Introduction to NS-2:

- Widely known as NS2, is simply an event driven simulationtool.

  - Useful in studying the dynamic nature of communicationnetworks.

  - Simulation of wired as well as wireless network functions and protocols (e.g.,routing algorithms, TCP, UDP) can be done usingNS2.

  - In general, NS2 provides users with a way of specifying such network protocols and simulating their correspondingbehaviors.

Basic Architecture of NS2
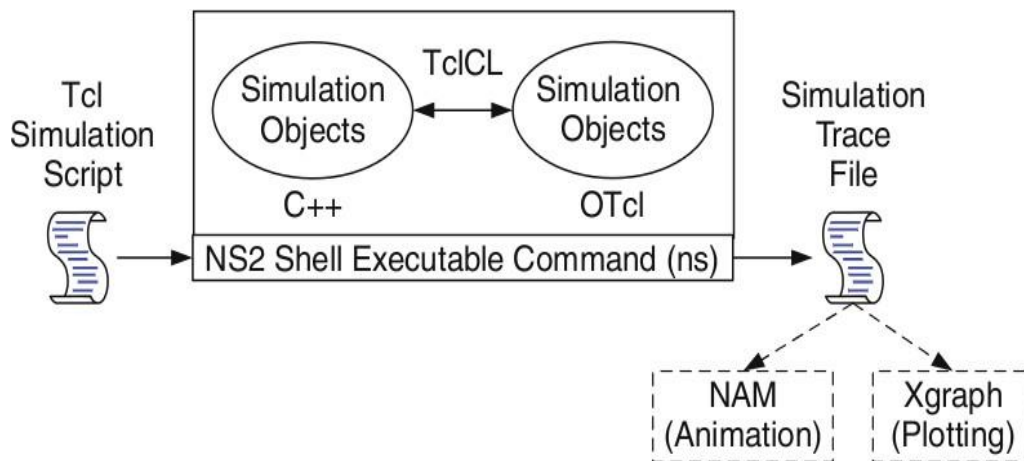
Tcl scripting

- Tcl is a general purpose scripting language.[Interpreter]
- Tcl runs on most of the platforms such as Unix, Windows, andMac.
- The strength of Tcl is itssimplicity.
- It is not necessary to declare a data type for variable prior to theusage.

Basics of TCL

Syntax:command   arg1   arg2   arg3

○ **Hello World!**

    puts stdout{Hello, World!}

    Hello, World!

     ○ **Variables**   CommandSubstitution

seta 5   set len [string lengthfoobar]

setb $a   set len [expr [string length foobar] +9]

○ **SimpleArithmetic**

expr 7.2 / 4

○ **Procedures**

proc Diag {a b} {

set c [expr sqrt($a * $a + $b * $b)]

return $c }

puts―Diagonalofa3,4righttriangleis[Diag34]‖

Output: Diagonal of a 3, 4 right triangle is5.0

○ **Loops**

while{$i <$n} {   for {set i 0} {$i < $n} {incr i}{

. . .   . . .

}   }

Wired TCL Script Components

Create the event scheduler

Open new files & turn on the tracing

Create the nodes

Setup the links

Configure the traffic type (e.g., TCP, UDP, etc)

Set the time of traffic generation (e.g., CBR, FTP)

Terminate the simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the nssimulator.

2. Definition of network nodes, links, queues andtopology.

3. Definition of agents and ofapplications.

4. The nam visualizationtool.

5. Tracing and randomvariables.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

<div style="border:1px solid">

set ns [new Simulator]

</div>

Which is thus the first line in the tcl script? This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), weneed tocreate thefilesusing ‒open‖ command:

#Open the Trace file

<div style="border:1px solid">

set tracefile1 [open out.tr w]

$ns trace-all $tracefile1

</div>

#Open the NAM trace file

<div style="border:1px solid">

set namfile [open out.nam w]

$ns namtrace-all $namfile

</div>

Theabovecreatesadtatracefilecalled ‒out.tr‖andanamvisualizationtracefilecalled ‒out.nam‖.Within the tcl script,these files are not called explicitly by their names,but instead by pointers that are declared above and called ‒tracefile1‖ and ‒namfile‖ respectively.Remark that they begins with a # symbol.The second line open the file ‒out.tr‖ to be used for writing,declared with the letter ‒w‖.The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

The last line tells the simulator to record all simulation traces in NAM input format.It also gives the file name that the trace will be written to later by the command $ns flush-trace.In our case,this will bethe filepointed at bythe pointer‒$namfile‖,i.e the file‒out.tr‖.

