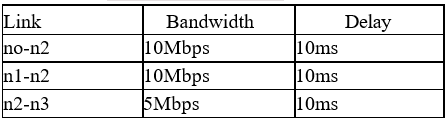
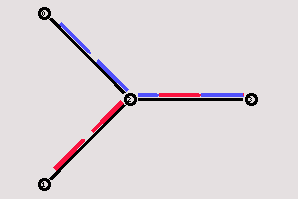
**PRACTICAL – 7**

**Part - A**

**AIM:** Design simple tcl script for Wired topology of 4 nodes in NS-2 and analyze various tcl parameters like network nodes, links, queues and topology. Queue Size :- 5 ,Duplex Link, Queue Type Droptail.





**ftp0:**

**Packet Size: 1000**

**Rate: 1**

**Interval: 150**

**THEORY:**

**NS2**

* **NS2 is a well known organization test system device.**
* **Ns2 is upheld by the working frameworks falling under linux appropriations, and macintosh working framework.**
* **It is a discrete occasion test system whose parent programming appropriation is NS(Network Simulator) . NS is accessible in 3 renditions dependent on varieties of highlights gave by them.**
* **NS is generally utilized for exploration and showing reason on themes, for example, geographies ,convention recreations , directing and multicasting conventions.**

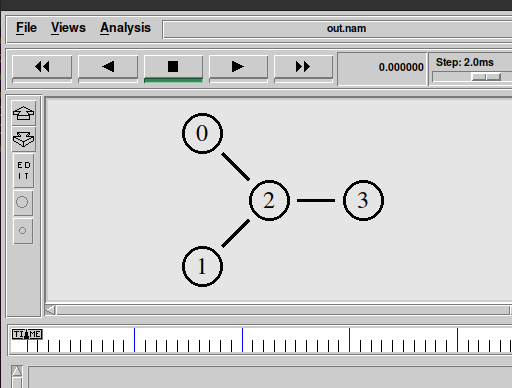
**Duplex Link - Duplex link is a link type to interface numerous hubs with one another in the organization test system.**

**Queue - Queue type relate to the type of connection between gadgets dependent on the transmission capacity and bundle size that various queues uphold.**

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

**Use of NS2 - NS2 is a serious rendition of Network Simulator(NS). It is generally famous for its administrations to gather and picture virtual organizations for recreation. It is better than cisco bundle tracer as every gadget in the association goes about as a hub in NS2 while a particular component qualities should be characterized in cisco(i.e if a hub is switch, switch ,and so forth.). There are numerous other comparable highlights which make NS2 more mainstream to utilize.**

