TERM 1 CN

```
set val(stop) 10.0;
set ns [new Simulator]
$ns color 1 blue
$ns color 2 red
set tracefile [open p1.tr w]
$ns trace-all $tracefile
set namfile [open p1.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]
#assign labels to nodes
$n0 label "TCP-source"
$n1 label "UDP-source"
$n2 label "UDP-destination"
$n3 label "TCP-destination"
$n4 label "Router"
#assign shapes to nodes
$n0 shape square
$n1 shape "square"
$n2 shape "hexagon"
$n3 shape "hexagon"
$n4 shape "circle"
#assign color to nodes
$n0 color green
$n1 color green
$n2 color red
$n3 color red
$n4 color black
#commands to stablish links between nodes
$ns duplex-link $n0 $n4 100.0Mb 40ms DropTail
$ns queue-limit $n0 $n4 5
$ns duplex-link $n4 $n3 100.0Mb 40ms DropTail
$ns queue-limit $n4 $n3 5
$ns duplex-link $n1 $n4 100.0Mb 40ms DropTail
$ns queue-limit $n1 $n4 5
$ns duplex-link $n4 $n2 100.0Mb 40ms DropTail
$ns queue-limit $n4 $n2 5
$ns duplex-link-op $n4 $n2 queuePos 0.5
$ns duplex-link-op $n4 $n2 queuePos 0.5
#assigning orientation
```

```
$ns duplex-link-op $n4 $n0 orient left-down
$ns duplex-link-op $n1 $n4 orient left-up
$ns duplex-link-op $n3 $n4 orient left-down
$ns duplex-link-op $n2 $n4 orient right-down
#attaching agent
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp0 $sink3
$tcp0 set packetSize_ 1000
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n2 $null2
$ns connect $udp1 $null2
$udp1 set packetSize_ 1000
$tcp0 set fid_ 1
$udp1 set fid_ 2
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $tcp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 3.0Mb
$cbr0 set random_ null
$ns at 0.01 "$cbr0 start"
$ns at 0.99 "$cbr0 stop"
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 3.0Mb
$cbr1 set random_ null
$ns at 0.1 "$cbr1 start"
$ns at 9.0 "$cbr1 stop"
proc finish {} {
        global ns tracefile namfile
        $ns flush-trace
        close $tracefile
        close $namfile
        exec nam p1.nam
        exit 0
}
$ns at $val(stop) "finish"
$ns run
```

TERM 2 CN

```
set val(stop) 25.0; #time of simulator end

#create ns object
set ns  [new Simulator]

#open the ns trace file
set tracefile [open p2.tr w];
$ns trace-all $tracefile;

#open the nam file
set namfile [open p2.nam w];
$ns namtrace-all $namfile;

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

#create labels for nodes

$n0 label "UDP SOURCE";
$n1 label "ONE"
$n2 label "TWO"
$n3 label "THREE"
$n4 label "FOUR"
$n5 label "FIVE"
$n6 label "UDP DESTINATION"

#give shapes to nodes

$n0 shape hexagon;
$n1 shape circle;
$n2 shape square;
$n3 shape square;
```

```
$n4 shape square;
$n5 shape square;
$n6 shape circle;

#give colors to nodes
$n0 color red;
$n1 color blue;
$n2 color blue;
$n3 color blue;
$n4 color blue;
$n5 color blue;
$n6 color black;

set lan [ $ns newLan "$n0 $n1 $n2 $n3 $n4 $n5 $n6" 1.0Mb 40ms LL
Queue|DropTail Mac|802_3 Channel ];

#setup 1 UDP connection
set udp1 [new Agent/UDP];
$ns attach-agent $n0 $udp1;
$udp1 set packetSize_ 1000;

set null2 [new Agent/Null];
$ns attach-agent $n6 $null2;

#connect source to destination
$ns connect $udp1 $null2;

#setup  cbr application over udp connection

set cbr1 [new Application/Traffic/CBR];
$cbr1 attach-agent $udp1;

#set interval
$cbr1 set interval_ 0.1;

#assign flow id
$ns color 1 red;
```

```
$udp1 set fid_ 1;

$ns at 0.1 "$cbr1 start";
$ns at 24.9 "$cbr1 stop";

#define a procedure
proc finish {} {
        global ns tracefile namfile;
        $ns  flush-trace;
        close $tracefile;
        close $namfile;
        exec nam p2.nam &;
        exit 0;
}
$ns at $val(stop) "finish";
$ns run
```