Thetermination ofthe program isdoneusinga‒finish‖ procedure.

#Define a „finish‟ procedure

```
Proc finish { } {

global ns tracefile1 namfile

$ns flush-trace

Close $tracefile1

Close $namfile

Exec nam out.nam &

Exit 0
```

The word proc declares a procedure in this case called **finish** and without arguments. The word **global** is used to tell that we are using variables declared outside the procedure. The simulator method ‖**flush-trace"** will dump the traces on the respective files. The tcl command ‖**close"** closes the trace files defined before and **exec** executes the nam program for visualization. The command **exit** will ends the application and return the number 0 as status to the system. Zero is the default for a clean exit. Other values can be used to say that is a exit because something fails.

At the end of ns program we should call the procedure ‖finish‖ and specify at what time the termination should occur. For example,

```
$ns at 125.0 "finish"
```

will be used to call ‖**finish**‖ at time 125sec.Indeed,the **at** method of the simulator allowsus to schedule events explicitly.

The simulation can then begin using the command

```
$ns run
```

Definition of a network of links and nodes

The way to define a node is

```
set n0 [$ns node]
```

The node is created which is printed by the variable n0. When we shall refer to that node in the script we shall thus write $n0.

Once we define several nodes, we can define the links that connect them. An example of a definition of a linkis:

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

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

Todefineadirectionallinkinsteadofabi-directionalone,weshouldreplace‖duplex-link‖by‖simplex-link‖.

In NS, an output queue of a node is implemented as a part of each link whose input is that node. The definition of the link then includes the way to handle overflow at that queue. In our case, if the buffer capacity of the output queue is exceeded then the last packet to arrive is dropped. Many alternative options exist, such as the RED (Random Early Discard) mechanism, the FQ (Fair Queuing), the DRR (Deficit Round Robin), the stochastic Fair Queuing (SFQ) and the CBQ (which including a priority and a round-robin scheduler).

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

> **#set Queue Size of link (n0-n2) to 20**
>
> **$ns queue-limit $n0 $n2 20**

Agents and Applications

We need to define routing (sources, destinations) the agents (protocols) the application that use them.

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are wellreceived.

There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

> **set tcp [new Agent/TCP]**

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.

The command

> **set sink [new Agent /TCPSink]**

Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

#Setup a UDP connection

> **set udp [new Agent/UDP]**
>
> **$ns attach-agent $n1 $udp**
>
> **set null [new Agent/Null]**
>
> **$ns attach-agent $n5 $null**
>
> **$ns connect $udp $null**
>
> **$udp set fid_2**

#setup a CBR over UDP connection

> **set cbr [new Application/Traffic/CBR]**
>
> **$cbr attach-agent $udp**
>
> **$cbr set packetsize_ 100**
>
> **$cbr set rate_ 0.01Mb**
>
> **$cbr set random_ false**

Above shows the definition of a CBR application using a UDP agent

The command **$ns attach-agent $n4 $sink** defines the destination node. The command **$ns connect $tcp $sink** finally makes the TCP connection between the source and destination nodes.

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be changed to another value, say 552bytes, using the command **$tcp set packetSize_ 552**.

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command **$tcp set fid_ 1** that assigns to the TCP connection a flow identification of ‒1‖.We shall later give the flow identification of‒2‖to the UDP connection.

CBR over UDP

A UDP source and destination is defined in a similar way as in the case of TCP.

Instead of defining the rate in the command $cbr set rate_ 0.01Mb, one can define the time interval between transmission of packets using the command.

> **$cbr set interval_ 0.005**

The packet size can be set to some value using

> **$cbr set packetSize_ <packet size>**

Scheduling Events

NS is a discrete event based simulation. The tcp script defines when event should occur. The initializing command set ns [new Simulator] creates an event scheduler, and events are then scheduled using the format:

> **$ns at <time><event>**

The scheduler is started when running ns that is through the command $ns run.

