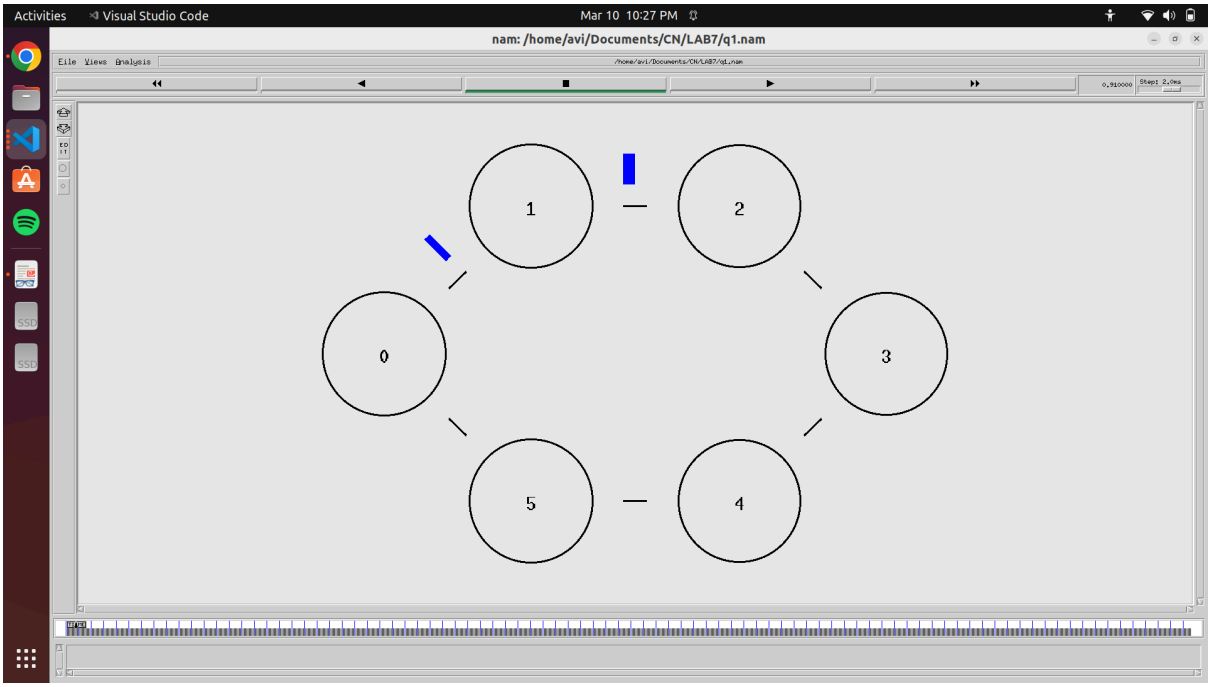
NAME - ABINASH GUPTA
ROLL NO. - 120CS0157
DCCN LAB 7

Q1. Write Tcl script to create scenario and study the performance of token ring protocols through simulation. Create 6 nodes that forms a network numbered from 1 to 6. Create duplex links between the nodes to form a Ring Topology with bandwidth of 100 Mbps and delay of 2ms. Setup TCP Connection between node 1 and node 4. Apply FTP Traffic over TCP. Finish the transmission at 100 sec

