SIMULATION OF LINK STATE ROUTING PROTOCOL

Link state routing is a technique in which each router shares the knowledge of its neighbourhood with every other router in the internetwork.

Link State Routing has two phases:

Reliable Flooding

- Initial state: Each node knows the cost of its neighbours.
- Final state: Each node knows the entire graph

Route Calculation

Each node uses Dijkstra's algorithm on the graph to calculate the optimal routes to all nodes.

```
set val(stop) 10.0     ;

set ns [new Simulator]

set tracefile [open ques2.tr w]
$ns trace-all $tracefile


set namfile [ open ques2.nam w]
$ns namtrace-all $namfile


set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n4 $n1 100MB 10ms DropTail
$ns queue-limit $n4 $n1 50

$ns duplex-link $n0 $n2 100MB 10ms DropTail
$ns queue-limit $n0 $n2 50

$ns duplex-link $n4 $n3 100MB 10ms DropTail
$ns queue-limit $n4 $n3 50

$ns duplex-link $n1 $n2 100MB 10ms DropTail
$ns queue-limit $n1 $n2 50
```

```
$ns duplex-link $n1 $n3 100MB 10ms DropTail
$ns queue-limit $n1 $n3 50

$ns duplex-link $n2 $n3 100MB 10ms DropTail
$ns queue-limit $n2 $n3 50



$ns cost $n4 $n1 1
$ns cost $n1 $n4 1

$ns cost $n0 $n2 5
$ns cost $n2 $n0 5

$ns cost $n4 $n3 5
$ns cost $n3 $n4 5

$ns cost $n1 $n2 7
$ns cost $n2 $n1 7
$ns cost $n1 $n3 3
$ns cost $n3 $n1 3

$ns cost $n2 $n3 8
$ns cost $n3 $n2 8


set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null1 [new Agent/Null]
$ns attach-agent $n4 $null1

$ns connect $udp0 $null1
$udp0 set packetSize_ 3000



set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 2000
$cbr0 set rate_ 1Mb
$cbr0 set random_ null
$ns at 0.0 "$cbr0 start"
$ns at 5.0 "$cbr0 stop"

#  define protocol
 $ns rtproto LS

 # define 'finish' procedure
proc finish {} {
      global ns tracefile namfile
```

```
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam ques2.nam &
    exit 0
}

# to run the NAM
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns run
```

Case 1:

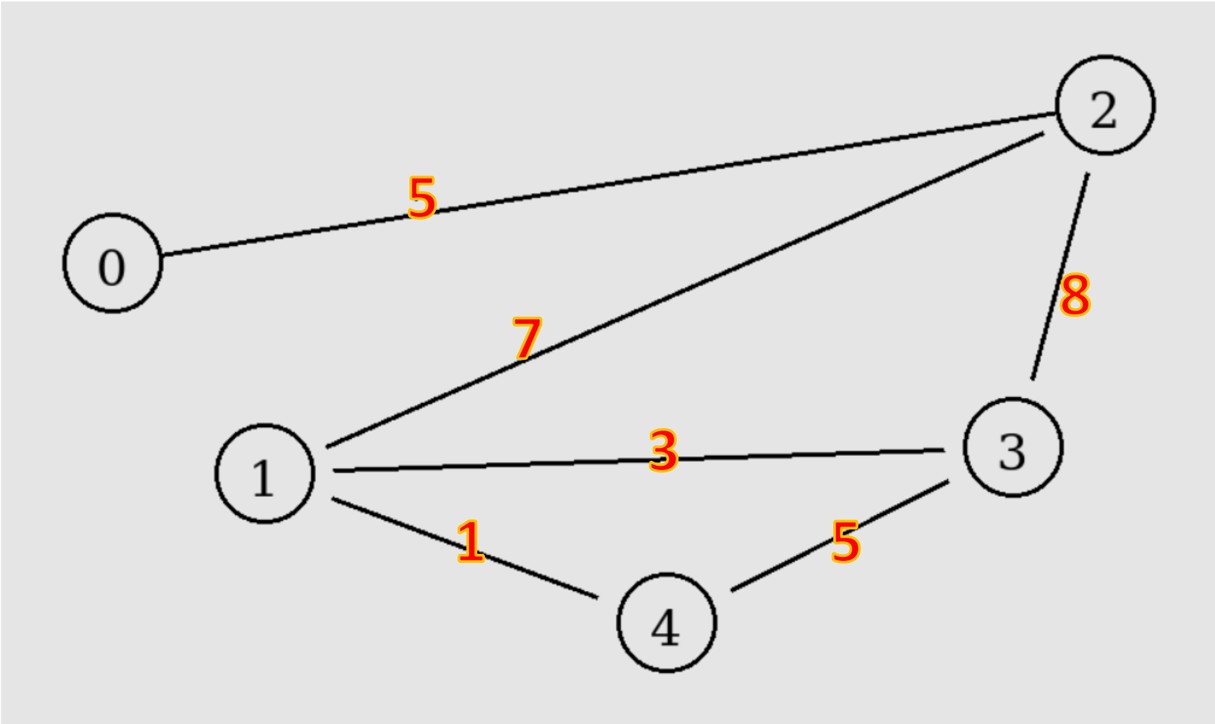
Consider a network having 5 nodes – n0, n1, n2, n3, n4

