DISTRIBUTED COMPUTING - CS5320

REPORT - ASSIGNMENT 2

-CS16BTCH11030
-SAAHIL SIROWA

● DESIGN AND IMPLEMENTATION DETAILS

1) Dijkstra Scholten Algorithm

There are two threads running per process. One thread implements send functionality and 2nd one implements receive functionality.

```
C = 0, D = 0, dparent = self, childLeft = number of spanning tree child
send():
        Root process initiates computation by sending Red and Blue messages to Ir and Ib cells respectively

        while(exit == false):
                Select a group of neighbors from adjList and send them a coloured message. If color is red increment D by 1.
                sleep()
                If(C == 1 && D == 0 && color is blue):
                        C = 0, dparent = self
                        Send Ack to dparent
                if(C == 0 && D == 0 && color is blue && childLeft == 0)
                        Send Terminate msg to static parent

        While end
        Send shut down msg to all spanning-tree children

receive():
        If blue msg received, process turns blue(passive).

        If red msg received:
                If C == 0:
                        C = 1, color i= red, dparent = sender
                else:
```

```
            Send Ack msg to sender, color = red

    If Ack msg received:
        Decrement D by 1

    If Terminate msg received:
        childLeft -= 1                              //1 child reports termination
        If nodeID is root and childLeft is zero:    //root detects termination
            Exit = true // exit sender thread
            Record time // endTime

    If shut down msg received:
        Exit = true //exit sender. Sender then sends shut down message to its spanning tree children
```

D is difference b/w number of messages sent and acks received.

C is flag denoting computation of the node is still ongoing.

Dparent is dynamic parent. Set when a node receives first message after C becomes zero(woken up another process).

childLeft is number of static spanning tree children.

Terminate message is sent static spanning tree parent, denotes node is done with computation

Acks are a response acknowledging receipt of message

Shut down message is floated when root detects termination

2) Spanning Tree-Based Algorithm

Parent = spanning tree parent, state = color of process(red/black), childLeft = number of children in spanning tree, tokenColor = either black or white

```
send():
    Root process initiates computation by sending Red and Blue messages to Ir and Ib cells respectively.

    while(exit == false || (exit == true && nodeID == root)):
        Select a group of neighbors from adjList and send them a coloured message. If color is red, tokenColor = black.
        sleep()
        If(childLeft is 0, nodeID is root, exit is true):
            If tokenColor is black:        //detection failed
                Send Repeat msg to children
                tokenColor = white, childLeft = # spanning tree children, parent = spanning tree parent
            Else:
                Root detects termination, break

        if(state == blue && childLeft == 0 && parent!= -1)   //send terminate message
            Send terminate message to parent
            parent = -1      //done so it don’t send multiple terminate messages

    While end
    Send shut down msg to all spanning-tree children

receive():
    If blue msg received:
        State = blue(passive)

    If red msg received:
        State = red

    If Terminate msg received:
        If tokenColor recvd is black make tokenColor of process, black //process blackened on receipt of black token
        childLeft -= 1;            //terminate received from one child
        If childLeft is zero and nodeID is root:          //root attempting to detect termination
            If tokenColor is black:            //detection failed
                Send Repeat Message to children
                Reset parent, childLeft, tokenColor to defaults
            Else:   //detection successful
                Exit = true

    If repeat message is received:
        Send Repeat Message to children
        Reset parent, childLeft, tokenColor to defaults

    If shut down msg received:
        Exit = true //exit sender. Sender then sends shut down message to its spanning tree children
```

childLeft is number of static spanning tree children.

tokenColor is either black or white. Black denotes either it has received a black token or sent a message to someone else

State is the color of the process(red/blue)

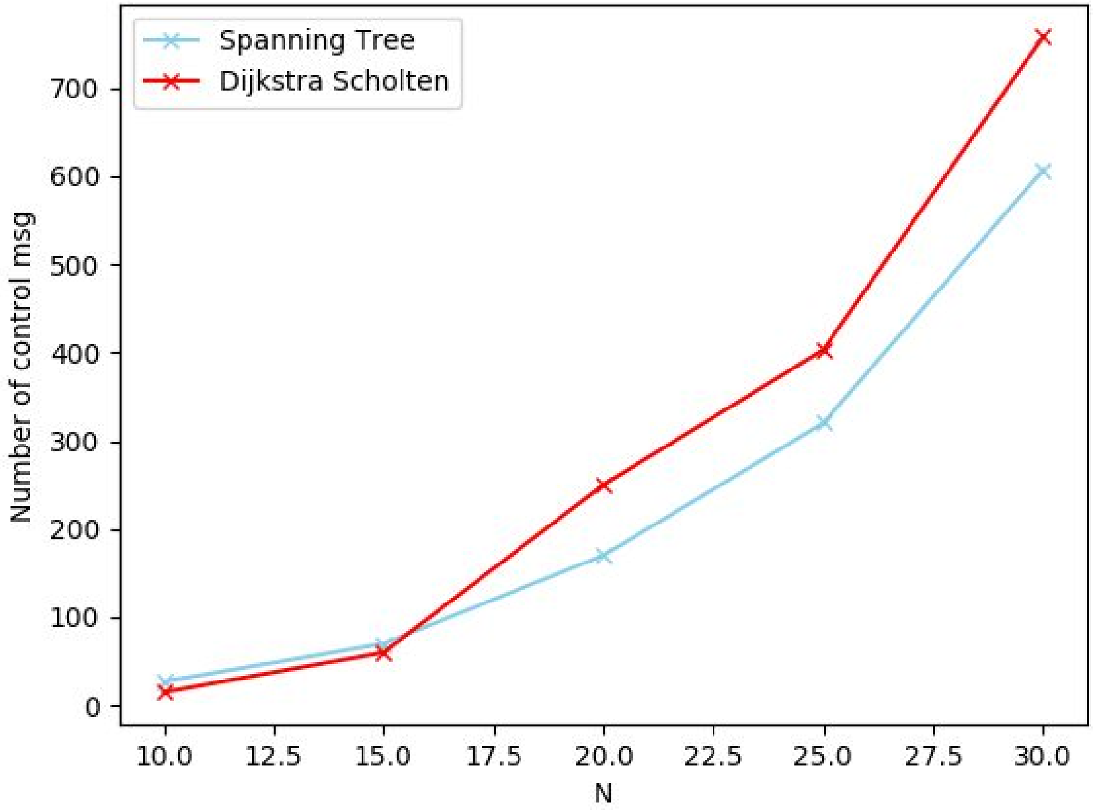
Terminate Message is sent static spanning-tree parent, denotes node is done with computation

