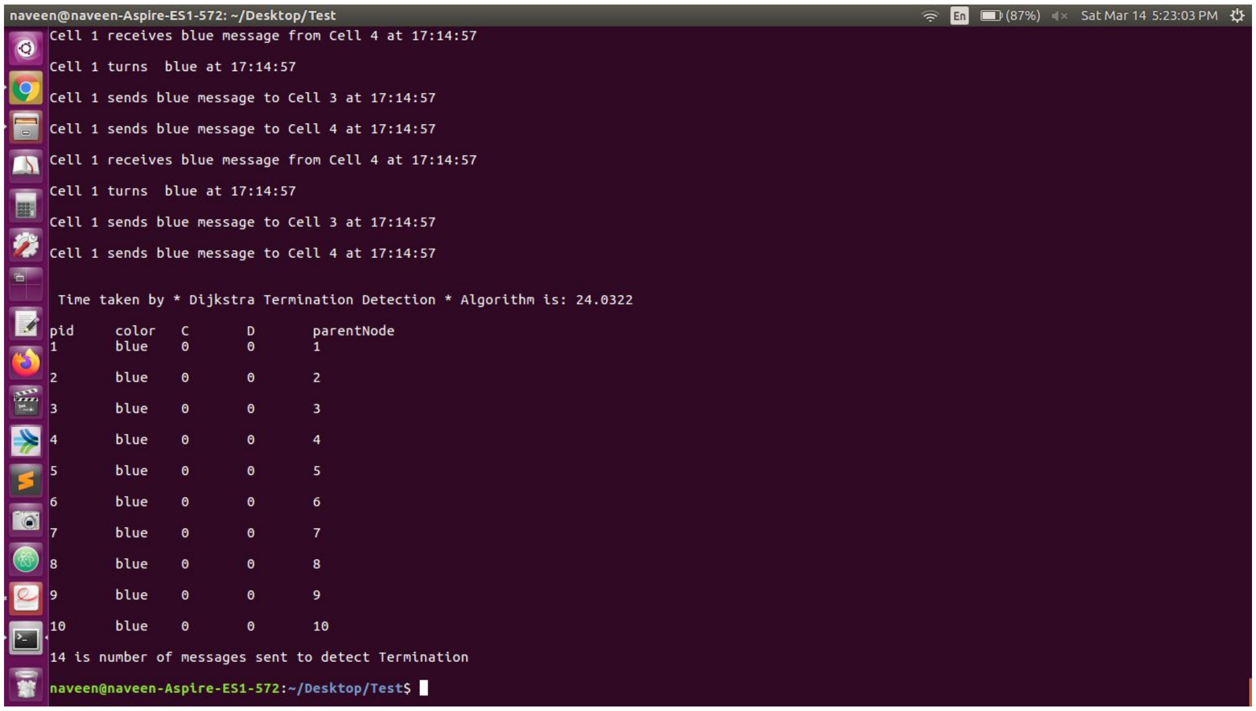
```
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ g++ -std=c++11 CS19MTECH11009_TD_1.cpp  -lpthread  -o main
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ ./main >10.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ g++ -std=c++11 CS19MTECH11009_TD_1.cpp  -lpthread  -o main
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ ./main >15.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ g++ -std=c++11 CS19MTECH11009_TD_1.cpp  -lpthread  -o main
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ ./main >20.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ g++ -std=c++11 CS19MTECH11009_TD_1.cpp  -lpthread  -o main
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ ./main >25.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ g++ -std=c++11 CS19MTECH11009_TD_1.cpp  -lpthread  -o main
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ ./main >30.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Dijkstra-Scolten$ 
```



```
naveen@naveen-Aspire-ES1-572:~/Desktop/Test$ ./Main
Socket Created Successfully for process 1
Socket Created Successfully for process 3
Socket Created Successfully for process 2
Socket Created Successfully for process 4
Socket Created Successfully for process 6
Socket Created Successfully for process 5
Socket Created Successfully for process 7
Socket Created Successfully for process 8
Socket Created Successfully for process 10
Socket Created Successfully for process 9
Cell 7 turns red at 17:14:28

Cell 5 turns red at 17:14:28

Cell 8 turns red at 17:14:28

Cell 7 sends red message to Cell 2 at 17:14:28

Cell 5 sends red message to Cell 4 at 17:14:28

Cell 5 sends red message to Cell 6 at 17:14:28

Cell 5 sends red message to Cell 10 at 17:14:28

Cell 8 sends red message to Cell 10 at 17:14:28

Cell 2 receives red message from Cell 7 at 17:14:29

Cell 4 receives red message from Cell 5 at 17:14:29

Cell 4 turns  red at 17:14:29

Cell 4 sends red message to Cell 5 at 17:14:29

Cell 2 turns  red at 17:14:29

Cell 3 turns blue at 17:14:29

Cell 2 turns blue at 17:14:29
```