**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 nam out.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"**

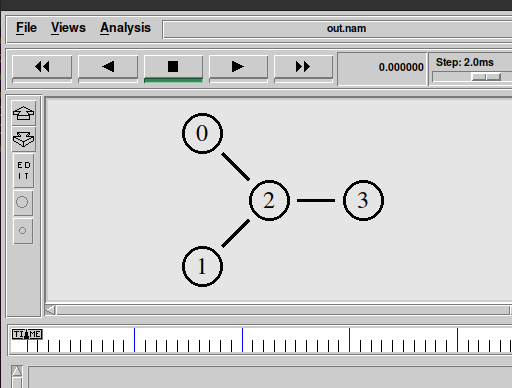
**$ns at 4.0 "$ftp stop"**

**$ns at 5.0 "finish"**

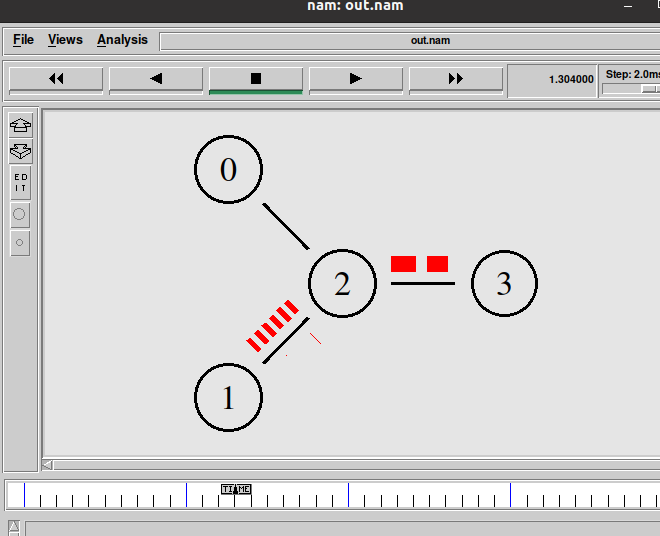
**$ns run**

**OUTPUT :**

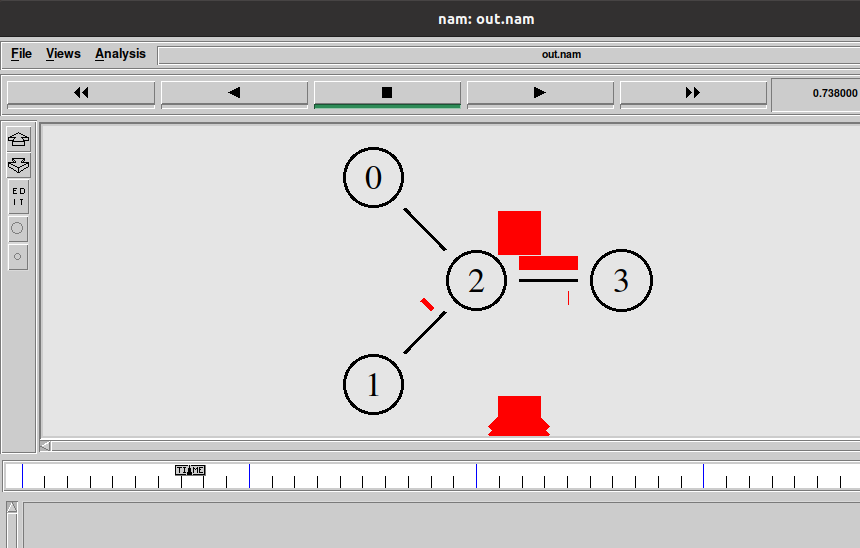
**{1} Network Topology:-**



**{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) :-**

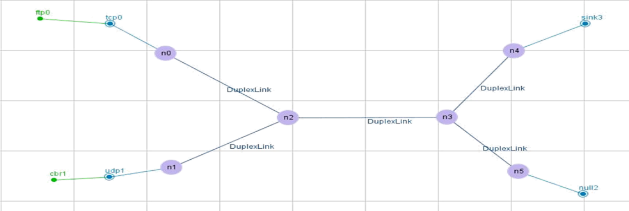


**CONCLUSION:**

In this useful essentials of network simulator 2 are secured. Likewise we have figured out how to set up tcp association utilizing ftp convention utilizing ns2 programming. Additionally prologue to various networking terms in network simulator, for example, line , hubs, interface ,and so on have been made.

**Part - B**

**AIM:** Design simple tcl script for Wired topology of 6 nodes in NS-2 and analyze various tcl parameters like network nodes, links, queues and 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**