The beginning and end of the FTP and CBR application can be done through the following command

> **$ns at 0.1 "$cbrstart"**
>
> **$ns at 1.0 " $ftpstart"**
>
> **$ns at 124.0 "$ftpstop"**
>
> **$ns at 124.5 "$cbrstop"**

Structure of Trace Files

When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown below, The meaning of the fieldsare:

Event	Time	From Node	To Node	PKT Type	PKT Size	Flags	Fid	Src Addr	Dest Addr	Seq Num	Pkt id

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the eventoccurs.
3. Gives the input node of the link at which the eventoccurs.
4. Gives the output node of the link at which the eventoccurs.
5. Gives the packet type (eg CBR orTCP)
6. Gives the packetsize
7. Someflags
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAMdisplay.
9. This is the source addressgiven inthe form of—node.port‖.
10. This is the destination address, given in the same form.
11. This is the network layer protocol's packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysispurposes
12. The last field shows the Unique id of thepacket.

XGRAPH

The xgraph program draws a graph on an x-display given data read from either data file or from standard input if no files are specified. It can display upto 64 independent data sets using different colors and line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels and alegend.

Syntax:

> **Xgraph [options] file-name**

Options are listed here

/-bd <color> (Border)

This specifies the border color of the xgraph window.

/-bg <color> (Background)

This specifies the background color of the xgraph window.

/-fg<color> (Foreground)

This specifies the foreground color of the xgraph window.

/-lf <fontname> (LabelFont)

All axis labels and grid labels are drawn using this font.

/-t<string> (Title Text)

This string is centered at the top of the graph.

/-x <unit name> (XunitText)

This is the unit name forthe x-axis. Its default is ─X‖.

/-y <unit name> (YunitText)

This is the unit name forthey-axis.Its default is ─Y‖.

Awk- An Advanced

awk is a programmable, pattern-matching, and processing tool available in UNIX. It works equally well with text and numbers.

awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files to see if they contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or incrementing a counter each time it finds a match.

Syntax:

> **awk option 'selection_criteria {action}' file(s)**

Here, selection_criteria filters input and select lines for the action component to act upon. The selection_criteria is enclosed within single quotes and the action within the curly braces. Both the selection_criteria and action forms an awk program.

Example: $ awk „/manager/ {print}" emp.lst

Variables

Awk allows the user to use variables of there choice. You can now print a serial number, using the variable kount, and apply it those directors drawing a salary exceeding 6700:

$ awk –F"|" „$3 == "director" && $6 > 6700 {

kount =kount+1

printf"%3f%20s%-12s%d\n",kount,$2,$3,$6}"empn.lst

THE –f OPTION: STORING awk PROGRAMS IN AFILE

You should holds large awk programs in separate file and provide them with the awk extension for easier identification. Let's first store the previous program in the file empawk.awk:

$ cat empawk.awk

Observe that this time we haven't used quotes to enclose the awk program. You can now use awk with the –f *filename* option to obtain the same output:

> **Awk –F"|" –f empawk.awk empn.lst**

THE BEGIN AND END SECTIONS

Awk statements are usually applied to all lines selected by the address, and if there are no addresses, then they are applied to every line of input. But, if you have to print something before processing the first line, for example, a heading, then the BEGIN section can be used gainfully. Similarly, the end section useful in printing some totals after processing is over.

The BEGIN and END sections are optional and take the form

BEGIN {action}

END {action}

These two sections, when present, are delimited by the body of the awk program. You can use them to print a suitable heading at the beginning and the average salary at theend.

BUILT-IN VARIABLES

Awk has several built-in variables. They are all assigned automatically, though it is also possible for a user to reassign some of them. You have already used NR, which signifies the record number of the current line. We'll now have a brief look at some of the other variable.

The FS Variable: as stated elsewhere, awk uses a contiguous string of spaces as the default field delimiter. FS redefines this field separator, which in the sample database happens to be the |. When used at all, it must occur in the BEGIN section so that the body of the program knows its value before it starts processing:

BEGIN {FS="|"}

This is an alternative to the –F option which does the samething.

The OFS Variable: when you used the print statement with comma-separated arguments, each argument was separated from the other by a space. This is awk's default output field separator, and can reassigned using the variable OFS in the BEGINsection:

BEGIN { OFS="~" }

When you reassign this variable with a ~ (tilde), awk will use this character for delimiting the print arguments. This is a useful variable for creating lines with delimited fields.