TCL Script -

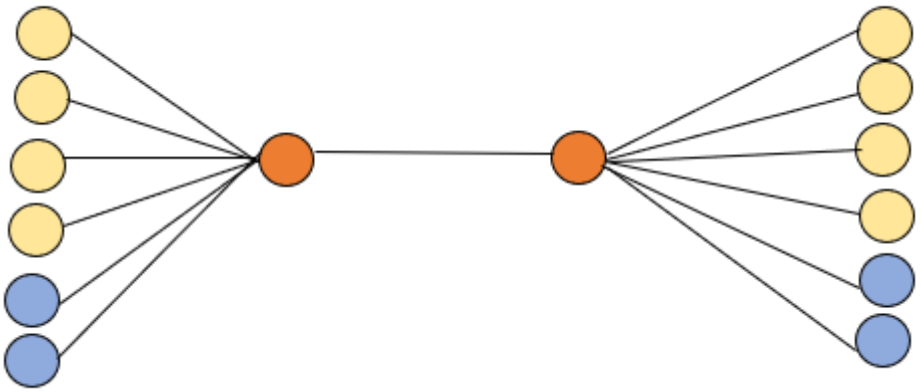
```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows
$ns color 1 Blue


#Define the nam trace file
set nf [open q1.nam w]
$ns namtrace-all $nf

#Define 'finish' procedure
proc finish {} {
   global ns nf
   $ns flush-trace

   #Close trace file
   close $nf

   #Execute nam on the trace file
   exec nam q1.nam &
   exit 0
}

#Create six nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

#Create links between the nodes
```

```
$ns duplex-link $n1 $n2 100Mb 2ms DropTail
$ns duplex-link $n2 $n3 100Mb 2ms DropTail
$ns duplex-link $n3 $n4 100Mb 2ms DropTail
$ns duplex-link $n4 $n5 100Mb 2ms DropTail
$ns duplex-link $n5 $n6 100Mb 2ms DropTail
$ns duplex-link $n6 $n1 100Mb 2ms DropTail

#Give node positions (for NAM)

$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-down
$ns duplex-link-op $n4 $n5 orient left-down
$ns duplex-link-op $n5 $n6 orient left
$ns duplex-link-op $n6 $n1 orient left-up


#Set up a TCP Connections

set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n1 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Set up a FTP over TCP Connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Schedule events for FTP agent
$ns at 0.0 "$ftp start"
$ns at 95.0 "$ftp stop"

#Call the finish procedure
$ns at 100.0 "finish"
#Run simulation

$ns run
```



Q2

Write a Tcl script that forms a network consisting of 6 nodes, numbered from 1 to 6. Each of source and destination has bandwidth of 300 Mbps and delay of 20 ms. Set the bottleneck link bandwidth as 500 sec and delay 10ms. Set the routing protocol to Droptail. Define different colors for different data flows. Send TCP packet from node 1 to node 4 and UDP packet from node 5 to 6. Start the TCP data transmission at 1 sec and UDP at 15 sec. Finish the transmission at 100 sec. Then run nam to view the
results.



Calculate the following performance metrics using awk script:

a) Throughput

b) Delay

c) Packet loss ratio

d) Jain Fairness index.
e) Plot throughput graph using gnuplot (Tahoe vs Reno)
f) Plot Jain Fairness index graph using gnuplot