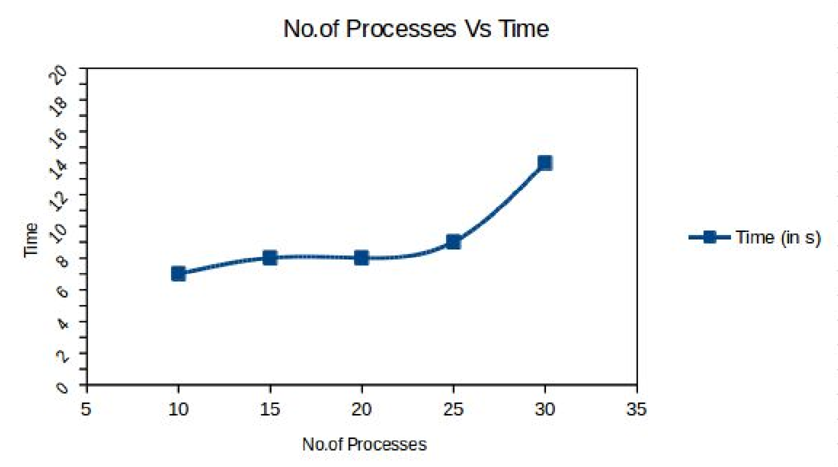
```
naveen@naveen-Aspire-ES1-572: ~/Desktop/Test                    (92%)  Sat Mar 14 5:14:51 PM
Cell 4 turns  red at 17:14:29
Cell 4 sends red message to Cell 5 at 17:14:29
Cell 2 turns  red at 17:14:29
Cell 3 turns blue at 17:14:29
Cell 2 turns blue at 17:14:29
Cell 6 turns blue at 17:14:29
Cell 1 turns blue at 17:14:29
Cell 5 receives red message from Cell 4 at 17:14:29
Cell 5 turns red at 17:14:29
Cell 3 sends blue message to Cell 1 at 17:14:29
Cell 2 sends blue message to Cell 7 at 17:14:29
Cell 3 sends blue message to Cell 2 at 17:14:29
Cell 2 sends blue message to Cell 3 at 17:14:29
Cell 2 sends blue message to Cell 7 at 17:14:29
Cell 6 sends blue message to Cell 5 at 17:14:29
Cell 7 receives red message from Cell 2 at 17:14:29
Cell 7 turns red at 17:14:29
Cell 6 sends blue message to Cell 9 at 17:14:29
Cell 1 sends blue message to Cell 4 at 17:14:29
Cell 7 sends red message to Cell 2 at 17:14:29
Cell 1 sends blue message to Cell 3 at 17:14:29
```