The NF variable: NF comes in quite handy for cleaning up a database of lines that don't contain the right number of fields. By using it on a file, say emp.lst, you can locate those lines not having 6 fields, and which have crept in due to faulty data entry:

$awk „BEGIN {FS = "|"}

NF! =6 { Print "Record No ", NR, "has", "fields"}" empx.lst

ExperimentNo:1

Simulate a three node point to point network with duplex links between them. Set queue size and vary the bandwidth and find number of packets dropped.

```
# Simulation parameters setup
set val(stop) 6.0 ; # stopping time of the simulation

# Initialization
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open 1.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open 1.nam w]
$ns namtrace-all $namfile

# Nodes Definition
#Create 3 nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Links Definition
#Createlinks between nodes
$ns duplex-link $n1 $n2 1000Kb 60ms DropTail
$ns queue-limit $n1 $n2 14
$ns duplex-link $n2 $n3 500Kb 60ms DropTail
$ns queue-limit $n2 $n3 4
$ns duplex-link-op $n1 $n2 queuePos 0.5
$ns duplex-link-op $n2 $n3 queuePos 0.2

# Agents Definition
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n1 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500

# Applications Definition
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
```

```
$ns at 0.2 "$ftp0 start"
$ns at 5.0 "$ftp0 stop"

# Termination
#Define a 'finish' procedure
proc finish { } {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam 1.nam &
exit 0
}

$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:*(Open a new editor using "vi command" and write awk file and save with ".awk" extension)*
#immediately after BEGIN should open braces „{„

```
BEGIN {
count=0;
total=0;
}
{
event=$1;
if(event=="d") {
count++;
}
}
END {
printf("No of packets dropped : %d\n",count);
}
```

Steps for execution

- *Openvieditorandtypeprogram.Programnameshouldhavetheextension".tcl "*

  [root@localhost ~]# vi lab1.tcl
- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enterkey**.*
- *Open vi editor and type **awk** program. Program name should have the extension ".awk"*

  [root@localhost ~]# vi lab1.awk

- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enterkey**.*
- *Run the simulationprogram*
  > **[root@localhost~]# ns lab1.tcl**
- *Here **"ns"** indicates network simulator. We get the topology shown in the snapshot.*
- *Now press the play button in the simulation window and the simulation will begins.*
- *After simulation is completed run **awk file** to see the output,*
  > **[root@localhost~]# awk –f lab1.awk1.tr**
- *To see the trace file contents open the file as,*
  > **[root@localhost~]# vi 1.tr**

___Trace file contains 12 columns:___
Event type, Event time, From Node, To Node, Packet Type, Packet Size, Flags (indicated by --------), Flow ID, Source address, Destination address, Sequence ID, PacketID

| Contents ofTraceFile | Topology | Output |

ExperimentNo:2

Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

set val(stop) 10.0 ; # time of simulation end

#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open 3.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open 3.nam w]
$ns namtrace-all $namfile

#Create 7 nodes
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 links between nodes
$ns duplex-link $n0 $n1 1Mb 50ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n0 $n3 1Mb 50ms DropTail
$ns queue-limit $n0 $n3 50
$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns queue-limit $n0 $n4 50
$ns duplex-link $n0 $n5 1Mb 50ms DropTail
$ns queue-limit $n0 $n5 2
$ns duplex-link $n0 $n2 1Mb 50ms DropTail
$ns queue-limit $n0 $n2 2
$ns duplex-link $n0 $n6 1Mb 50ms DropTail
$ns queue-limit $n0 $n6 1

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right-up
$ns duplex-link-op $n0 $n2 orient right
$ns duplex-link-op $n0 $n3 orient right-down
$ns duplex-link-op $n0 $n4 orient left-down
$ns duplex-link-op $n0 $n5 orient left
$ns duplex-link-op $n0 $n6 orient left-up

```
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] received ping answer from $from with round-trip-time $rtt ms."
}

set p1 [new Agent/Ping]
set p2 [new Agent/Ping]
set p3 [new Agent/Ping]
set p4 [new Agent/Ping]
set p5 [new Agent/Ping]
set p6 [new Agent/Ping]