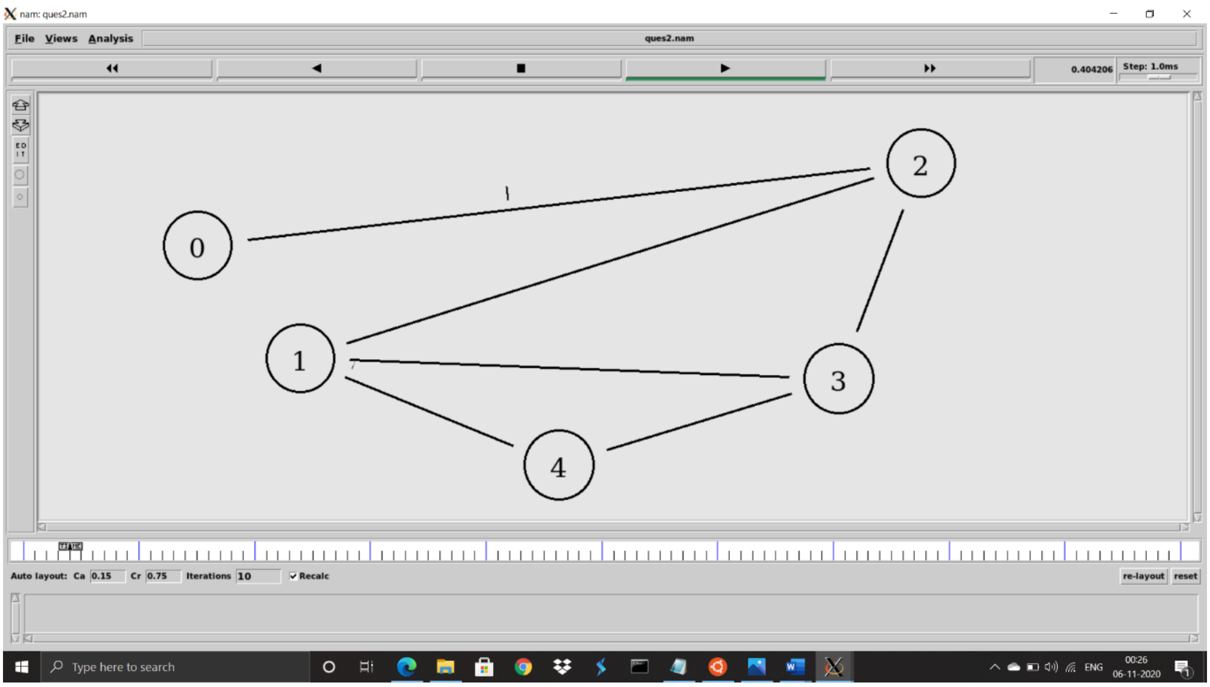
Weights are associated to the links between each node.