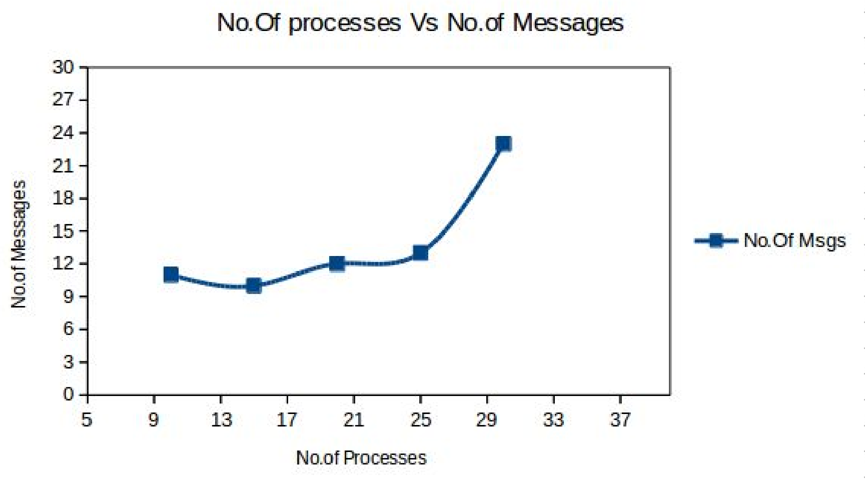
```
naveen@naveen-Aspire-ES1-572: ~/Desktop/Test                    (87%)  Sat Mar 14 5:23:03 PM
Cell 1 receives blue message from Cell 4 at 17:14:57
Cell 1 turns  blue at 17:14:57
Cell 1 sends blue message to Cell 3 at 17:14:57
Cell 1 sends blue message to Cell 4 at 17:14:57
Cell 1 receives blue message from Cell 4 at 17:14:57
Cell 1 turns  blue at 17:14:57
Cell 1 sends blue message to Cell 3 at 17:14:57
Cell 1 sends blue message to Cell 4 at 17:14:57

 Time taken by * Dijkstra Termination Detection * Algorithm is: 24.0322
pid     color   C       D       parentNode
1       blue    0       0       1

2       blue    0       0       2

3       blue    0       0       3

4       blue    0       0       4

5       blue    0       0       5

6       blue    0       0       6

7       blue    0       0       7

8       blue    0       0       8

9       blue    0       0       9

10      blue    0       0       10
14 is number of messages sent to detect Termination
naveen@naveen-Aspire-ES1-572:~/Desktop/Test$
```

The graphs related to the Dijkstra-Scholten Algorithm:

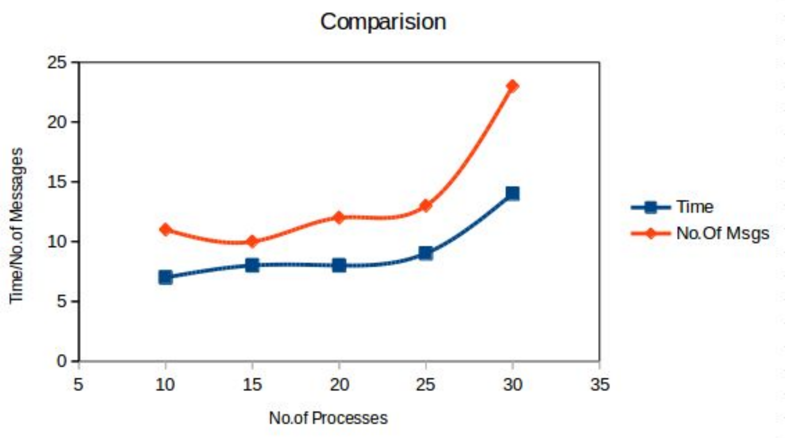
N	Time (in s)
10	7.00476
15	8.00798
20	8.00943
25	9.02353
30	14.01108



N	No.Of Msgs
10	11
15	10
20	12
25	13
30	23



N	Time	No.Of Msgs
10	7.00476	11
15	8.00798	10
20	8.00943	12
25	9.02353	13
30	14.01108	23



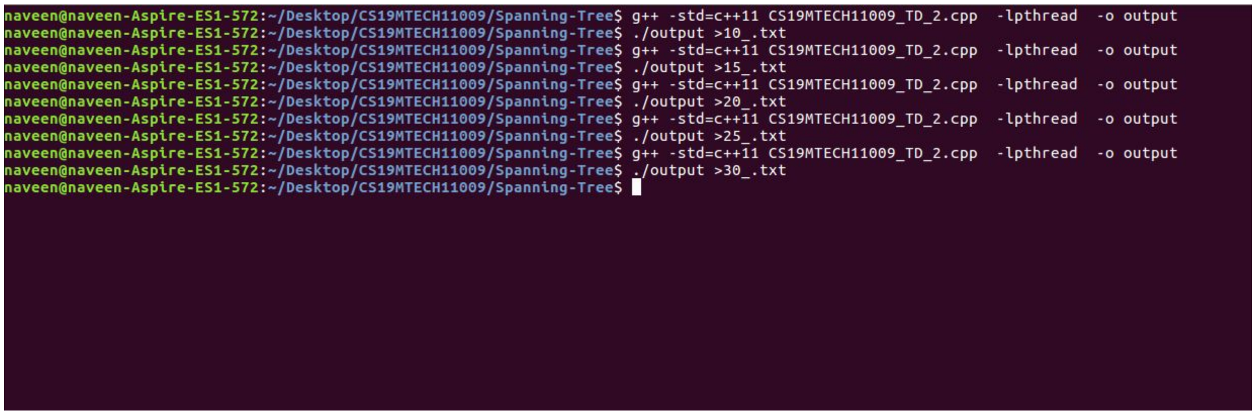
A spanning-tree-based Termination Detection Algorithm:

This algorithm is explained in book A&M section 7.5. In this algorithm we will have a fixed spanning tree. We use this spanning tree to detect the termination. I have used the same concept explained in the book.

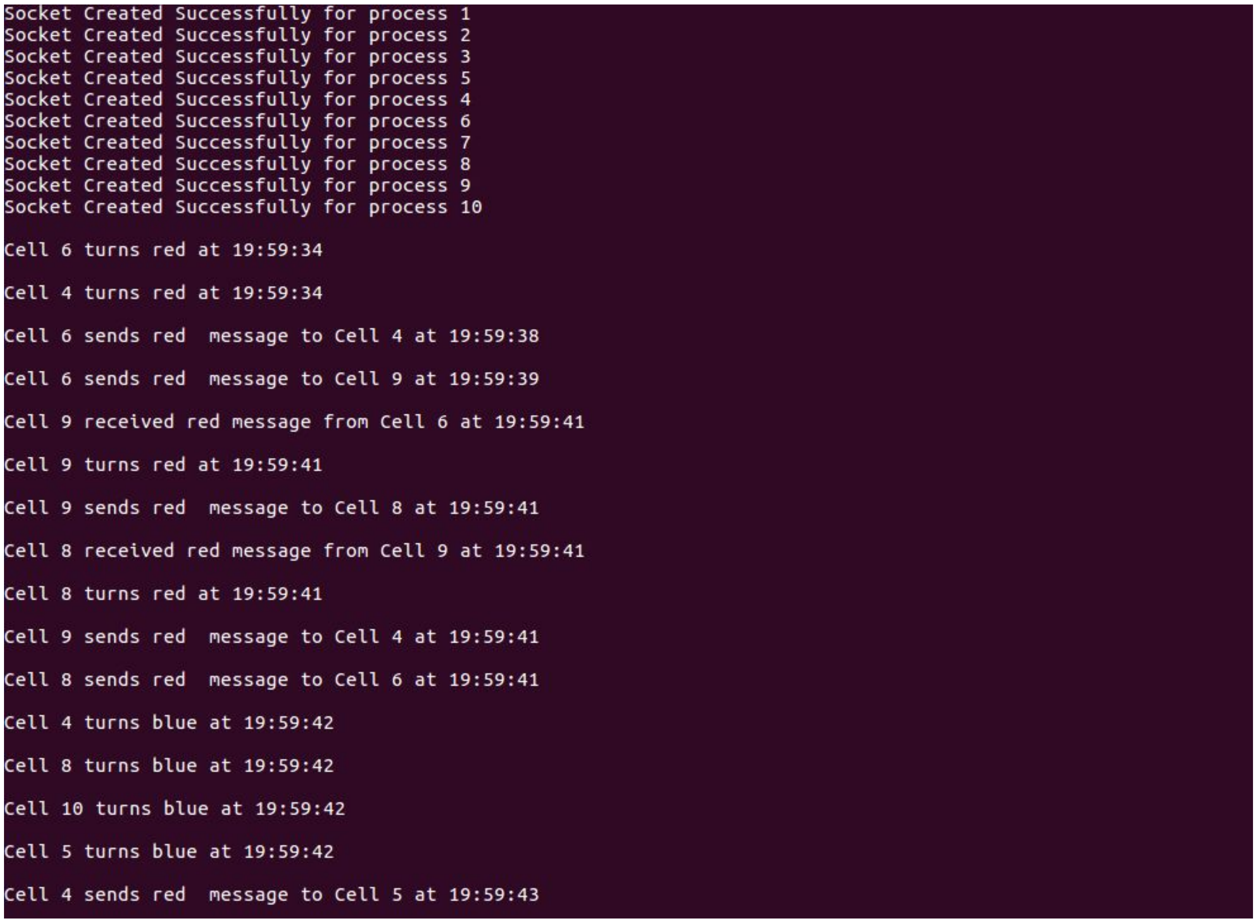
Programming Approach:

I'm solving this assignment using cpp language. In this assignment I have created a Class which is responsible for send , receive, socket creation and internal computation of the process. Each process will have all these functions. Sending and receiving the messages is the way in which processes will communicate with each other. I performed some computations in each of the send and receive functions mentioned in Process class inorder to get information to detect the termination.