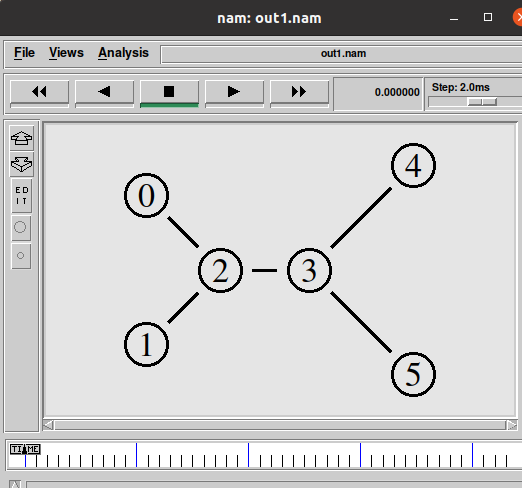
**Interval: 150**

**THEORY:**

**Difference Between TCP and UDP connection:-**

* **TCP is an association arranged convention. While, UDP is a connectionless convention.**
* **As TCP gives blunder checking support and furthermore ensures conveyance of information to the objective switch this make it more dependable when contrasted with UDP. UDP doesn't give blunder control.**
* **TCP likewise underpins sequencing which implies that information bundles follow just a single way while getting sent. While sequencing isn't trailed by UDP.**
* **TCP is moderate and along these lines isn't a lot of proficient to the extent transmission speed of information and its exhibition are concerned. Then again UDP is a lot quicker and is in this way more productive.**

**TOPOLOGY:**



**PROGRAM CODE :**

**set sn [new Simulator]**

**$sn color 1 Blue**

**$sn color 2 Red**

**set nt [open out1.nam w]**

**$sn namtrace-all $nt**

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

**$sn trace-all $tr**

**proc finish {} {**

**global sn nt 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"**

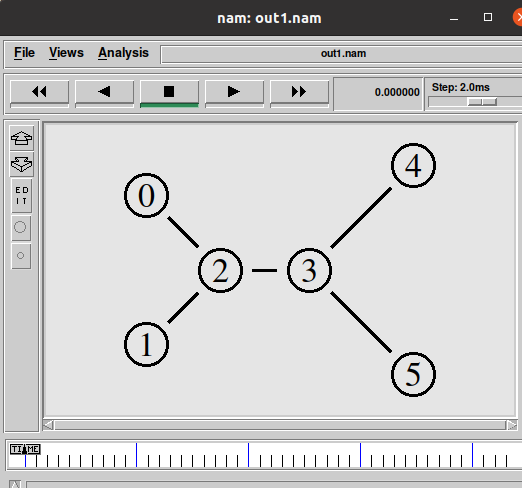
**$sn at 4.5 "$cbr stop"**

**$sn at 5.0 "finish"**

**$sn run**

**OUTPUT :**

**{1} Network Topology:-**



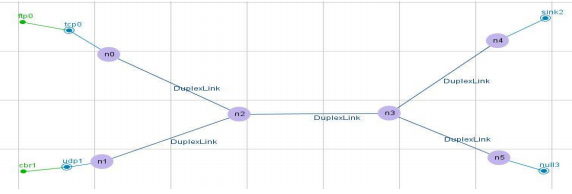
**{2} Data Packet transfer in the network using TCP and UDP protocol:-**

**CONCLUSION:**

In this functional the ideas of UDP convention has been appeared. The foundation of association utilizing UDP convention in the organization test system has been appeared .

**Part - 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**

**SFQ**

**This lining system depends on reasonable lining calculation and proposed by John Nagle in 1987. Since it is illogical to have one line for every discussion SFQ utilizes a hashing calculation which isolates the traffic over a set number of lines. It isn't so effective than different lines instruments however it additionally requires less computation while being completely reasonable. It is classified "Stochastic" because of the explanation that it doesn't really relegate a line for each meeting; it has a calculation which partitions traffic over a confined number of lines utilizing a hashing calculation. SFQ allocates an entirely enormous number of FIFO lines**.