1. **Source:** Node n0
2. **Destination:** Node n4

Source Node	Destination Node	Cost
n0	n2	5
n1	n2	7
n1	n3	3
n1	n4	1
n2	n3	8
n3	n4	5

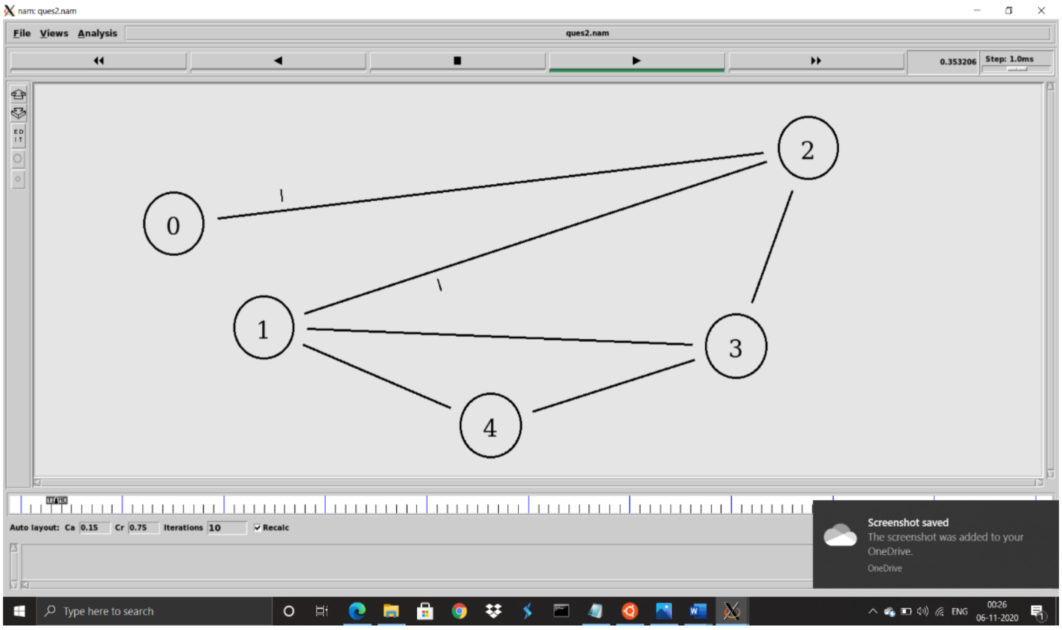
3. Cost Assignment



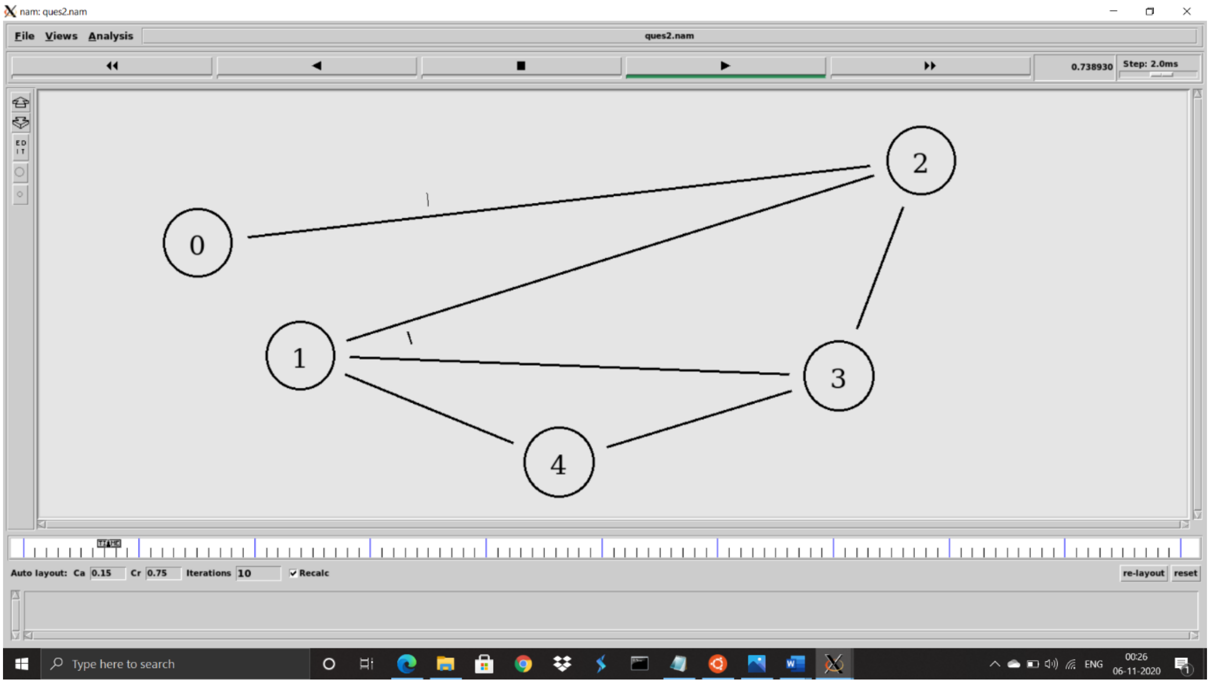


1. Possible Paths between the source and the destination are
    a. n0->>n2->>n3->>n4 cost:18
    b. n0->>n2->>n1->>n4 cost:13
    c. n0->>n2->>n1->>n3->>n4 cost:20