TERM 3 CN

```
set val(stop) 50
set ns [new Simulator]
#open the ns trace file
set tracefile [open p3.tr w]
$ns trace-all $tracefile
#open the ns nam file
set namfile [open p3.nam w]
$ns namtrace-all $namfile
set n0 [$ns node]
set n1 [$ns node]
#assign labels to nodes
$n0 label "SERVER"
$n1 label "CLIENT"
#assign shapes to nodes
$n0 shape square
$n1 shape square
#assign color to nodes
$n0 color red
$n1 color blue
#attaching agent
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500
#attaching application
set ftp0 [new Application/FTP]
$ ftp0 attach-agent $tcp0
$ns at 0.01 "$ftp0 start"
$ns at 20.2 "$ftp0 stop"
#commands to stablish links between nodes
$ns duplex-link $n0 $n1 100.0Mb 40ms DropTail
$ns queue-limit $n0 $n1 5
#assigning orientation
$ns duplex-link-op $n0 $n1 orient right
$ns color 1 red
$tcp0 set fid_ 1
$ftp0 set Type_ FTP
#Define a finish procedure
proc finish {} {
        global ns tracefile namfile
        $ns flush-trace
        close $tracefile
        close $namfile
        exec nam p1.nam &
        exec awk -f first.awk   p3.tr &
        exec awk -f graph.awk p3.tr > p3.dat &
        exec xgraph p3.dat -geometry 800x400 -t “Bytes_Recieved_at_client” -x
        “Time_in_sec” -y “Bytes_in_bps” &
        exit 0
}
$ns at $val(stop) "finish";
#Run the simulation
$ns run;
```

First.awk

```
BEGIN {
                count = 0
                time = 0
                total_bytes_sent = 0
                total_bytes_recieved = 0
}
{
                if($1==”r” && $4==1 && $5==”tcp”)
                total_bytes_recieved += $6
                if($1==”+” && $3==0 && $5==”tcp”)
                total_bytes_sent += $6
}
END {
                system(“Clear”)\
                printf(“\nTransmission time required to transfer the file is %1f ”,$2);
                printf(“\nActual data sent from the server  is %1f  Mbps”,
                total_bytes_sent/1000000);
                printf(“\nData received by the client  is %1f  Mbps”,
                total_bytes_recieved /1000000);
}
```

graph.awk

```
BEGIN {
                count = 0
                time = 0
}
{
                if($1==”r” && $4==1 && $5==”tcp”)
                {
                        count += $6
                        time += $2
                        printf(“\n %1f  \t %1f”,time(count)/1000000)
                }
}
```

TERM 4 CN

```
set ns [new Simulator]

set tracefile [open t4.tr w]
$ns trace-all $tracefile;

$ns color 1 red;
$ns color 2 blue;

set namfile [open t4.nam w]
$ns namtrace-all $namfile;

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

$n0 shape "circle";
$n1 shape "square";
$n2 shape "circle";
$n3 shape "circle";
$n4 shape "circle";
$n5 shape "circle";
$n6 shape "circle";
$n7 shape "circle";

$n0 label "TCP-SOURCE";
$n1 label "UDP-SOURCE";
$n2 label "Router";
$n3 label "Router";
$n6 label "TCP-Sink";
$n7 label "UDP-Null";

set tcp0 [new Agent/TCP]
```

```
$ns attach-agent $n0 $tcp0

set sink6 [new Agent/TCPSink]
$ns attach-agent $n6 $sink6

$ns connect $tcp0 $sink6

$ns duplex-link $n0 $n2 1.0Mb 10ms DropTail
$ns duplex-link $n1 $n2 1.0Mb 10ms DropTail
$ns duplex-link $n2 $n3 2.0Mb 10ms DropTail
$ns duplex-link $n3 $n4 1.0Mb 10ms DropTail
$ns duplex-link $n3 $n5 1.0Mb 10ms DropTail
$ns duplex-link $n4 $n6 1.0Mb 10ms DropTail
$ns duplex-link $n5 $n7 1.0Mb 10ms DropTail

$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 $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
$ns duplex-link-op $n4 $n6 orient right
$ns duplex-link-op $n5 $n7 orient right

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

set null7 [new Agent/Null]
$ns attach-agent $n7 $null7

$ns connect $udp1 $null7

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp1
$cbr0 set rate_ 0.5Mb
```

```
$tcp0 set packetSize_ 1500
$udp1 set packetSize_ 1500

$tcp0 set fid_ 1
$udp1 set fid_ 2

$ns at 0.1 "$cbr0 start"
$ns at 10.0 "$cbr0 stop"

$ns at 0.1 "$ftp0 start"
$ns at 10.0 "$ftp0 stop"

proc finish {} {
        global ns tracefile namfile
        $ns flush-trace
        close $tracefile
        close $namfile
        exec nam t4.nam &
        exec awk -f term4.awk t4.tr &
        exec xgraph data.dat -geometry 800*400 "Throughput" -y "Bandwidth" -bg white &
        exit 0
}
$ns at 10.1 "finish";
$ns run;
```