$ns attach-agent $n1 $p1
$ns attach-agent $n2 $p2
$ns attach-agent $n3 $p3
$ns attach-agent $n4 $p4
$ns attach-agent $n5 $p5
$ns attach-agent $n6 $p6

$ns connect $p1 $p4
$ns connect $p2 $p5
$ns connect $p3 $p6

$ns at 0.2 "$p1 send"
$ns at 0.4 "$p2 send"
$ns at 0.6 "$p3 send"
$ns at 1.0 "$p4 send"
$ns at 1.2 "$p5 send"
$ns at 1.4 "$p6 send"

proc finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam 3.nam &
exit 0
}

$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 {
count=0;
```

```
}
{
event=$1;
if(event=="d") {
count++;
}
}
END {
printf("No of packets dropped : %d\n",count);
}
```

Steps for execution:

1) *Open vi editor and type program. Program name should have the extension " .tcl"*
       [root@localhost ~]# vi lab2.tcl
2) *Save the program by pressing* ***"ESC key"*** *first, followed by* ***"Shift and :"*** *keys simultaneously and type* ***"wq"*** *and press* ***Enterkey***.
3) *Open vi editor and type* ***awk*** *program. Program name should have the extension "***.awk***"*
       [root@localhost ~]# vi lab2.awk
4) *Save the program by pressing* ***"ESC key"*** *first, followed by* ***"Shift and :"*** *keys simultaneously and type* ***"wq"*** *and press* ***Enterkey***.
5) *Run the simulationprogram*
       [root@localhost~]# ns lab2.tcl
    i)   *Here* ***"ns"*** *indicates network simulator. We get the topology shown in the snapshot.*
    ii)   *Now press the play button in the simulation window and the simulation will begins.*
6) *After simulation is completed run* ***awk file*** *to see the output,*
       [root@localhost~]# awk –f lab2.awk 3.tr
7) *To see the trace file contents open the file as,*
       [root@localhost~]# vi 3.tr

ExperimentNo:3

Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source /destination


```
# Simulation parameters setup
set val(stop) 10.0 ;# time of simulation end

# Initialization
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open 3.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open 3.nam w]
$ns namtrace-all $namfile
$ns color 1 Blue
$ns color 2 Red

