PRACTICAL-8(A)

AIM:

Design simple tcl script for transferring FTP & CBR traffic in Wireless topology of 6 nodes using NS-2

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:

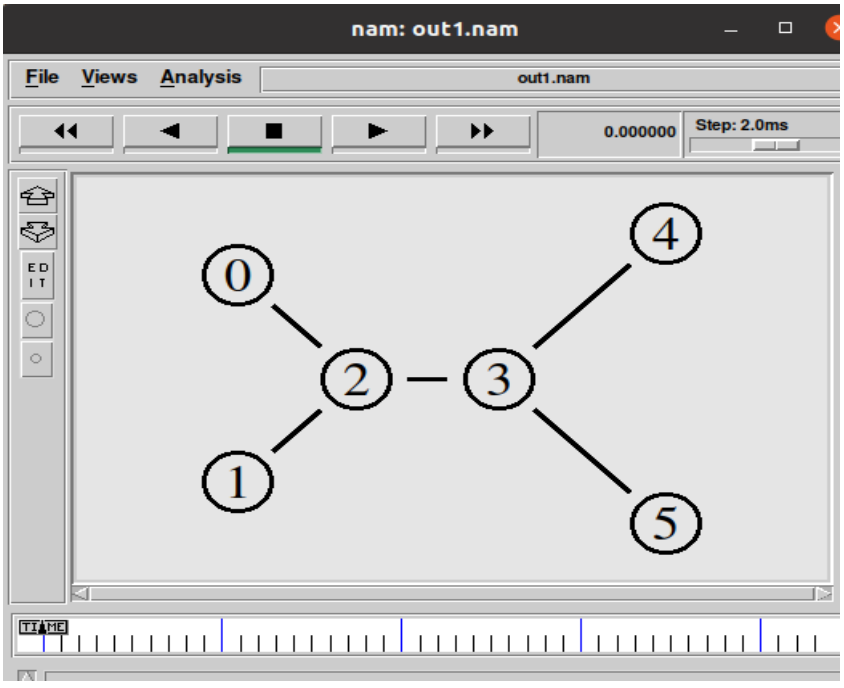
- Tcl is a general purpose multi-paradigm system programming language.

- It is a scripting language that aims at providing the ability for applications to communicate with each other.

- On the other hand, Tk is a cross platform widget toolkit used for building GUI in many languages.

- This tutorial covers various topics ranging from the basics of the Tcl/Tk to its scope in various applications.

TOPOLOGY:



PROGRAM CODE:

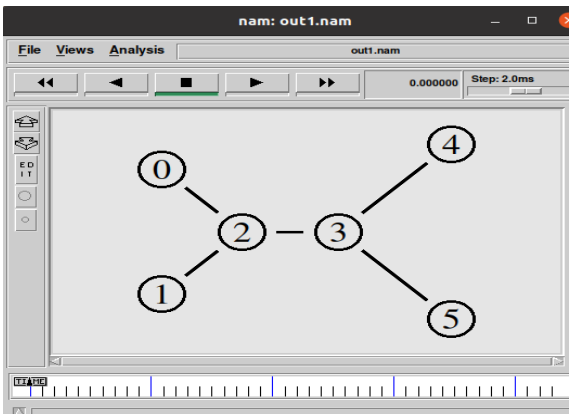
```
set sn [new Simulator]
$sn color 1 Blue
$sn color 2 Red
set nt [open out1.nam w]
$sn namtrace-all $nt

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:

Network Topology:



PRACTICAL-8(B)

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 names:

  A: Alfred

  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:

- 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.

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)     }


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))  }
```

CONCLUSION:

- In this practical, we learned about TCL script
- We also learnt about AWK script.