Solution :-

TCL Script

```
set ns [new Simulator]

$ns color 1 Blue
$ns color 2 Red
$ns color 3 Darkgreen
$ns color 4 Orange
$ns color 5 Darkviolet
$ns color 6 Brown

set nf [open Ex.nam w]
$ns namtrace-all $nf

set tf [open Ex.tr w]
$ns trace-all $tf

proc finish {} {
   global ns nf tf
   $ns flush-trace
   close $nf
   close $tf
   exec nam Ex.nam &
   exec awk -f Ex.awk Ex.tr &
   #exec awk -f Thrpt.awk Ex.tr > Th.tr &
   exit 0
}


set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
```

```
set n10 [$ns node]
set n11 [$ns node]
set n12 [$ns node]
set n13 [$ns node]

$ns at 0.0 "$n0 color blue"
$ns at 0.0 "$n1 color red"
$ns at 0.0 "$n2 color darkgreen"
$ns at 0.0 "$n3 color orange"
$ns at 0.0 "$n4 color darkviolet"
$ns at 0.0 "$n5 color brown"
$ns at 0.0 "$n6 color blue"
$ns at 0.0 "$n7 color red"
$ns at 0.0 "$n8 color darkgreen"
$ns at 0.0 "$n9 color orange"
$ns at 0.0 "$n10 color darkviolet"
$ns at 0.0 "$n11 color brown"

$ns at 0.0 "$n0 label \"Node #1(SENDER)\""
$ns at 0.0 "$n1 label \"Node #2(SENDER)\""
$ns at 0.0 "$n2 label \"Node #3(SENDER)\""
$ns at 0.0 "$n3 label \"Node #4(SENDER)\""
$ns at 0.0 "$n4 label \"Node #5(SENDER)\""
$ns at 0.0 "$n5 label \"Node #6(SENDER)\""
$ns at 0.0 "$n6 label \"Node #1(RECEIVER)\""
$ns at 0.0 "$n7 label \"Node #2(RECEIVER)\""
$ns at 0.0 "$n8 label \"Node #3(RECEIVER)\""
$ns at 0.0 "$n9 label \"Node #4(RECEIVER)\""
$ns at 0.0 "$n10 label \"Node #5(RECEIVER)\""
$ns at 0.0 "$n11 label \"Node #6(RECEIVER)\""

$ns duplex-link $n0 $n12 300Mb 20ms DropTail
$ns duplex-link-op $n0 $n12 orient 285deg

$ns duplex-link $n1 $n12 300Mb 20ms DropTail
$ns duplex-link-op $n1 $n12 orient 315deg

$ns duplex-link $n2 $n12 300Mb 20ms DropTail
$ns duplex-link-op $n2 $n12 orient 345deg

$ns duplex-link $n3 $n12 300Mb 20ms DropTail
$ns duplex-link-op $n3 $n12 orient 15deg
```

```
$ns duplex-link $n4 $n12 300Mb 20ms DropTail
$ns duplex-link-op $n4 $n12 orient 45deg

$ns duplex-link $n5 $n12 300Mb 20ms DropTail
$ns duplex-link-op $n5 $n12 orient 75deg

$ns duplex-link $n12 $n13 1Mb 10ms DropTail
$ns duplex-link-op $n12 $n13 orient right

$ns duplex-link $n13 $n6 300Mb 20ms DropTail
$ns duplex-link-op $n13 $n6 orient 75deg

$ns duplex-link $n13 $n7 300Mb 20ms DropTail
$ns duplex-link-op $n13 $n7 orient 45deg

$ns duplex-link $n13 $n8 300Mb 20ms DropTail
$ns duplex-link-op $n13 $n8 orient 15deg

$ns duplex-link $n13 $n9 300Mb 20ms DropTail
$ns duplex-link-op $n13 $n9 orient 345deg

$ns duplex-link $n13 $n10 300Mb 20ms DropTail
$ns duplex-link-op $n13 $n10 orient 315deg

$ns duplex-link $n13 $n11 300Mb 20ms DropTail
$ns duplex-link-op $n13 $n11 orient 285deg

$ns duplex-link-op $n12 $n13 queuePos 0.5

$ns queue-limit $n12 $n13 4

set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n6 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 1
$tcp1 set window_ 8
$tcp1 set ssthresh_ 3
$tcp1 set windowInit_ 1
```

```
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set tcp2 [new Agent/TCP]
$ns attach-agent $n1 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n7 $sink2
$ns connect $tcp2 $sink2
$tcp2 set fid_ 2
$tcp2 set window_ 8
$tcp2 set ssthresh_ 3
$tcp2 set windowInit_ 1

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

set tcp3 [new Agent/TCP]
$ns attach-agent $n2 $tcp3
set sink3 [new Agent/TCPSink]
$ns attach-agent $n8 $sink3
$ns connect $tcp3 $sink3
$tcp3 set fid_ 3
$tcp3 set window_ 8
$tcp3 set ssthresh_ 3
$tcp3 set windowInit_ 1


set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3

set tcp4 [new Agent/TCP]
$ns attach-agent $n3 $tcp4
set sink4 [new Agent/TCPSink]
$ns attach-agent $n9 $sink4
$ns connect $tcp4 $sink4
$tcp4 set fid_ 4
$tcp4 set window_ 8
$tcp4 set ssthresh_ 3
$tcp4 set windowInit_ 1

set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
```

```
set udp5 [new Agent/UDP]
$ns attach-agent $n4 $udp5
set null5 [new Agent/Null]
$ns attach-agent $n10 $null5
$ns connect $udp5 $null5
$udp5 set fid_ 5

set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packet_size_ 1000

set udp6 [new Agent/UDP]
$ns attach-agent $n5 $udp6
set null6 [new Agent/Null]
$ns attach-agent $n11 $null6
$ns connect $udp6 $null6
$udp6 set fid_ 6

set cbr6 [new Application/Traffic/CBR]
$cbr6 attach-agent $udp6
$cbr6 set packet_size_ 1000


set f_cwnd [open cwnd.tr w]

proc Record {} {
   global f_cwnd tcp1 tcp2 tcp3 tcp4 ns
   set intval 0.1
   set now [$ns now]
   set cwnd1 [$tcp1 set cwnd_]
   set cwnd2 [$tcp2 set cwnd_]
   set cwnd3 [$tcp3 set cwnd_]
   set cwnd4 [$tcp4 set cwnd_]



   puts $f_cwnd "$now $cwnd1  $cwnd2  $cwnd3 $cwnd4"
   $ns at [expr $now + $intval] "Record"
}

$ns at 0.1 "Record"
$ns at 0.0 "$ftp1 start"
$ns at 0.0 "$ftp2 start"
```

```
$ns at 0.0 "$ftp3 start"
$ns at 0.0 "$ftp4 start"
$ns at 0.0 "$cbr5 start"
$ns at 0.0 "$cbr6 start"

$ns at 10.0 "finish"

$ns run
```