**PROGRAM CODE :**

**For SFQ:**

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open prac2.nam w]

$ns namtrace-all $nf

set nr [open prac2.tr w]

$ns trace-all $nr

#Define a 'finish' procedure

proc finish {} {

global ns nf nr

$ns flush-trace

#Close the NAM trace file

close $nf

close $nr

#Execute NAM on the trace file

exec nam prac2.nam &

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]

$n0 shape circle

$n1 shape circle

$n2 shape circle

$n3 shape circle

$n4 shape circle

$n5 shape circle

$n0 label "1 Node"

$n1 label "2 Node"

$n2 label "3 Node"

$n3 label "4 Node"

$n4 label "5 Node"

$n5 label "6 Node"

$ns duplex-link $n0 $n2 10Mb 10ms SFQ

$ns duplex-link $n1 $n2 10Mb 10ms SFQ

$ns duplex-link $n2 $n3 5Mb 5ms SFQ

$ns duplex-link $n3 $n4 10Mb 10ms SFQ

$ns duplex-link $n3 $n5 10Mb 10ms SFQ

$ns queue-limit $n0 $n2 10

$ns queue-limit $n1 $n2 10

$ns queue-limit $n2 $n3 5

$ns queue-limit $n3 $n4 10

$ns queue-limit $n3 $n5 10

$ns duplex-link-op $n0 $n2 orient down-right

$ns duplex-link-op $n1 $n2 orient up-right

$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n3 $n4 orient up-right

$ns duplex-link-op $n3 $n5 orient down-right

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n4 $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

$ftp set interval\_ 150

set udp [new Agent/UDP]

$udp set class\_ 2

$ns attach-agent $n1 $udp

set sink1 [new Agent/Null]

$ns attach-agent $n5 $sink1

$ns connect $udp $sink1

$udp set fid\_ 2

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 interval\_ 150

$ns at 1.0 "$ftp start"

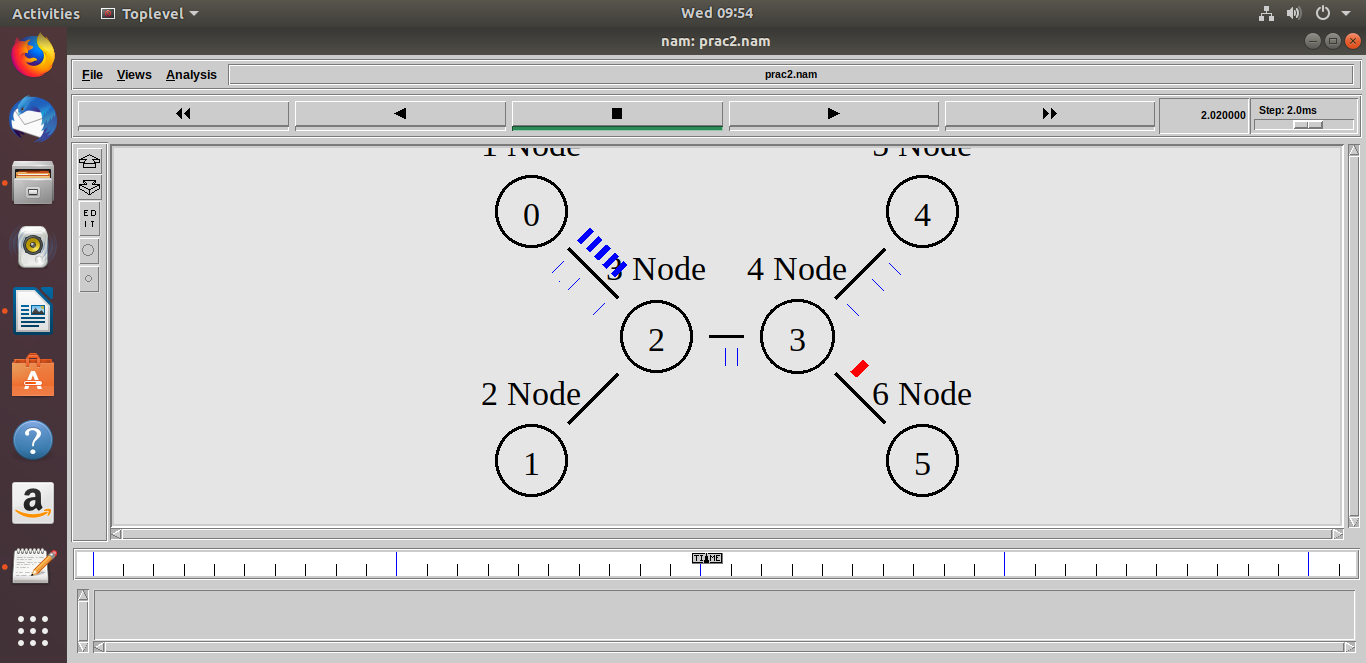
$ns at 4.0 "$ftp stop"