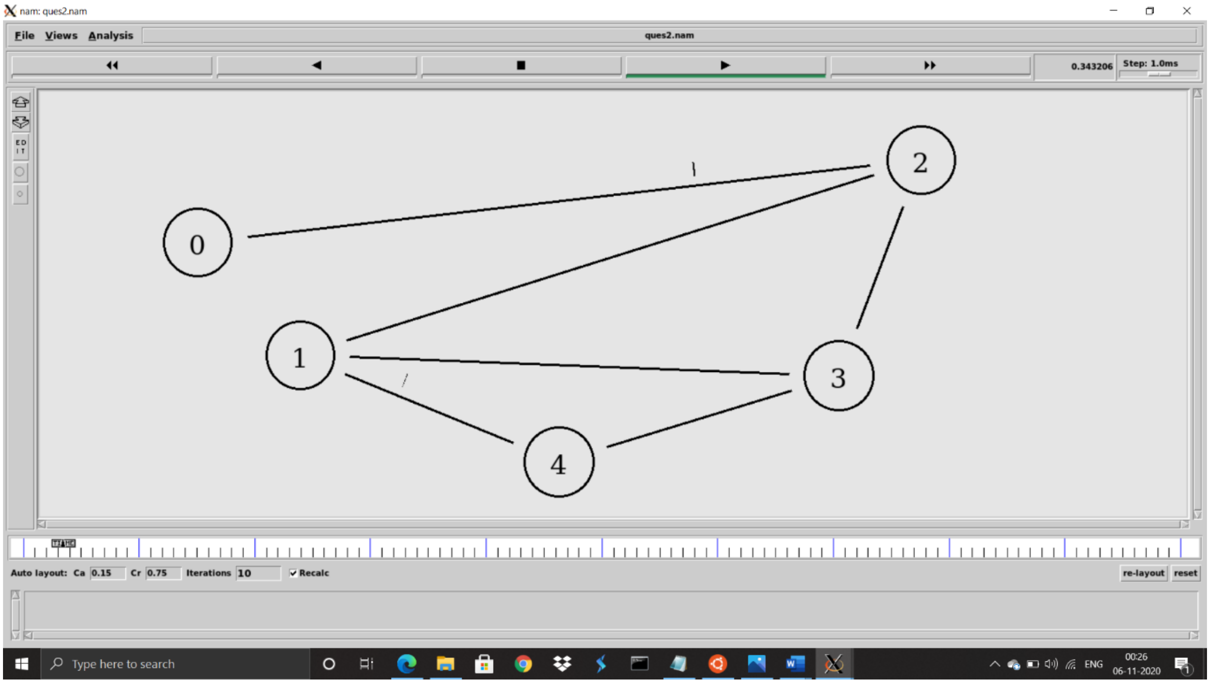
d. n0->>n2->>n3->>n1->>n4 cost:17



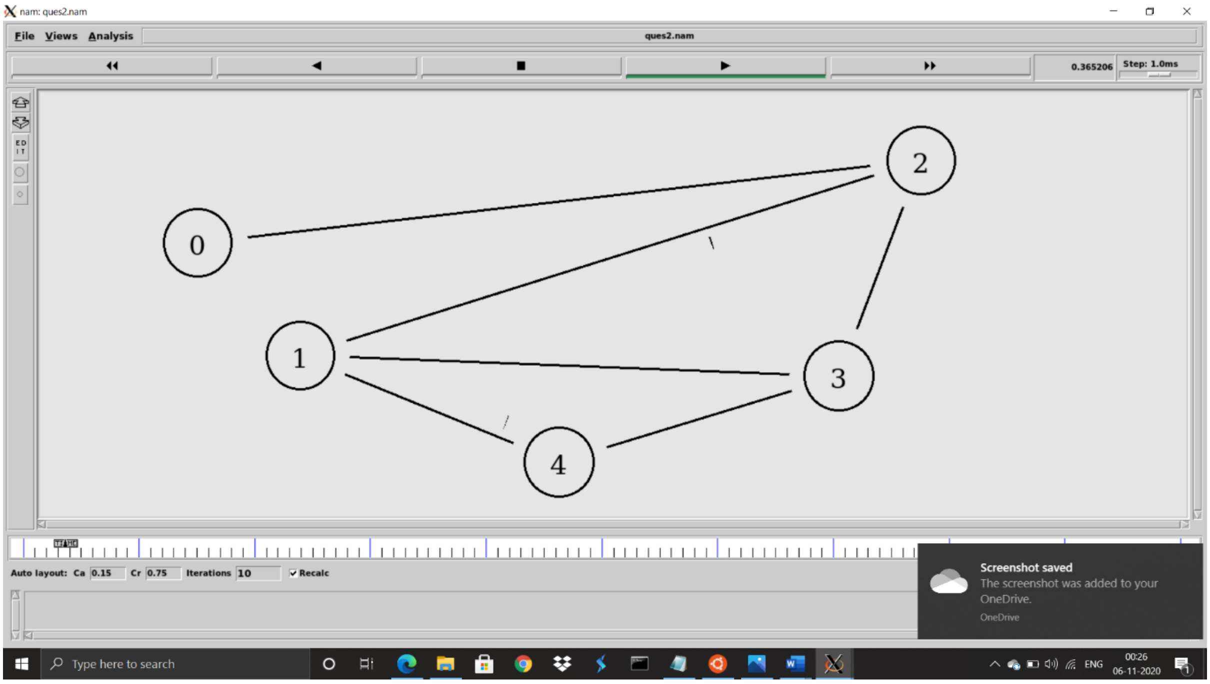
2. Shortest Path according to Link State Algorithm would be **n0->>n2->>n1->>n4 having cost:13**