Delay.awk (for Avg Delay calculation)

```
BEGIN{
   receiveNum = 0;
}

{
   event = $1
   time = $2
   from_node = $3;
   to_node = $4;
   pkt_type = $5;
   src_addr = $9;
   dest_addr = $10;
   pkt_id = $12

   if (event == "+" && pkt_type != "ack")
   {
       fro = int(from_node);
       src = int(src_addr);
       if (fro == src)
       {
           sendTime[pkt_id] = time
       }
   }


   if (event == "r" && pkt_type != "ack")
   {
       to = int(to_node);
       dst = int(dest_addr);
       if (to == dst)
       {
           receiveNum++
           recvTime[pkt_id] = time
```

```
            delay[pkt_id] = recvTime[pkt_id] - sendTime[pkt_id];
            tot_delay += delay[pkt_id];
        }
    }
}

END{
    if (receiveNum != 0)
    {
        avg_delay = tot_delay / receiveNum
    }
    else
    {
        avg_delay = 0
    }
    printf("Total Delay : %f\n",tot_delay);
    printf("Received packets : %f\n",receiveNum );
    printf("avgDelay: %f s\n", avg_delay)
    printf("overall delay: %f s\n", tot_delay)
    printf("overall packet: %d\n", receiveNum)
}
```

Throughput.awk (For Avg. Throughput calculation)

```
#!/usr/bin/awk -f

BEGIN {
    # Set default values for variables
    tcp_byte_count = 0
    udp_byte_count = 0
    total_start_time = 0
    total_end_time = 0
}

# Process each packet in the trace file
{
    # Check if the packet is TCP or UDP
    if ($5 == "tcp")
    {
        tcp_byte_count += $6

        if (tcp_start_time == 0 || $2 < tcp_start_time)
        {
```

```
            tcp_start_time = $2
        }
        if ($2 > tcp_end_time)
        {
            tcp_end_time = $2
        }
    }
    else if ($5 == "cbr")
    {
        udp_byte_count += $6

        if (udp_start_time == 0 || $2 < udp_start_time)
        {
            udp_start_time = $2
        }
        if ($2 > udp_end_time)
        {
            udp_end_time = $2
        }
    }

    if (total_start_time == 0 || $2 < total_start_time)
    {
        total_start_time = $2
    }
    if ($2 > total_end_time)
    {
        total_end_time = $2
    }
}

# Print final throughput if needed
END {
    if (tcp_byte_count > 0 || udp_byte_count > 0)
    {
        total_time = total_end_time - total_start_time
        tcp_time = tcp_end_time - tcp_start_time
        udp_time = udp_end_time - udp_start_time

        BC = tcp_byte_count + udp_byte_count

        avg_throughput = BC * 8 / total_time / 1000000
        tcp_throughput = tcp_byte_count * 8 / tcp_time / 1000000
```

```
        udp_throughput = udp_byte_count * 8 / udp_time / 1000000

        printf("Avg throughput: %.2f Mbps\n", avg_throughput)
        printf("Start time: %f s\n", total_start_time)
        printf("End time: %f s\n", total_end_time)
        printf("Total time: %f s\n", total_time)

        printf("\nTCP throughput: %.2f Mbps\n", tcp_throughput)
        printf("Start time: %f s\n", tcp_start_time)
        printf("End time: %f s\n", tcp_end_time)
        printf("Total time: %f s\n", tcp_time)

        printf("\nUDP throughput: %.2f Mbps\n", udp_throughput)
        printf("Start time: %f s\n", udp_start_time)
        printf("End time: %f s\n", udp_end_time)
        printf("Total time: %f s\n", udp_time)
    }
}
```