# Nodes Definition
#Create 9 nodes
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]

# Create LAN
$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 50ms LL Queue/DropTail

# Links Definition
#Createlinks between nodes
$ns duplex-link $n1 $n0 2.0Mb 50ms DropTail
$ns queue-limit $n1 $n0 50
$ns duplex-link $n2 $n0 2.0Mb 50ms DropTail
$ns queue-limit $n2 $n0 50
$ns duplex-link $n0 $n3 1.0Mb 50ms DropTail
$ns queue-limit $n0 $n3 7
#Give node position (for NAM)
$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
```
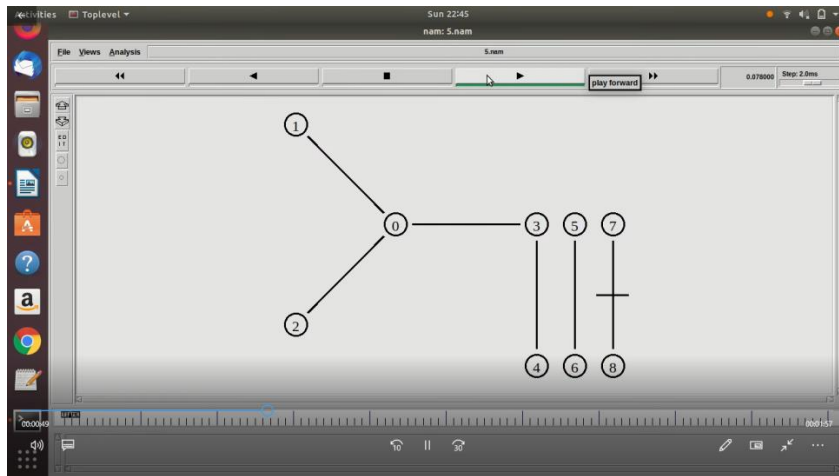
$ns duplex-link-op $n0 $n3 orient right

Agents Definition
#Setup a TCP/Reno connection
set tcp0 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp0
set sink1 [new Agent/TCPSink]

$ns attach-agent $n7 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500
$tcp0 set class_ 1
set tfile1 [open cwnd1.tr w]
$tcp0 attach $tfile1
$tcp0 trace cwnd_

#Setup a TCP/Vegas connection
set tcp5 [new Agent/TCP/Vegas]
$ns attach-agent $n2 $tcp5
set sink6 [new Agent/TCPSink]
$ns attach-agent $n8 $sink6
$ns connect $tcp5 $sink6
$tcp5 set packetSize_ 1500
$tcp5 set class_ 2
set tfile2 [open cwnd2.tr w]
$tcp5 attach $tfile2
$tcp5 trace cwnd_

Applications Definition
#Setup a FTP Application over TCP/Reno connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 0.3 "$ftp0 start"
$ns at 8.0 "$ftp0 stop"

#Setup a FTP Application over TCP/Vegas connection
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp5
$ns at 0.3 "$ftp4 start"
$ns at 8.0 "$ftp4 stop"

Termination
#Define a 'finish' procedure
proc finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam 3.nam &
exit 0
}

$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:*(Open a new editor using "vi command" and write awk file and save with ".awk" extension)*
BEGIN{}
{
if($6=="cwnd_")
{ printf("%f\t%f\n",$1,$7); }
}
END{}

Steps for execution

- *Open vi editor and type program. Program name should have the extension " .tcl"*
  [root@localhost ~]# vi lab3.tcl
- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enterkey**.*
- *Open vi editor and type **awk** program. Program name should have the extension ".awk"*
  [root@localhost ~]# vi lab3.awk
- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enterkey**.*
- *Run the simulationprogram*
  [root@localhost~]# ns lab3.tcl
    - *After simulation is completed run **awk file** to see the output , **[root@localhost~]# awk –f lab3.awk cwnd1.tr > a1 [root@localhost~]# awk –f lab3.awk cwnd2.tr >a2 [root@localhost~]# xgraph a1a2***

- *Here we are using the congestion window trace files i.e. **cwnd1.tr** and **cwnd2.tr** and we are redirecting the contents of those files to new files say **a1** and **a2** using **output redirection operator(>)**.*
- *To see the trace file contents open the file as,*
    [root@localhost~]# vi 3.tr

ExperimentNo:4

Simulate simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.

```
# Simulation parameters setup
set val(chan) Channel/WirelessChannel      ;# channel type
set val(prop) Propagation/TwoRayGround    ;# radio-propagation model
set val(netif) Phy/WirelessPhy             ;# network interface type
set val(mac) Mac/802_11           ;# MAC type
set val(ifq) Queue/DropTail/PriQueue       ;# interface queue type
set val(ll) LL                    ;# link layer type
set val(ant) Antenna/OmniAntenna    ;# antenna model
set val(ifqlen) 50                 ;# max packet in ifq
set val(nn) 2                      ;# number of mobilenodes
set val(rp) DSDV                   ;# routing protocol
set val(x) 700                     ;# X dimension of topography
set val(y) 444                     ;# Y dimension of topography
set val(stop) 10.0                 ;# time of simulation end

# Initialization
#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 out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

# Mobile node parameter setup
$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


# Nodes Definition
#Create 2 nodes
set n0 [$ns node]
$n0 set X_ 268
$n0 set Y_ 339
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 428
$n1 set Y_ 344
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20

# Generate movement
$ns at .1 " $n0 setdest 600 344 100 "
$ns at .1 " $n1 setdest 300 339 100 "

# Agents Definition
#Setup a TCP connection
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

# Applications Definition
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 5.0 "$ftp0 stop"

#Define a 'finish' procedure
proc finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam out.nam &
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:*(Open a new editor using "vi command" and write awk file and save with ".awk" extension)*

```
BEGIN{
count=0;
total=0;
}
{
event =$1;
if(event=="D") {
count++;
}
}
END{
printf("No of packets dropped : %d\n",count);
}
```

