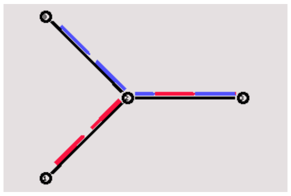
PRACTICAL -7(A)

AIM: Design simple tcl script for wired topology of4 nodes in NS-2 and analyze various tcl parameters like network nodes, links, and queuesand topology. Queue Size: - 5, Duplex Link, Queue Type Droptail.



Link	Bandwidth	Delay
no-n2	10Mbps	10ms
n1-n2	10Mbps	10ms
n2-n3	5Mbps	10ms

ftp0:
Packet Size: 1000
Rate: 1
Interval: 150

THEORY:

❖ NS2:-

- ✓ NS2 is a popular network simulator tool.
- ✓ Ns2 is supported by the operating systems falling under Linux distributions, and Mac operating system.
- ✓ It is a discrete event simulator whose parent software distribution is *NS (Network Simulator)*. NS is available in 3 versions based on variations of features provided by them.
- ✓ NS is commonly used for research and teaching purpose on topics such as topologies, protocol simulations, routing and multicasting protocols.

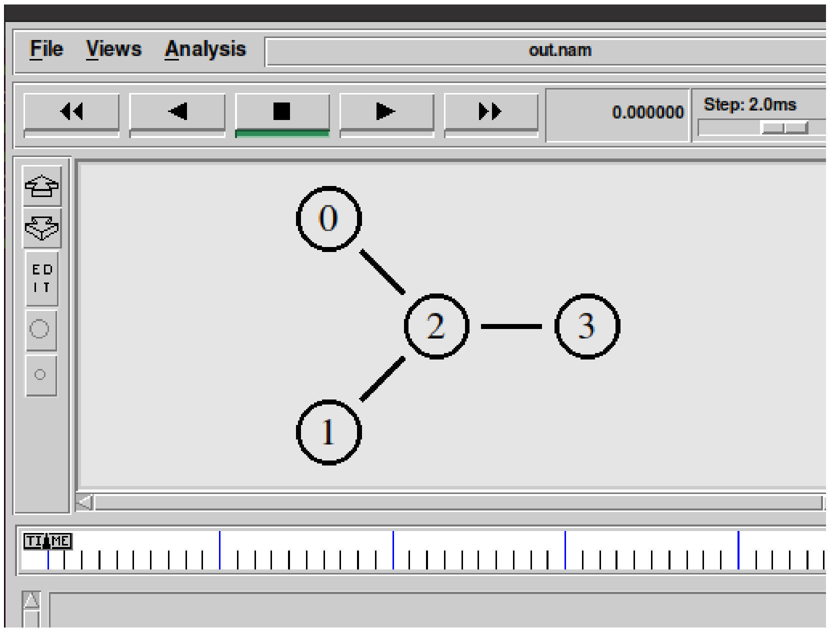
❖ **Duplex Link:-** Duplex link is a link type to connect multiple nodes with each other in the network simulator.

❖ **Queue:-**Queue type corresponds to the type of link between devices based on the bandwidth and packet size that different queues support.

- **Nodes:-**Nodes can be referenced as a virtual representation of any device present in actual topology.

- **Use of NS2:-** NS2 is an advanced version of Network Simulator (NS). It is mostly popular for its services to assemble and visualize virtual networks for simulation. It is better than cisco packet tracer as each device in the connection acts as a node in NS2 while a specific element characteristics needs to be defined in cisco (i.e. if a node is router, switch ,etc.). There are many other similar features which make NS2 more popular to use.