Loss_ratio.awk (To calculate loss ratio)

```
BEGIN {
    send=0;
    received=0;
    dropped=0;
}

{
# Trace line format: normal
    event = $1;
    time = $2;
    from_node = $3;
    to_node = $4;
    pkt_type = $5;
    pkt_size = $6;
    flgs = $7;
    f_id = $8;
    src_addr = $9;
    dest_addr = $10;
    seq_no = $11;
    pkt_id = $12;
#packet delivery ratio
    if (event == "+" && pkt_type != "ack")
```

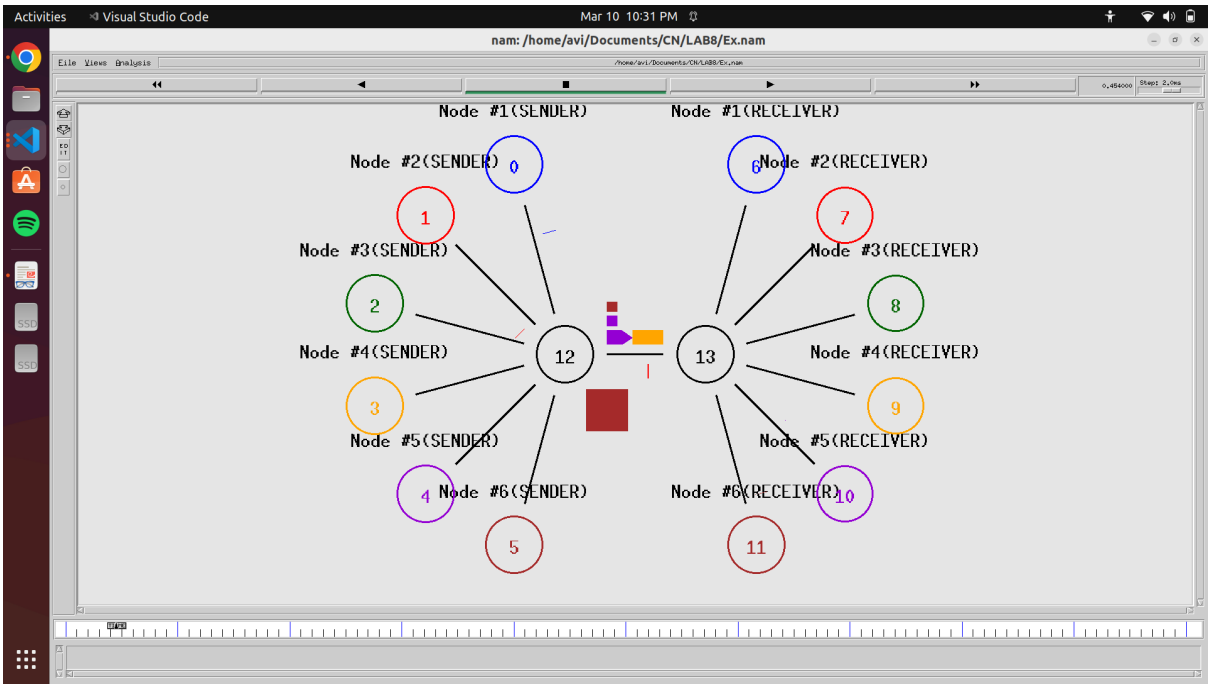
```
    {
        fro = int(from_node);
        src = int(src_addr);
        if (fro == src)
        {
            send++
        }
    }

    if (event == "r" && pkt_type != "ack")
    {
        to = int(to_node);
        dst = int(dest_addr);
        if (to == dst)
        {
            received++
        }
    }

    if (event == "d")
    {
        dropped++;
    }
}
 END{
    print "\nGeneratedPackets = " send;
    print "ReceivedPackets = " received;
    print "Total Dropped Packets = " dropped;
    print ("\nPacket Delivery Ratio = ", (received/send)*100" %");
    print ("Packet Loss Ratio = ", (dropped/send)*100" %");
}
```

Jain Index Calculation :

```
BEGIN{
    sum = 0;
    Sq_sum = 0;
    cnt = 0;
    JI = 0;
}

{
    event = $1
```