$ns at 2.0 "$cbr start"

$ns at 4.5 "$cbr stop"

$nsat5.0"finish"  
$ns r

**OUTPUT :**



**FQ (Fair Queuing)**

It is a lining component that is utilized to permit numerous parcels stream to similarly share the connection limit. Switches have different lines for each yield line for each client. At the point when a line as accessible as inactive switches filters the lines through cooperative effort and takes first parcel to next line. FQ likewise guarantee about the most extreme throughput of the organization. For more effectiveness weighted line system is additionally utilized.

**PROGRAM CODE :**

**For FQ:**

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open prac3.nam w]

$ns namtrace-all $nf

set nr [open prac3.tr w]

$ns trace-all $nr

#Define a 'finish' procedure

proc finish {} {

global ns nf nr

$ns flush-trace

#Close the NAM trace file

close $nf

close $nr

#Execute NAM on the trace file

exec nam prac3.nam &

exec awk -f through.awk prac3.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]

$n0 shape circle

$n1 shape circle

$n2 shape circle

$n3 shape circle

$n4 shape circle

$n5 shape circle

$n0 label "1 Node"

$n1 label "2 Node"

$n2 label "3 Node"

$n3 label "4 Node"

$n4 label "5 Node"

$n5 label "6 Node"

$ns duplex-link $n0 $n2 10Mb 10ms FQ

$ns duplex-link $n1 $n2 10Mb 10ms FQ

$ns duplex-link $n2 $n3 5Mb 5ms FQ

$ns duplex-link $n3 $n4 10Mb 10ms FQ

$ns duplex-link $n3 $n5 10Mb 10ms FQ

$ns queue-limit $n0 $n2 10

$ns queue-limit $n1 $n2 10

$ns queue-limit $n2 $n3 5

$ns queue-limit $n3 $n4 10

$ns queue-limit $n3 $n5 10

$ns duplex-link-op $n0 $n2 orient down-right

$ns duplex-link-op $n1 $n2 orient up-right

$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n3 $n4 orient up-right

$ns duplex-link-op $n3 $n5 orient down-right

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n4 $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

$ftp set interval\_ 150

set udp [new Agent/UDP]

$udp set class\_ 2

$ns attach-agent $n1 $udp

set sink1 [new Agent/Null]