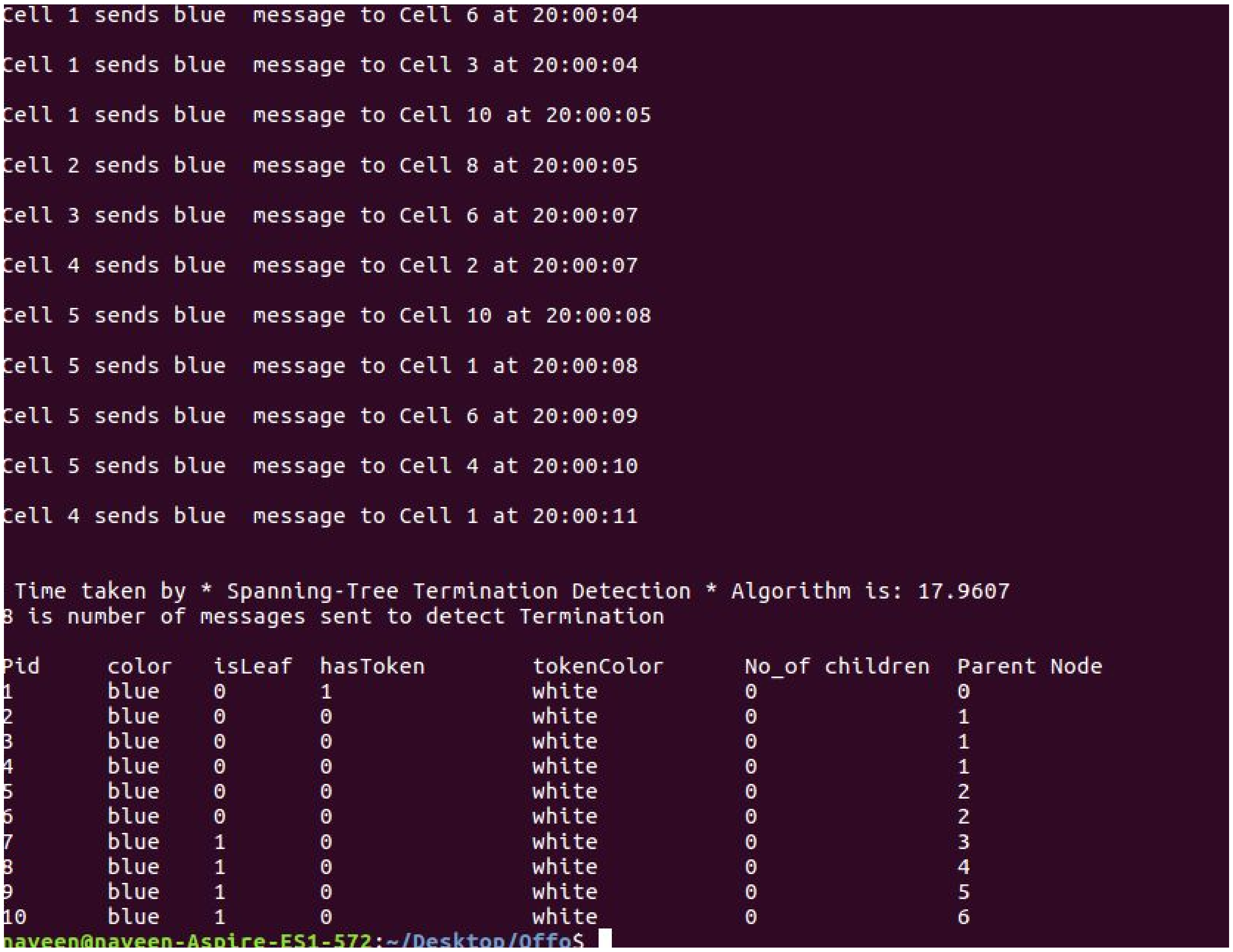
I also created a main file, called CS19MTECH11009_TD_2.cpp, in which we start with the main function. In the main function , I have taken the text file as input to read all the required data to move forward with the computations. After reading a text file, I have created **n** processes using thread, during the process creation the sockets and constructor of Process class will be created for each process. After creating all the processes, I have chosen the red process as mentioned in the assignment description and started the computation.The communication will be happening between the processes , we will determine the termination if and only if all the processes are in blue color. In this case, the root node has to get a token from its children and the token should not be black and the process should be passive.



```
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ g++ -std=c++11 CS19MTECH11009_TD_2.cpp  -lpthread  -o output
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ ./output >10_.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ g++ -std=c++11 CS19MTECH11009_TD_2.cpp  -lpthread  -o output
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ ./output >15_.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ g++ -std=c++11 CS19MTECH11009_TD_2.cpp  -lpthread  -o output
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ ./output >20_.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ g++ -std=c++11 CS19MTECH11009_TD_2.cpp  -lpthread  -o output
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ ./output >25_.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ g++ -std=c++11 CS19MTECH11009_TD_2.cpp  -lpthread  -o output
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$ ./output >30_.txt
naveen@naveen-Aspire-ES1-572:~/Desktop/CS19MTECH11009/Spanning-Tree$
```