3. The packets therefore flow from n0 to n2 to n1 to n4 as shown in these diagrams.



4. Thus, the Link state algorithm always choses the optimal path.

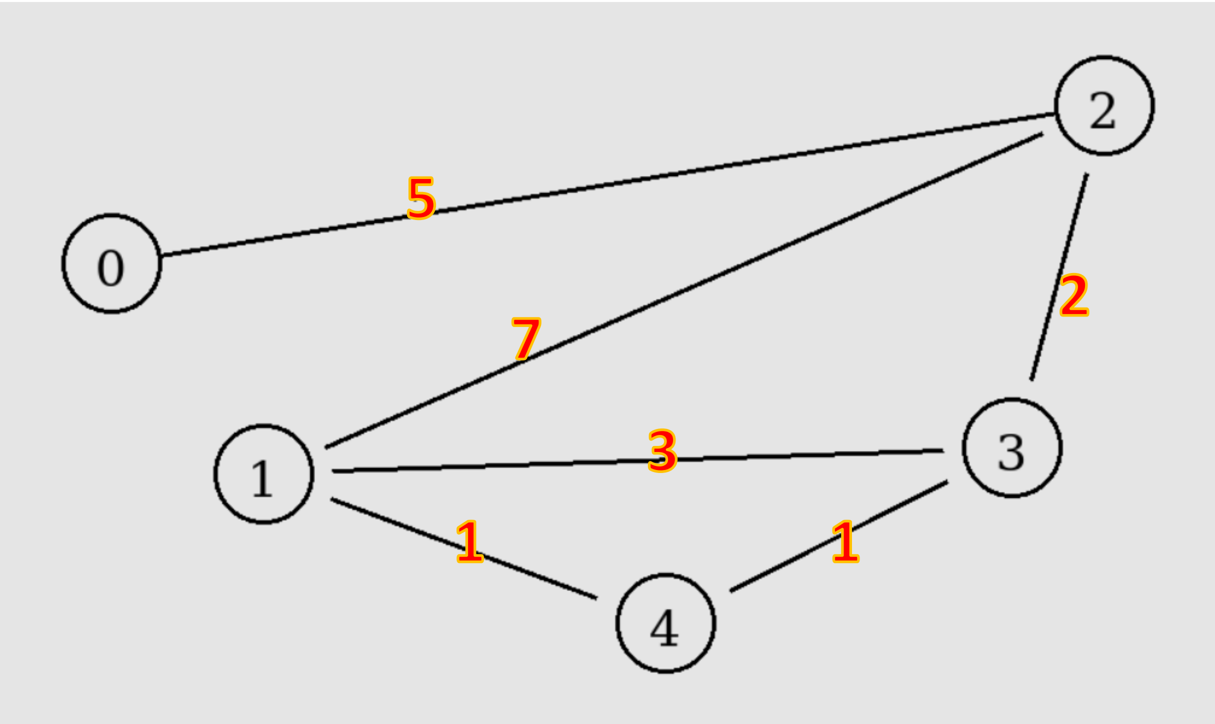


Case 2: Consider the same network as shown previously. Now the costs have been updated. We need to check if Link State Algorithm works well after updating the costs.