$ns attach-agent $n5 $sink1

$ns connect $udp $sink1

$udp set fid\_ 2

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 interval\_ 150

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

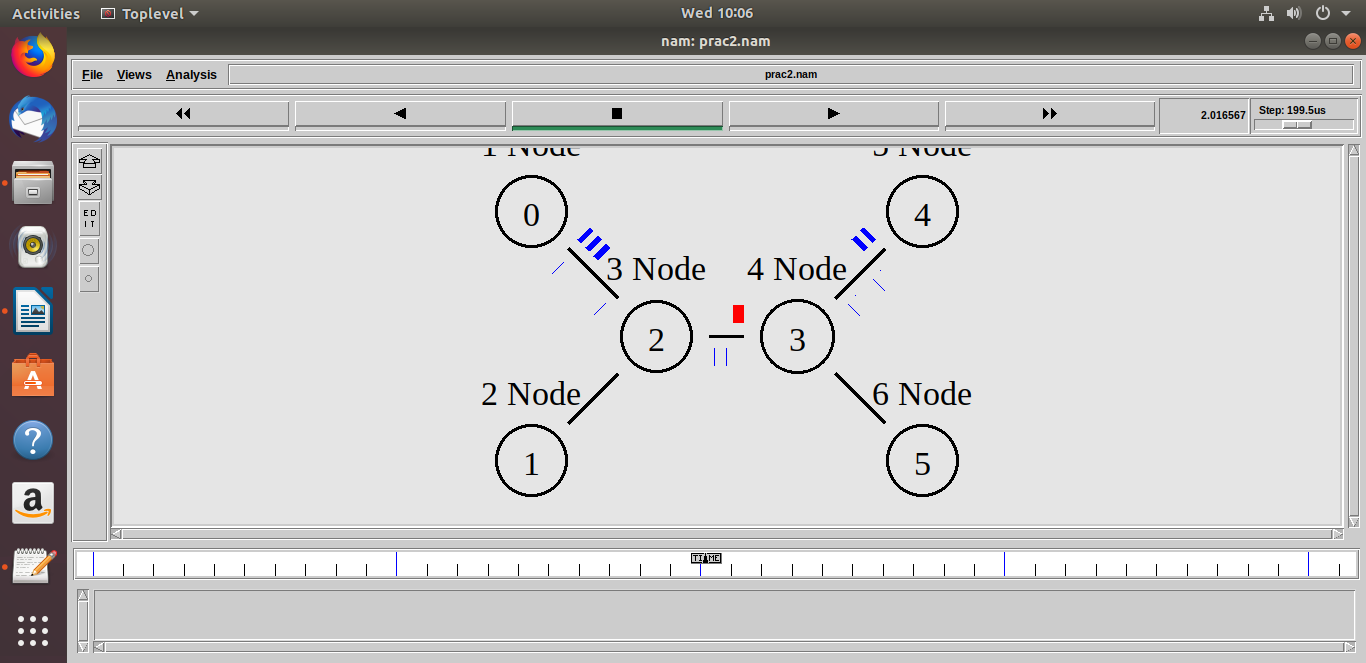
$ns at 2.0 "$cbr start"

$ns at 4.5 "$cbr stop"

$ns at 5.0 "finish"

$ns run

**OUTPUT :**



**FQ (Fair Queuing)**

It is a basic line instrument that is utilized by the switches that when parcels ought to be drop. In this instrument every bundle is dealt with indistinguishably and when line completely filled the recently approaching parcels are dropped until line have adequate space to acknowledge approaching traffic

**PROGRAM CODE :**

**For Droptall:**

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open prac3.nam w]

$ns namtrace-all $nf

set nr [open prac3.tr w]

$ns trace-all $nr

#Define a 'finish' procedure

proc finish {} {

global ns nf nr

$ns flush-trace

#Close the NAM trace file

close $nf

close $nr

#Execute NAM on the trace file

exec nam prac3.nam &

exec awk -f through.awk prac3.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]

$n0 shape circle

$n1 shape circle

$n2 shape circle

$n3 shape circle

$n4 shape circle

$n5 shape circle

$n0 label "1 Node"

$n1 label "2 Node"

$n2 label "3 Node"

$n3 label "4 Node"

$n4 label "5 Node"

$n5 label "6 Node"

$ns duplex-link $n0 $n2 10Mb 10ms DropTail

$ns duplex-link $n1 $n2 10Mb 10ms DropTail

$ns duplex-link $n2 $n3 5Mb 5ms DropTail

$ns duplex-link $n3 $n4 10Mb 10ms DropTail

$ns duplex-link $n3 $n5 10Mb 10ms DropTail

$ns queue-limit $n0 $n2 10

$ns queue-limit $n1 $n2 10

$ns queue-limit $n2 $n3 5

$ns queue-limit $n3 $n4 10

$ns queue-limit $n3 $n5 10

$ns duplex-link-op $n0 $n2 orient down-right

$ns duplex-link-op $n1 $n2 orient up-right

$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n3 $n4 orient up-right

$ns duplex-link-op $n3 $n5 orient down-right

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n4 $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