TOPOLOGY:-



PROGRAM CODE:-

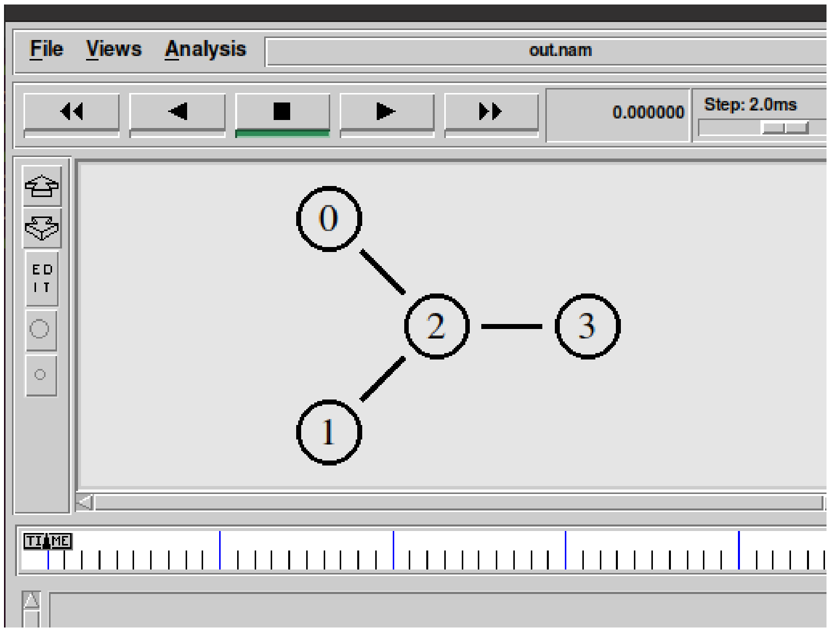
```
set ns [new Simulator]
```

```
$ns color 1 Blue
$ns color 2 Red

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

proc finish {} {
	global ns nf
	$ns flush-trace
	close $nf

	exec namout.nam&
	exit 0
}

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

$ns duplex-link $n0 $n2 10Mb 10ms DropTail
$ns duplex-link $n1 $n2 10Mb 10ms DropTail
$ns duplex-link $n2 $n3 5Mb 10ms DropTail

$ns queue-limit $n2 $n3 5

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n2 $n3 queuePos 0.5

set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

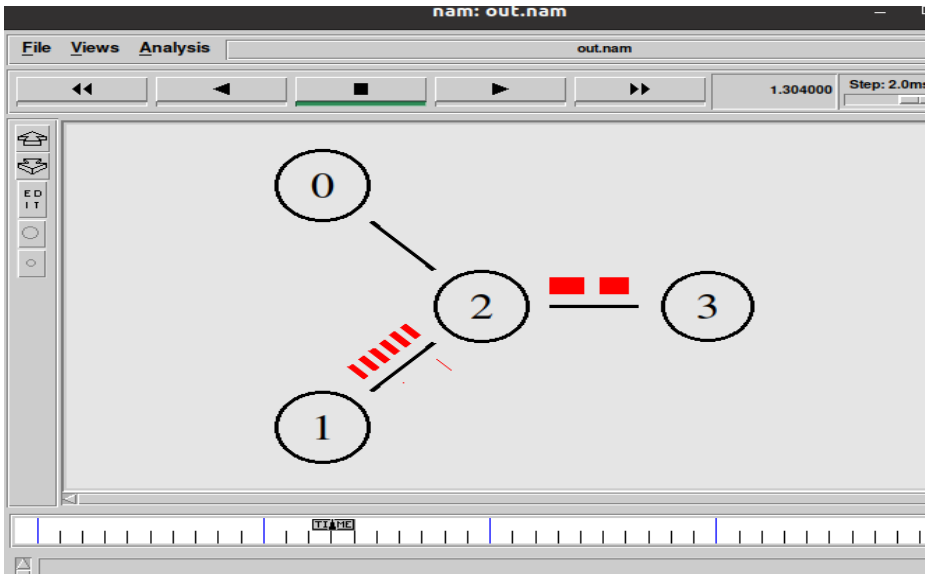
```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
$ftp set packet_size_ 1000
$ftp set rate_ 1mb

set tcp [new Agent/TCP]
$tcp set class_ 1
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 2


set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
$ftp set packet_size_ 1000
$ftp set rate_ 1mb