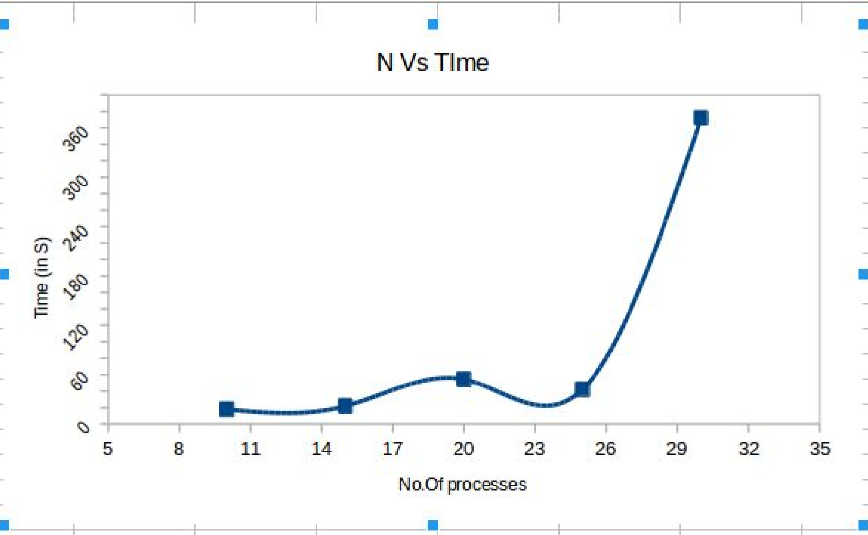
```
Socket Created Successfully for process 1
Socket Created Successfully for process 2
Socket Created Successfully for process 3
Socket Created Successfully for process 5
Socket Created Successfully for process 4
Socket Created Successfully for process 6
Socket Created Successfully for process 7
Socket Created Successfully for process 8
Socket Created Successfully for process 9
Socket Created Successfully for process 10

Cell 6 turns red at 19:59:34

Cell 4 turns red at 19:59:34

Cell 6 sends red  message to Cell 4 at 19:59:38

Cell 6 sends red  message to Cell 9 at 19:59:39

Cell 9 received red message from Cell 6 at 19:59:41

Cell 9 turns red at 19:59:41

Cell 9 sends red  message to Cell 8 at 19:59:41

Cell 8 received red message from Cell 9 at 19:59:41

Cell 8 turns red at 19:59:41

Cell 9 sends red  message to Cell 4 at 19:59:41

Cell 8 sends red  message to Cell 6 at 19:59:41

Cell 4 turns blue at 19:59:42

Cell 8 turns blue at 19:59:42

Cell 10 turns blue at 19:59:42

Cell 5 turns blue at 19:59:42

Cell 4 sends red  message to Cell 5 at 19:59:43
```