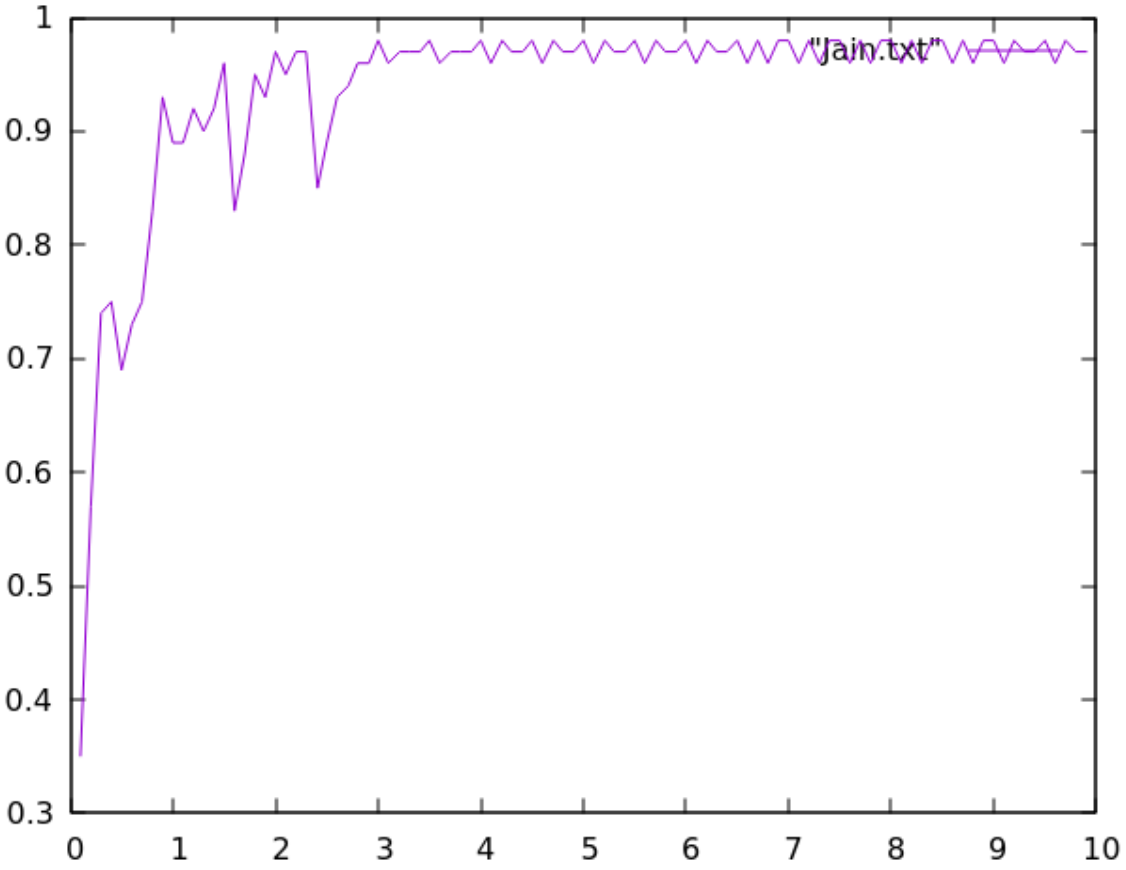
```
    time = $2
    from_node = $3;
    to_node = $4;
    pkt_type = $5;
    pkt_size = $6;
    f_id = $8;
    src_addr = $9;
    dest_addr = $10;
    pkt_id = $12

    if (event == "+" && pkt_type != "ack")
    {
        fro = int(from_node);
        src = int(src_addr);
        if (fro == src)
        {
            if (sendTime[f_id] == 0 || time < sendTime[f_id])
            {
                sendTime[f_id] = time
            }
        }
    }


    if (event == "r" && pkt_type != "ack")
    {
        to = int(to_node);
        dst = int(dest_addr);
        if (to == dst)
        {
            if (time > recvTime[f_id])
            {
                recvTime[f_id] = time
            }
            recvTime[f_id] = time
            node_thr[f_id] += pkt_size;
        }
    }
}

END{
    for (i in node_thr)
    {
```