Awk file:

```
BEGIN {
     count=0;
     time=0;
     dp=0;
}
{
     if($1=="r"&&$3==2&&$4==3)
```

```
    {
     count+= $6;
     time=$2;
    }
    if( $1=="d")
    {
     cp+=1;
    }
}
END{
  printf("\n Total amount of data sent %d",count);
  printf("\n Total time of data trasmission %if",time);
  printf("\n Throughput:%if Mbps",(count/time)*(8/1000000));
  printf("\n Packet Dropped=%d\n",cp);
```

TERM 5 CN

```
set val(stop) 5.0;
set ns [ new Simulator ]

$ns rtproto DV

set tracefd [ open term5.tr w ]
$ns trace-all $tracefd

set nf [ open term5.nam w ]
$ns namtrace-all $nf

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]

$n0 color red
$n3 color green

$n0 shape hexagon
$n3 shape hexagon

$ns duplex-link $n0 $n1 1.0Mb 10ms DropTail;
$ns duplex-link $n1 $n2 1.0Mb 10ms DropTail;
$ns duplex-link $n2 $n3 1.0Mb 10ms DropTail;
$ns duplex-link $n0 $n6 1.0Mb 10ms DropTail;
$ns duplex-link $n6 $n5 1.0Mb 10ms DropTail;
$ns duplex-link $n5 $n4 1.0Mb 10ms DropTail;
$ns duplex-link $n4 $n3 1.0Mb 10ms DropTail;

$ns duplex-link-op $n0 $n1 orient down-right;
$ns duplex-link-op $n1 $n2 orient down;
$ns duplex-link-op $n2 $n3 orient down-left;
$ns duplex-link-op $n3 $n4 orient up-left
```

```
$ns duplex-link-op $n4 $n5 orient up
$ns duplex-link-op $n5 $n6 orient up
$ns duplex-link-op $n0 $n6 orient down-left;

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

set null3 [new Agent/Null]
$ns attach-agent $n3 $null3

$ns connect $udp0 $null3

set cbr0 [new application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

$ns at 0.1 "$cbr0 start"
$ns at 10.0 "$cbr0 stop"

$ns rtmodel-at 1.0 down $n1 $n2
$ns rtmodel-at 2.0 up $n1 $n2

$udp0 set fid_ 1
$ns color 1 blue

proc finish {} {
        global ns nf tracefd
        $ns flush-trace
        close $nf
        close $tracefd
        exec nam term5.nam &
        exit 0;
}
$ns at $val(stop) "finish"
$ns run
```