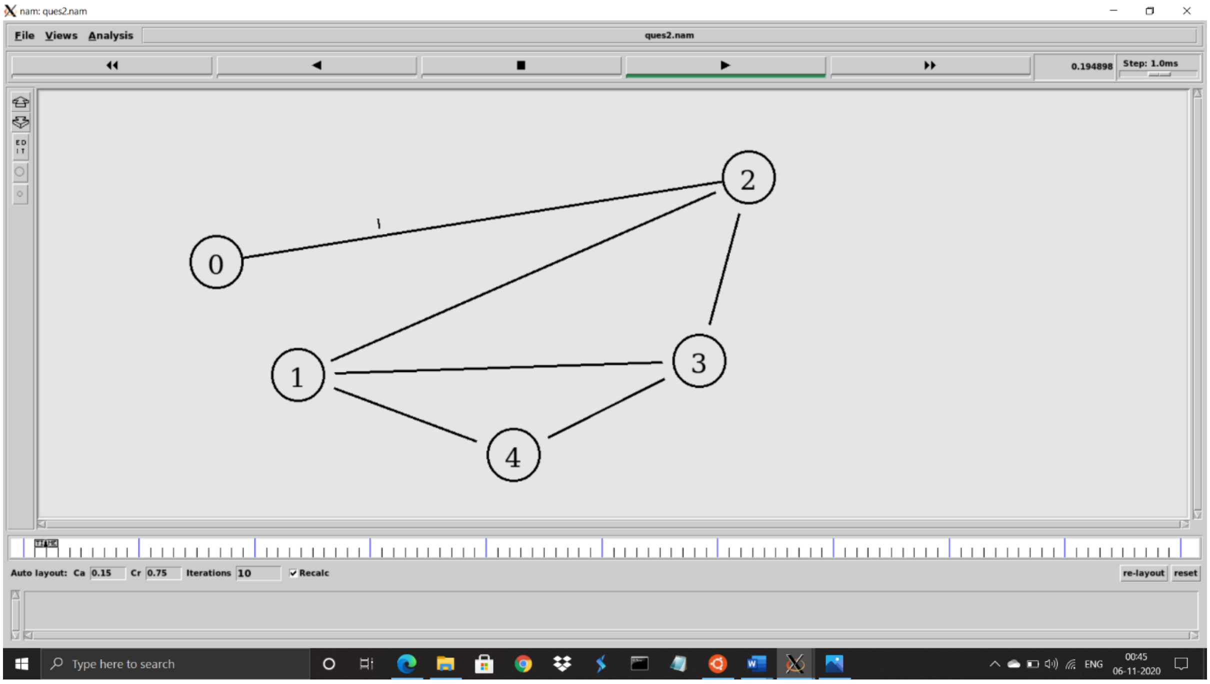
Source Node	Destination Node	Cost
n0	n2	5
n1	n2	7
n1	n3	3
n1	n4	1
n2	n3	2
n3	n4	1