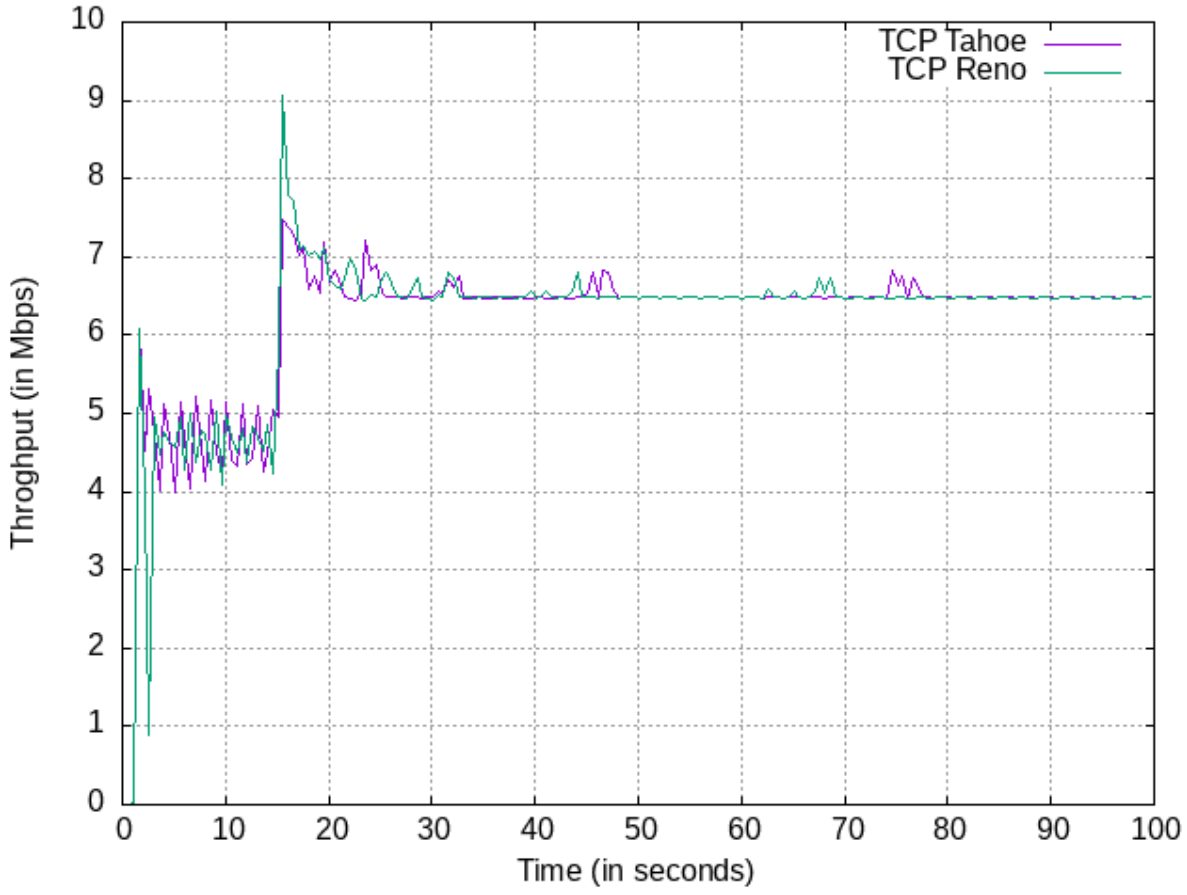
```
        Th = (node_thr[i]/(recvTime[i] - sendTime[i]))*(8/1000);
        sum += Th;
        Sq_sum += (Th*Th);
        cnt++;
    }
    JI = ((sum*sum)/(cnt*Sq_sum));
    printf("\nJain Index = %f\n", JI);
}
```



Output -

```
Sent Packets = 4003
Received Packets = 3979
Dropped Packets = 12
Packet Delivery Ratio = 99.400450
Packet Loss Ratio = 0.299775
Sent = 4114.32 KB
Received = 4089.52 KB
Start = 0.00 s
End = 10.00 s
Throughtput = 3.27 Kbps
Jain Index = 0.98
```


"jain.txt"
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
1
2
3
4
5
6
7
8
9
10


TCP Tahoe
TCP Reno
Throghput (in Mbps)
Time (in seconds)
0
1
2
3
4
5
6
7
8
9
10
0
10
20
30
40
50
60
70
80
90
100