Repeats are a sent to restart to the algorithm

Shut down message is floated when root detects termination

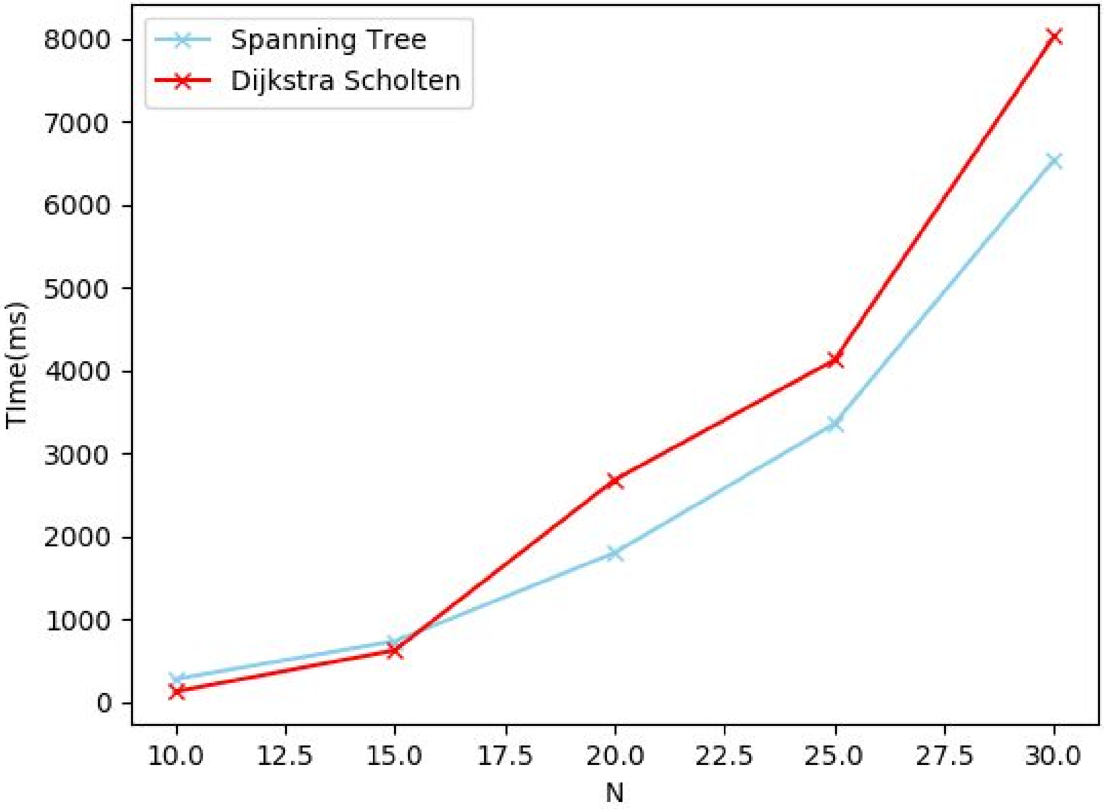
- ## RESULTS AND GRAPHS

1) #Control Msgs vs N



` **Analysis:** It can be observed that for small values of ‘N’ #control msgs is lower for Dijkstra Scholten than Spanning Tree. At higher values of ‘N’ Dijkstra Scholten performs worse than Spanning Tree-based algorithm. Message complexity for Dijkstra Scholten is theta(#basic msgs). It means its lower and upper bounds are of the same order. While the best case complexity for Spanning Tree is Omega(N) and the worst case is O(N*(#basic msgs)). At low values, Dijkstra’s Algorithm performs better because the worst case can be easily observed in small graphs and hence count is dominated by worst case complexities(more repeats than usual). In bigger graphs, it is not easy to tap into worst case scenarios and hence count is dominated by best case complexities(fewer repeats than expected for spanning tree). This results in better performance for Spanning Tree than Dijkstra.

2) Termination Detection Time vs N



Analysis: Initutively time should follow the trend observed in #control messages. At low values, Dijkstra performs better than the spanning tree(more repeats than usual). At high values Spanning Tree performs better(fewer repeats than expected for spanning tree) than Dijkstra.