$ns at 0.5 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 5.0 "finish"
$ns run
```

OUTPUT:-

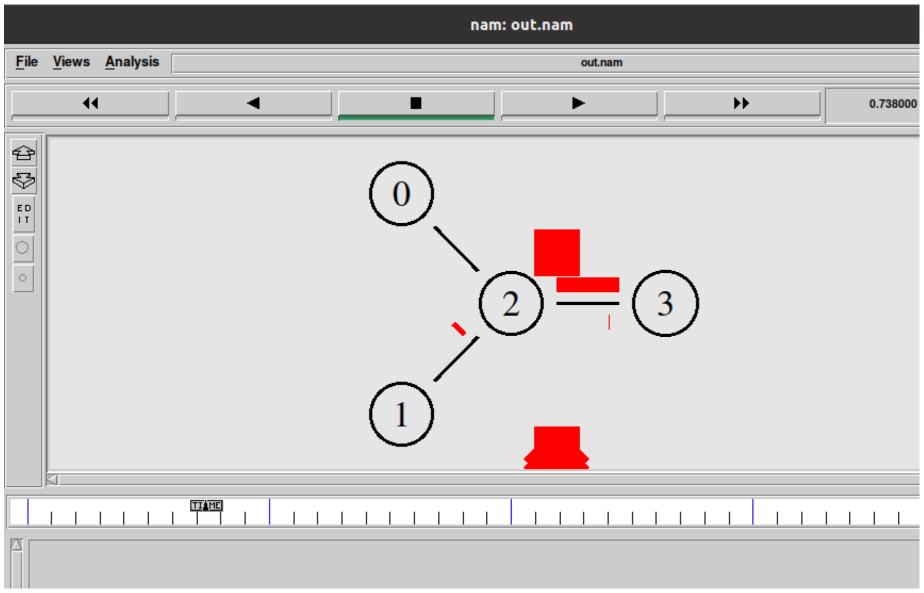
{1} Network Topology:-



{2} Data transfer in progress between nodes 1 &3:-



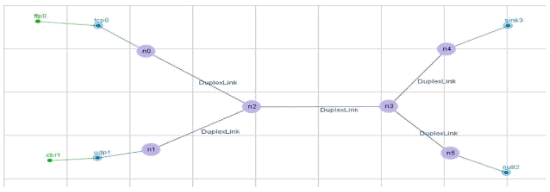
{3} Packet Drop due to excess packet in queue (i.e. more packets than queue capacity):-



CONCLUSION: In this practical basics of network simulator 2 are covered. Also we have learnt how to establish tcp connection using ftp protocol using ns2 programming. Also introduction to different networking terms in network simulator such as queue, nodes, link, etc have been made.

PRACTICAL -7(B)

AIM: Design simple tcl script for wired topology of6 nodes in NS-2 and analyze various tcl parameters like network nodes, links, queuesand topology.



Set the following parameters for Duplex Link:

Link	Bandwidth	Delay	Queue Type	Queue Size
no-n2	10Mbps	10ms	RED	10
n1-n2	10Mbps	10ms	RED	10
n2-n3	5Mbps	???	RED	???
n3-n4	10Mbps	10ms	RED	10
n3-n5	10Mbps	10ms	RED	10