$ftp set interval\_ 150

set udp [new Agent/UDP]

$udp set class\_ 2

$ns attach-agent $n1 $udp

set sink1 [new Agent/Null]

$ns attach-agent $n5 $sink1

$ns connect $udp $sink1

$udp set fid\_ 2

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 interval\_ 150

$ns at 1.0 "$ftp start"

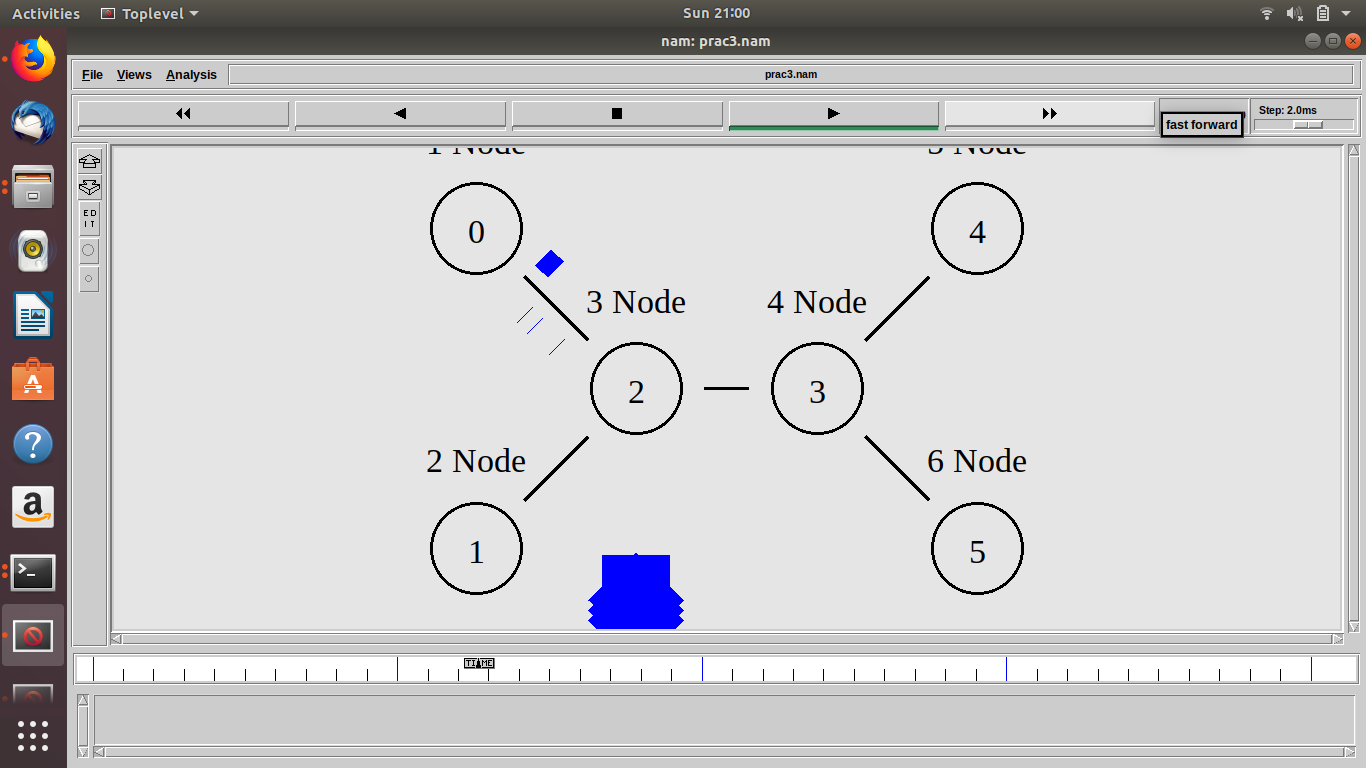
$ns at 4.0 "$ftp stop"