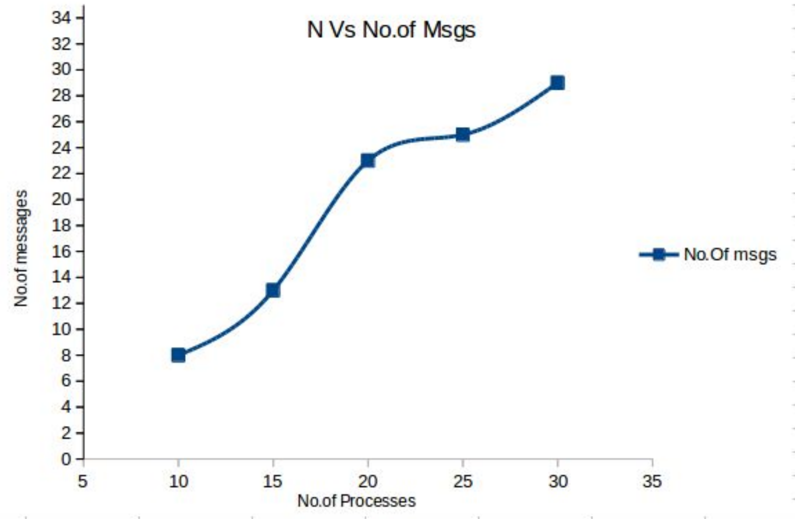
```
Cell 1 sends blue  message to Cell 6 at 20:00:04

Cell 1 sends blue  message to Cell 3 at 20:00:04

Cell 1 sends blue  message to Cell 10 at 20:00:05

Cell 2 sends blue  message to Cell 8 at 20:00:05

Cell 3 sends blue  message to Cell 6 at 20:00:07

Cell 4 sends blue  message to Cell 2 at 20:00:07

Cell 5 sends blue  message to Cell 10 at 20:00:08

Cell 5 sends blue  message to Cell 1 at 20:00:08

Cell 5 sends blue  message to Cell 6 at 20:00:09

Cell 5 sends blue  message to Cell 4 at 20:00:10

Cell 4 sends blue  message to Cell 1 at 20:00:11


 Time taken by * Spanning-Tree Termination Detection * Algorithm is: 17.9607
8 is number of messages sent to detect Termination

Pid     color   isLeaf  hasToken        tokenColor      No_of children  Parent Node
1       blue    0       1               white           0               0
2       blue    0       0               white           0               1
3       blue    0       0               white           0               1
4       blue    0       0               white           0               1
5       blue    0       0               white           0               2
6       blue    0       0               white           0               2
7       blue    1       0               white           0               3
8       blue    1       0               white           0               4
9       blue    1       0               white           0               5
10      blue    1       0               white           0               6
naveen@naveen-Aspire-ES1-572:~/Desktop/Offo$
```

Graphs related to this algorithm:

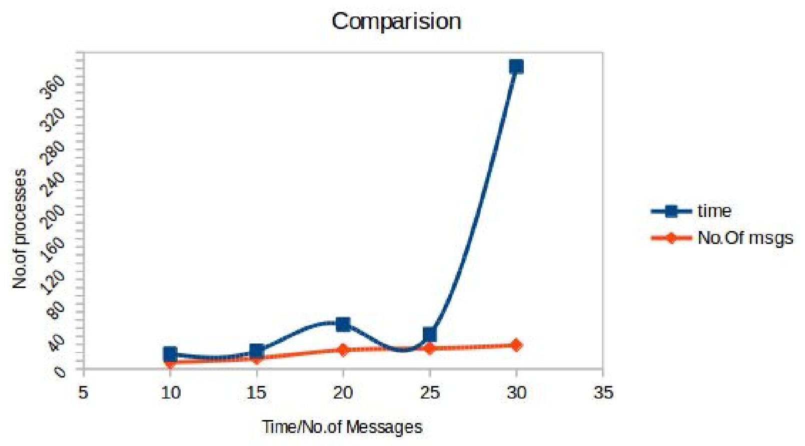
N	*time (in S)*
10	17.9604
15	22.0066
20	54.311
25	42.0399
30	371.964



N	*No.Of msgs*
10	8
15	13
20	23
25	25
30	29



N	*time*	*No.Of msgs*
10	17.9604	8
15	22.0066	13
20	54.311	23
25	42.0399	25
30	371.964	29


Comparision
No.of processes
Time/No.of Messages
time
No.Of msgs