ftp0:- (Both node with ftp)
Packet Size: 1000
Rate: 1
Interval: 150
cbr0:- (Both node with cbr)
Packet Size: 1500
Rate: 0.05
Interval: 150

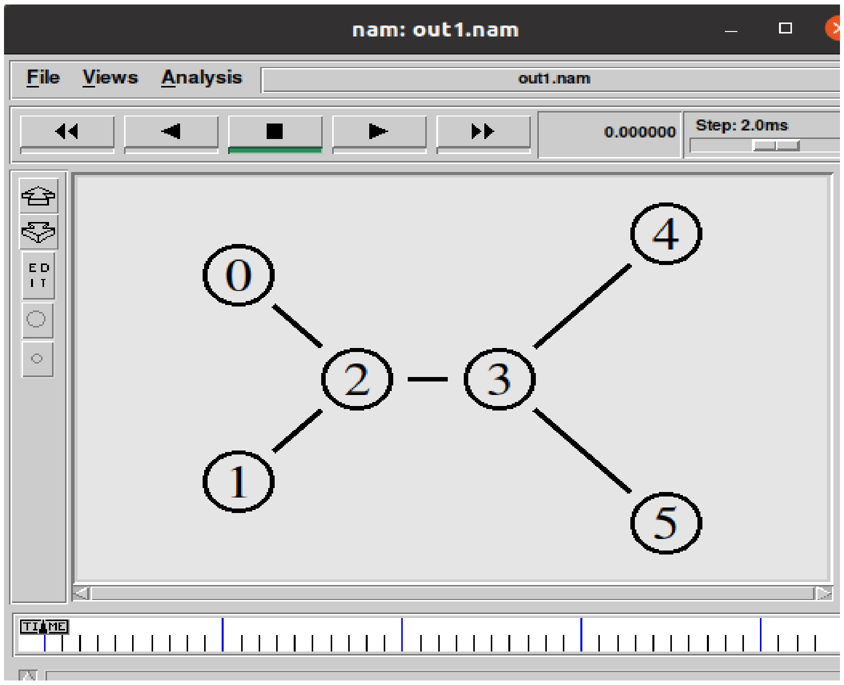
<u>THEORY</u>:

❖ Difference Between TCP and UDP connection:-

✓ TCP is a connection oriented protocol. While, UDP is a connectionless protocol.

- ✓ As TCP provides error checking support and also guarantees delivery of data to the destination router this make it more reliable as compared to UDP. UDP does not provide error control.
- ✓ TCP also supports sequencing which means that data packets follow only one path while getting transmitted. While sequencing is not followed by UDP.
- ✓ TCP is slow and thus is not much efficient as far as transmission speed of data and its performance are concerned. On the other hand UDP is much faster and is thus more efficient.

TOPOLOGY:-



PROGRAM CODE:-

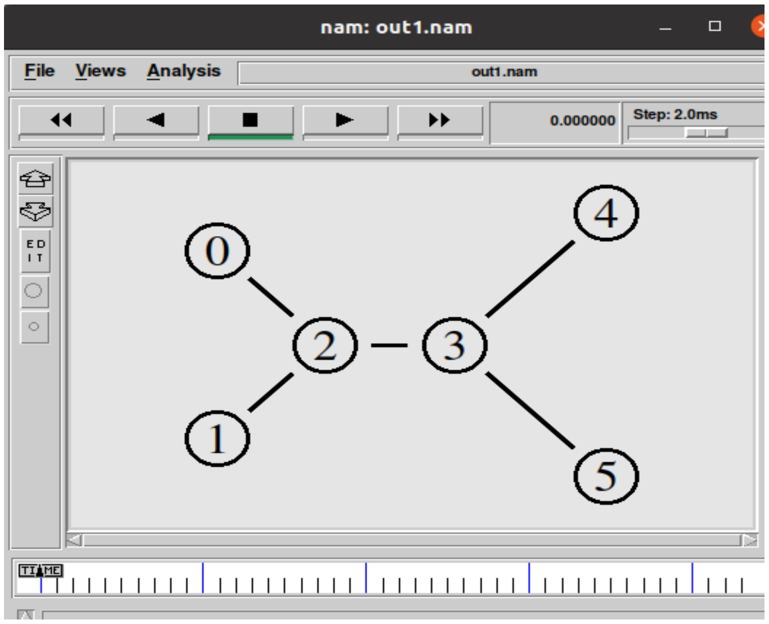
```
set sn [new Simulator]
$sn color 1 Blue
$sn color 2 Red
set nt [open out1.nam w]
$snnamtrace-all $nt