$ns at 2.0 "$cbr start"

$ns at 4.5 "$cbr stop"

$ns at 5.0 "finish"

**OUTPUT :**



**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.

**PROGRAM CODE :**

**For RED**

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open prac3.nam w]

$ns namtrace-all $nf

set nr [open prac3.tr w]

$ns trace-all $nr

#Define a 'finish' procedure

proc finish {} {

global ns nf nr

$ns flush-trace

#Close the NAM trace file

close $nf

close $nr

#Execute NAM on the trace file

exec nam prac3.nam &

exec awk -f through.awk prac3.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]

$n0 shape circle

$n1 shape circle

$n2 shape circle

$n3 shape circle

$n4 shape circle

$n5 shape circle

$n0 label "1 Node"

$n1 label "2 Node"

$n2 label "3 Node"

$n3 label "4 Node"

$n4 label "5 Node"

$n5 label "6 Node"

$ns duplex-link $n0 $n2 10Mb 10ms RED

$ns duplex-link $n1 $n2 10Mb 10ms RED

$ns duplex-link $n2 $n3 5Mb 5ms RED

$ns duplex-link $n3 $n4 10Mb 10ms RED

$ns duplex-link $n3 $n5 10Mb 10ms RED

$ns queue-limit $n0 $n2 10

$ns queue-limit $n1 $n2 10

$ns queue-limit $n2 $n3 5

$ns queue-limit $n3 $n4 10

$ns queue-limit $n3 $n5 10

$ns duplex-link-op $n0 $n2 orient down-right

$ns duplex-link-op $n1 $n2 orient up-right

$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n3 $n4 orient up-right

$ns duplex-link-op $n3 $n5 orient down-right

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n4 $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

$ftp set interval\_ 150

set udp [new Agent/UDP]

$udp set class\_ 2

$ns attach-agent $n1 $udp

set sink1 [new Agent/Null]

$ns attach-agent $n5 $sink1

$ns connect $udp $sink1

$udp set fid\_ 2

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 interval\_ 150

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

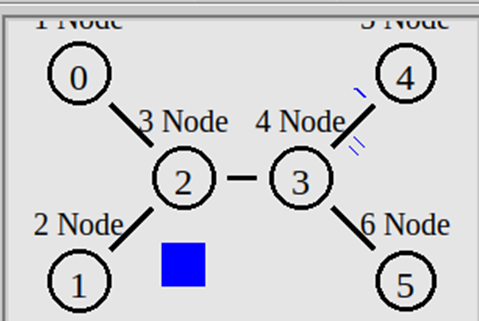
$ns at 2.0 "$cbr start"

$ns at 4.5 "$cbr stop"

$ns at 5.0 "finish"

$ns run

**OUTPUT :**



**Conclusion:**

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

**Part - 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

which make NS2 more mainstream to utilize.

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

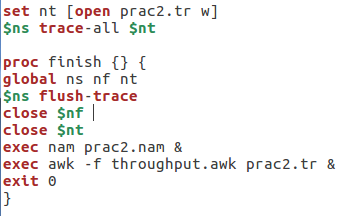
$5 node id value

$6 id of next hop

$7 next hop id value

And so on

**SYNTAX for creating trace file:**

****

**PROGRAM CODE :**

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)

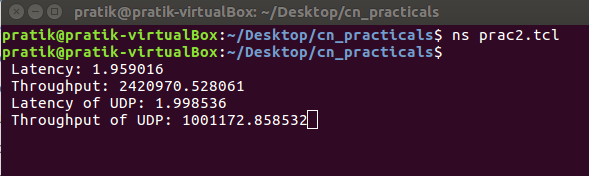
#printf("\npacket received= %d",receive)

printf("\npacket ratio= %f",receive/(drop+receive)) }

**Link AWK script with tr file in tcl script:**

exec awk –f throughput.awk prac2.tr &

**OUTPUT :**



**CONCLUSION:**

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