TERM 6 CN

```
#simulation parameters setup
set val(chan) Channel/WirelessChannel;
set val(prop) Propagation/TwoRayGround;
set val(netif) Phy/WirelessPhy;
set val(mac) Mac/802_11;
set val(ifq) CMUPriQueue;
set val(ll) LL;
set val(ant) Antenna/OmniAntenna;
set val(ifqlen) 50;
set val(nn) 6;
set val(rp) DSR;
set val(X) 700;
set val(Y) 700;
set val(stop) 60.0;

#create a ns simulator
set ns [new Simulator]

#setup topography object
set topo [new Topography]
$topo load_flatgrid $val(X) $val(Y)
create-god $val(nn)

#open the NS trace file
set tracefile [open lab6.tr w]
$ns trace-all $tracefile

#open the NS nam file
set namfile [open lab6.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(X) $val(Y)

#create   wireless   channel
set chan [new $val(chan)];
```

```
$ns node-config -adhocRouting $val(rp) \
                -llType         $val(ll) \
                -macType    $val(mac) \
                -ifqType    $val(ifq) \
                -ifqLen         $val(ifqlen) \
                -antType    $val(ant) \
                -propType    $val(prop) \
                -phyType    $val(netif) \
                -channel    $chan\
                -topoInstance $topo\
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#create node with initial positions
set n0 [$ns node]
$n0 set X_ 150
$n0 set Y_ 300
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20

set n1 [$ns node]
$n1 set X_ 300
$n1 set Y_ 500
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20

set n2 [$ns node]
$n2 set X_ 500
$n2 set Y_ 500
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20

set n3 [$ns node]
$n3 set X_ 300
$n3 set Y_ 100
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
```

```
set n4 [$ns node]
$n4 set X_ 500
$n4 set Y_ 100
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20

set n5 [$ns node]
$n5 set X_ 650
$n5 set Y_ 300
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20


set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5
$ns connect $tcp0 $sink5
$tcp0 set packetSize_ 1500

set  ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 3.0 "$ftp0 start"
$ns at 60.0 "$ftp0 stop"

#allow node n3 to  move towards node n1 with speed 5m/sec
$ns at 4.0 "$n3 setdest 300. 500.0 5,0"

#define a finish function
proc finish {} {
                global ns tracefile namfile
                $ns  flush-trace
                close $tracefile
                close $namfile
```

```
        exec nam lab6.nam &
        exec awk -f  lab6.awk lab6.tr &
        exec xgraph lab.dat -geometry 800*400 -t "Packets received by each wirless node" -x "Node Number" -y "Packet Received" &
        exit 0
}
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts\"done\";$ns halt"
$ns run
```

Awk file:

```
BEGIN{
    count1=0;
    count2=0;
    count3=0;
    count4=0;
    count5=0;
}
{
    if( $1=="r" && $3=="_1_" && $4=="RTR")
        count1++;
    if( $1=="r" && $4=="RTR" && $3=="_2_")
        count2++;
    if( $1=="r" && $4=="RTR" && $3=="_3_")
        count3++;
    if( $1=="r" && $4=="RTR" && $3=="_4_")
        count4++;
    if( $1=="r" && $4=="RTR" && $3=="_5_")
        count5++;
    if( $1=="SFESTs")
```

```
        printf("\n%lf\t%d\t%s\t%d\t%s\t%s\t%d\t%s\t%s",$2,$5,$6,$7,$11,$12,$1
3,$14,$15);
}
END{
        printf("\n Packets received by node 1 %d",count1);
        printf("\n Packets received by node 2 %d",count2);
        printf("\n Packets received by node 3 %d",count3);
        printf("\n Packets received by node 4 %d",count4);
        printf("\n Packets received by node 5 %d",count5);
}
```

TERM 7 CN

```
#include<stdio.h>
int main()
{
    int i=0, count=0;
    char databits[80];

    printf(“Enter Data Bits : ”);
    scanf(“%s”,databits);
    printf(“Data Bits Before Bit Stuffing : %s”,databits);
    printf(“\n Data Bits After Bit Stuffing :”);
    for(i=0;i<strlen(databits);i++)
    {
        if(databits[i]==’1’)
            count++;
        else
            count=0;
            printf(“%c”,databits[i])
        if(count==5)
        {
            print(“0”);
            count=0;
        }
    }
    return 0;
}
```

TERM 8 CN

```
#include <stdio.h>
int main()
{
    FILE *fp1, *fp2, *fopen();
    char c;
    int i, cnt=49;
    fp1= fopen(“/root/Desktop/input.txt”,”r”);
    fp2=fopen((“/root/Desktop/output.txt”,”w”);
    if(fp1==NULL)
    {
        printf(“cannot open input.txt\n”);
        exit(1);
    }
    else
    {
        c=getc(fp1);
        while(c!=EOF)
        {
            putc(cnt++,fp2);
            fputs(“192.168.1.1 , 192168.1.2”,fp2);
            for(i=0;i<5;i++)
            {
                putc(c,fp2);
                c=getc(fp1);
            }
            putc(10,fp2)
        }
        printf(“Frames generated in output.txt File”);
        fclose(fp1);
        fclose(fp2);
    }
    return 0;
}
```