set tr [open out1.tr w]
```

```
$sn trace-all $tr

proc finish {} {
        global snnt tr
        $sn flush-trace
        close $nt
        close $tr
        exec nam out1.nam &
        exit 0
}
set n0 [$sn node]
set n1 [$sn node]
set n2 [$sn node]
set n3 [$sn node]
set n4 [$sn node]
set n5 [$sn node]

$sn duplex-link $n0 $n2 10Mb 10ms RED
$sn duplex-link $n1 $n2 10Mb 10ms RED
$sn duplex-link $n2 $n3 5Mb 6ms RED
$sn duplex-link $n3 $n4 10Mb 20ms RED
$sn duplex-link $n3 $n5 10Mb 20ms RED

$sn queue-limit $n3 $n4 10

$sn duplex-link-op $n0 $n2 orient right-down
$sn duplex-link-op $n1 $n2 orient right-up
$sn duplex-link-op $n2 $n3 orient right
$sn duplex-link-op $n3 $n4 orient right-up
$sn duplex-link-op $n3 $n5 orient right-down

$sn duplex-link-op $n2 $n3 queuePos 0.5
$sn duplex-link-op $n3 $n4 queuePos 0.5

set tcp [new Agent/TCP]
$tcp set class_ 1
$sn attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$sn attach-agent $n4 $sink
```

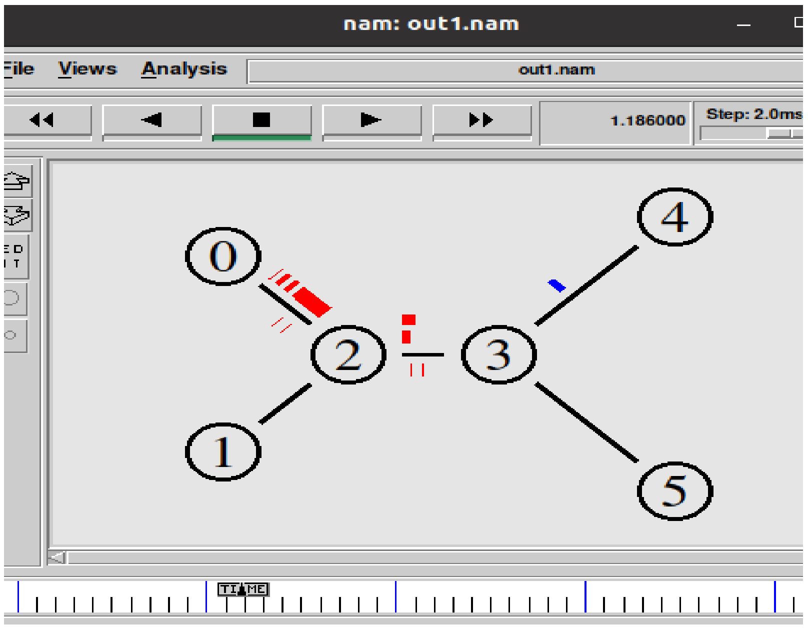
```
$sn connect $tcp $sink
$tcp set fid_ 2

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
$ftp set packet_size_ 1000
$ftp set packet_size_ 1mb


set udp [new Agent/UDP]
$sn attach-agent $n1 $udp

set null [new Agent/Null]
$sn attach-agent $n4 $null
$sn connect $udp $null
$udp set fid_ 1

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1500
$cbr set rate_ 0.05mb
$cbr set random_ false

$sn at 0.2 "$cbr start"
$sn at 0.8 "$ftp start"
$sn at 4.0 "$ftp stop"
$sn at 4.5 "$cbr stop"

$sn at 5.0 "finish"