Modified costs

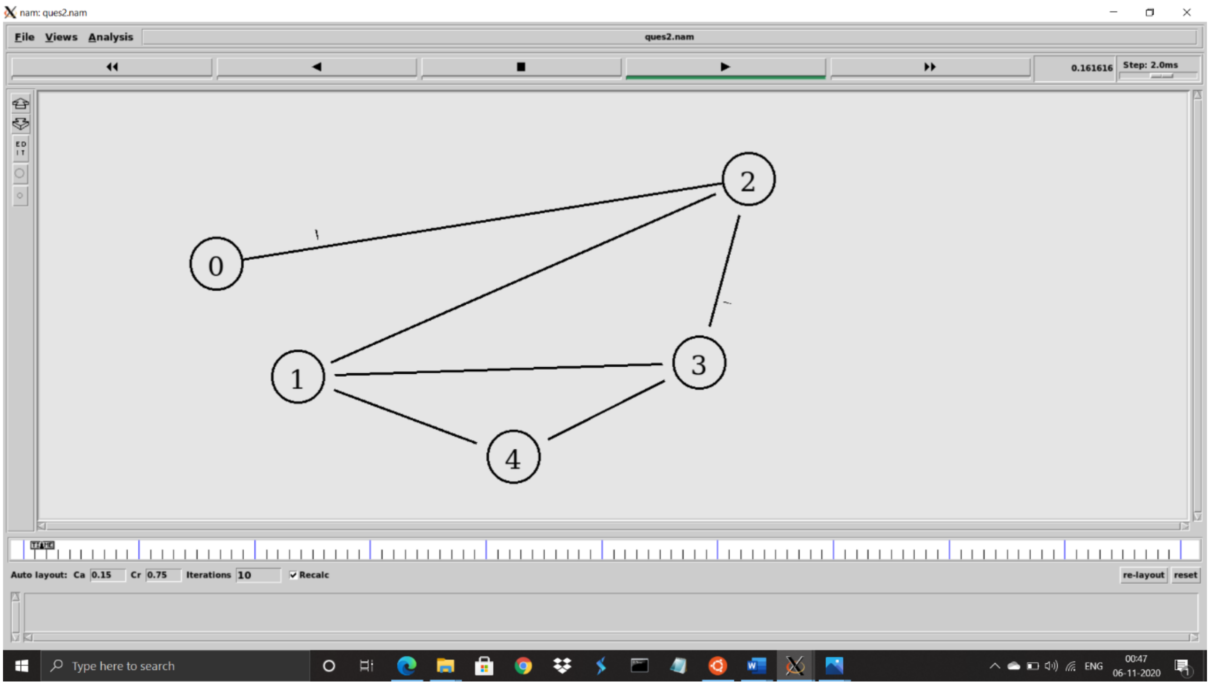
1. **Cost Assignment**



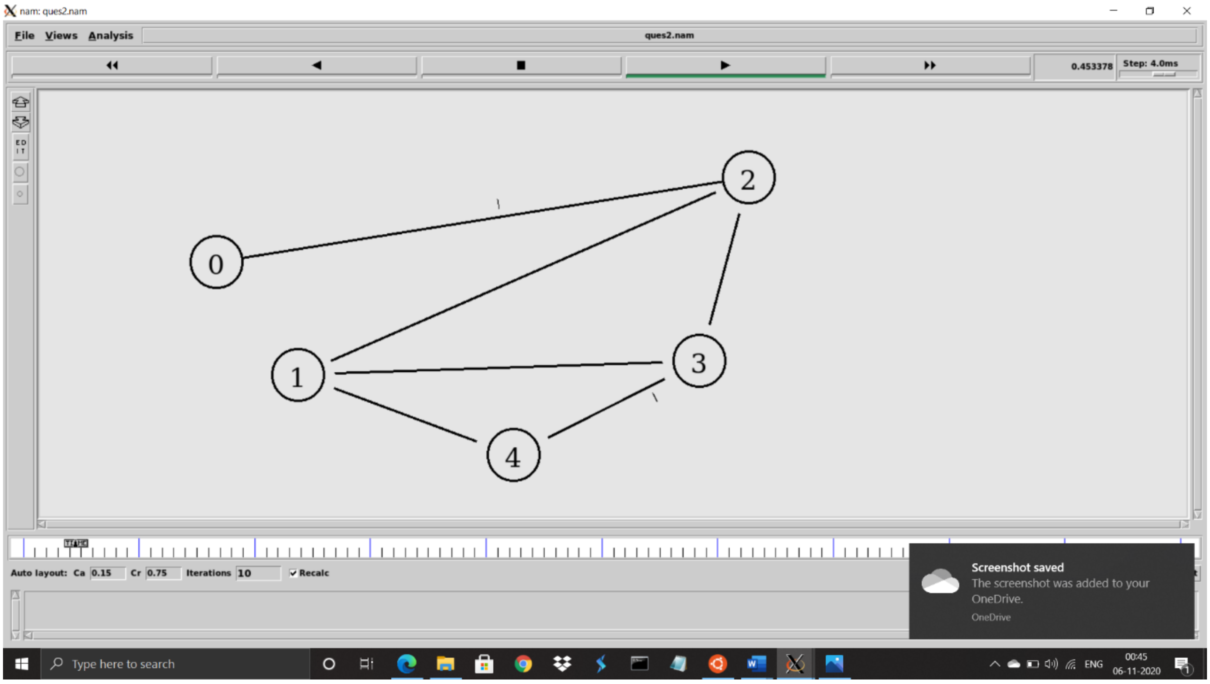
1. Possible Paths between the source and the destination are
   a. **n0->>n2->>n3->>n4 cost:8**
   b. n0->>n2->>n1->>n4 cost:13
   c. n0->>n2->>n1->>n3->>n4 cost:16
   d. n0->>n2->>n3->>n1->>n4 cost:11



2. Shortest Path according to Link State Algorithm would be **n0->>n2->>n3->>n4** **having cost:8**



3. The packets therefore flow from n0 to n2 to n3 to n4 as shown in these diagrams.



4. Thus, the Link state algorithm always choses the optimal path.