$sn run
```

OUTPUT:-

{1} Network Topology:-



{2} Data Packet transfer in the network using TCP and UDP protocol:-

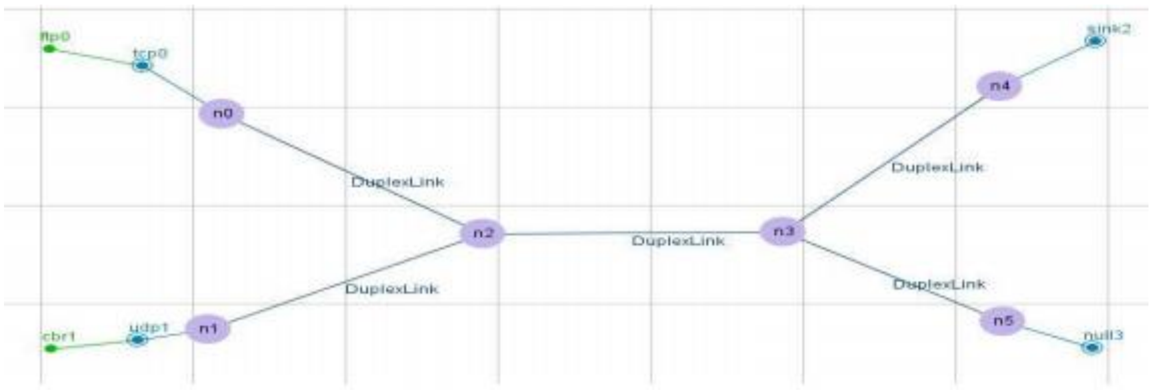


<u>CONCLUSION</u>: - In this practical the concepts of UDP protocol has been shown. The establishment of connection using UDP protocol in the network simulator has been shown.

PRACTICAL -7(C)

AIM:

To demonstrate various queuing mechanisms and make comparative analysis of various queuing techniques. (Using trace file) (Drop Tail, RED, SFQ, FQ).



ftp0:n0
Packet Size: 1000
Rate: 1
Interval: 150

cbr0:n1
Packet Size: 1500
Rate: 0.05
Interval: 150

THEORY:

SFQ

This queuing mechanism is based on fair queuing algorithm and proposed by John Nagle in 1987. Because it is impractical to have one queue for each conversation SFQ uses a hashing algorithm which divides the traffic over a limited number of queues. It is not so efficient than other queues mechanisms but it also requires less calculation while being almost perfectly fair. It is called "Stochastic" due to the reason that it does not actually assign a queue for every session; it has an algorithm which divides traffic over a restricted number of queues using a hashing algorithm._SFQ assigns a pretty large number of FIFO queues.

FQ (Fair Queuing)

It is a queuing mechanism that is used to allow multiple packets flow to comparatively share the link capacity. Routers have multiple queues for each output line for every user. When a line as available as idle routers scans the queues through round robin and takes first packet to next queue. FQ also ensure about the maximum throughput of the network. For more efficiency weighted queue mechanism is also used.

Drop Tail

It is a simple queue mechanism that is used by the routers that when packets should to be drop. In this mechanism each packet is treated identically and when queue filled to its maximum capacity the newly incoming packets are dropped until queue have sufficient space to accept incoming traffic.

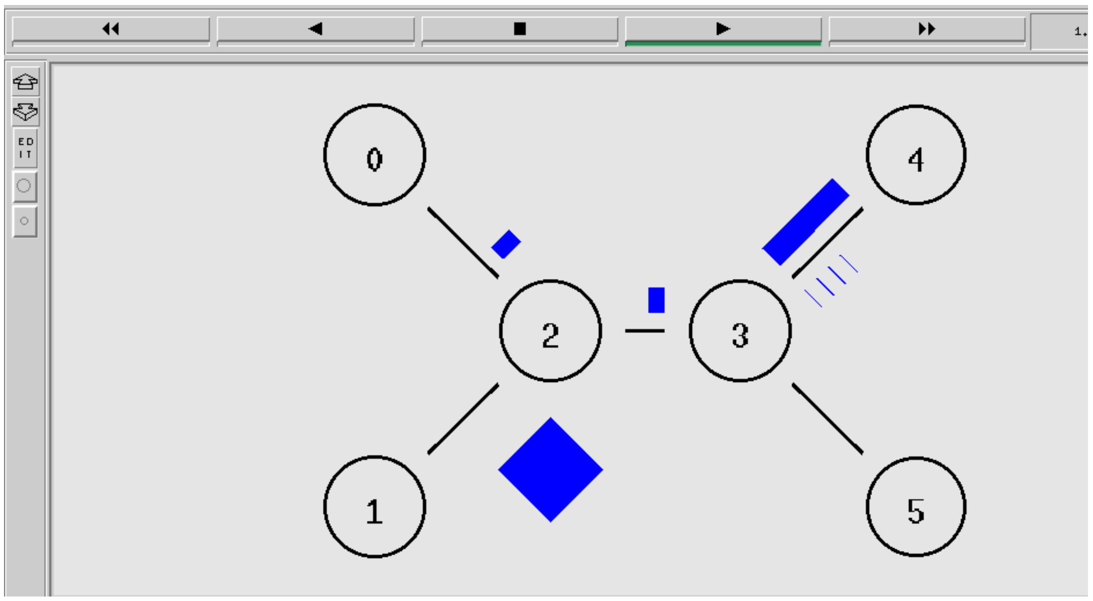
RED

Random Early Detection (RED) is a congestion avoidance queuing mechanism (as opposed to a congestion administration mechanism) that is potentially useful, particularly in high-speed transit networks. Sally Floyd and Van Jacobson projected it in various papers in the early 1990s.It is active queue management mechanism. It operates on the average queue size and drop packets on the basis of statistics information. If the buffer is empty all incoming packets are acknowledged. As the queue size increase the probability for discarding a packet also increase. When buffer is full probability becomes equal to 1 and all incoming packets are dropped.

CODE:

```
$ns duplex-link $n0 $n2 5Mb 10ms DropTail
$ns duplex-link $n1 $n2 5Mb 10ms DropTail
$ns duplex-link $n2 $n3 5Mb4ms DropTail
$ns duplex-link $n3 $n4 5Mb 10ms DropTail
$ns duplex-link $n3 $n5 5Mb 10ms DropTail

$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10
$ns queue-limit $n2 $n3 2
$ns queue-limit $n3 $n4 10
$ns queue-limit $n3 $n5 10
```

OUTPUT:



- **Changing the queue type – delay – queue limit:**

Queue Type	Delay (ms)	Queue limit	BANDWIDTH (Mb)
RED	10	10	5
Droptail	4	2	5
FQ	0	0	5
SFQ	0	0	5

CONCLUSION:

This practical helped me to gain a basic understanding of queuing techniques and by performing; it made me clear about the difference between all of them.

PRACTICAL -7(D)

AIM: To demonstrate the use of AWK script with NS2 trace file of scenario A. Find Out Throughput, Packet delivery ratio, Number Drop Packets for all Queues.

THEORY:

AWK Script
AWK is a high level programming language which is used to process text files, named after its three original author's name:
A: Alfred Aho
W: Peter Weinberger
K: Brian Kernighan
AWK Scripts are very good in processing the data from the log (trace files) which we get from NS2. If you want to process the trace file manually.
AWK PROGRAM STRUCTURE
AWK program structure contains mainly three parts;
1. Begin
2. Content
3. End

```
BEGIN {<initialization>}

<pattern1> {<actionSet1>}
<pattern2> {<actionSet2>}
   . . .


END {<final finalActionSet>}
```

BEGIN: Begin deals with what to be executed prior to text file processing, normally which is used to initialize variable values or constants.

CONTENT: Script which process the text file. In this part, AWK moves lines by lines (i.e., records by records) and executes the <actionSet> if the current line match with the pattern. The actions repeat until AWK reaches the end of the file.

END: This part explains what to be executed after the text file processing ie. what to print on the terminal or to show output in terminal.

EXECUTION
AWK has two types of execution;

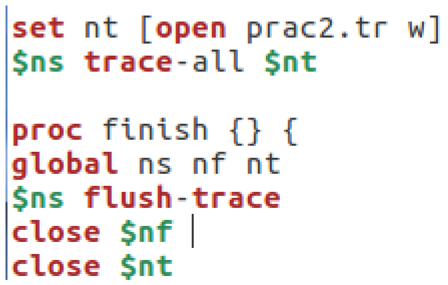
1) Plain Execution.
2) Match and Execute.

Plain Execution: Simply AWK statements.

Match and execute: The second type of execution is “Match and Execute”, when executes plain execution statements only if the current line (of the input text file) matches with a predefined pattern. The pattern can be either: 1. Logical Expression 2. Regular Expression.

TRACE FILE AND AWK SCRIPT
AWK Scripts are very good in processing the data column wise. For example the first column in the above trace file represents r, s which indicates receive, sent respectively. If we want to trace the entire r and s alone from this trace file we can represent it as $1
So
$1: Action (r, s, d.f)
$2 Time
$3 Time value
$4 node id
$5 node id value
$6 id of next hop
$7 next hop id value
And so on

SYNTAX for creating trace file:



```
set nt [open prac2.tr w]
$ns trace-all $nt

proc finish {} {
global ns nf nt
$ns flush-trace
close $nf
close $nt
```

<u>CODE:</u>

```
BEGIN {
        start_time=0
        finish_time=0
        get_start_time=0
        throughput=0
        latency=0
        file_size=0
}
{
        if ($1 == "r" && $4== 4)
        {
                print($1 , $4 , file_size)
                file_size+=$6
                if (get_start_time == 0)
                {
                        get_start_time =1
                        start_time=$2
                }
                finish_time =$2
        }
}
END{
        latency=finish_time-start_time
        throughput=(file_size*8)/latency
        printf ("%f\n", latency)
        printf ("%f\n", throughput)
}
```

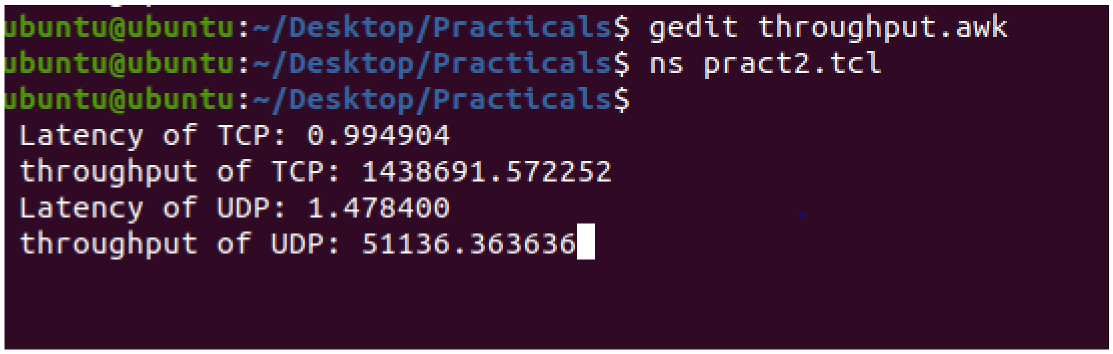
Packets Dropped, Received and it's ratio:

```
BEGIN{
        drop=0
        receive=0}
{if($1=="d"){
                drop++;}
        else($1=="r")
        {receive++;}}
END{
        printf("\npacket dropped= %d",drop)
        #printf("\npacket received= %d",receive)
        printf("\npacket ratio= %f",receive/(drop+receive))
}
```

Link AWK script with tr file in tcl script:

```
exec awk –f throughput.awk prac2.tr &
```

OUTPUT:



```
ubuntu@ubuntu:~/Desktop/Practicals$ gedit throughput.awk
ubuntu@ubuntu:~/Desktop/Practicals$ ns pract2.tcl
ubuntu@ubuntu:~/Desktop/Practicals$
 Latency of TCP: 0.994904
 throughput of TCP: 1438691.572252
 Latency of UDP: 1.478400
 throughput of UDP: 51136.363636
```

CONCLUSION: By performing this practical I have understood use of AWK file.