Steps for execution

- *Open vi editor and type program. Program name should have the extension " .tcl"*
  [root@localhost ~]# vi lab4.tcl
- *Save the program by pressing "ESC key" first, followed by "Shift and :" keys simultaneously and type "wq" and press Enterkey.*
- *Open vi editor and type awk program. Program name should have the extension ".awk"*
  [root@localhost ~]# vi lab4.awk
- *Save the program by pressing "ESC key" first, followed by "Shift and :" keys simultaneously and type "wq" and press Enterkey.*
- *Run the simulationprogram*
  [root@localhost~]# ns lab4.tcl
  - *Here "ns" indicates network simulator. We get the topology shown in the snapshot.*
  - *Now press the play button in the simulation window and the simulation will begins.*
- *After simulation is completed run awk file to see the output,*
  [root@localhost~]# awk –f lab4.awk out.tr
- *To see the trace file contents open the file as,*
  [root@localhost~]# vi out.tr

ExperimentNo:5

Congestion Control Using Leaky Bucket Algorithm

Aim: *C Program for Congestion control using Leaky Bucket Algorithm*

The main concept of the leaky bucket algorithm is that the output data flow remains constant despite the variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water (or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero. From network perspective, leaky bucket consists of a finite queue (bucket) where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, leaky bucket transmits one packet from the queue in a fixed time (e.g. at every clock tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two approaches (e.g. leaky bucket with water (a) and with packets(b)).
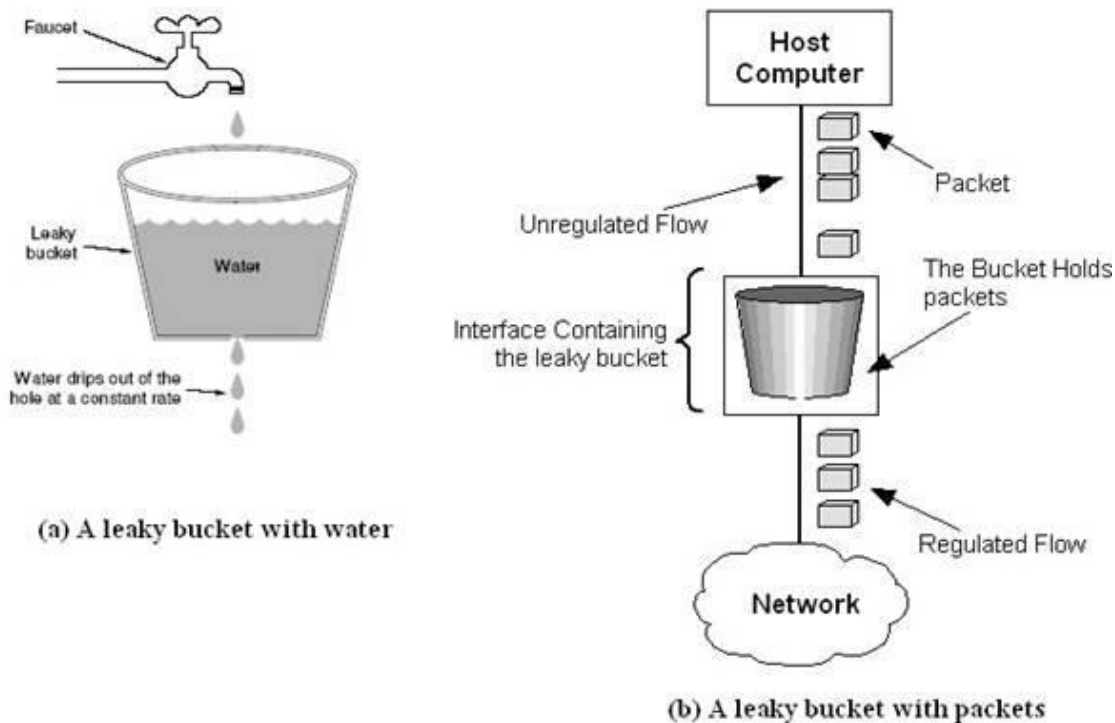
Figure 2.4 - The leaky bucket traffic shaping algorithm

While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into

account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

Implementation Algorithm:

Steps:

1. Read The Data ForPackets
2. Read The QueueSize
3. Divide the Data intoPackets
4. Assign the random Propagation delays for each packets to input into thebucket (input_packet).

5. wlile((Clock++<5*total_pack

ets)and

(out_packets<total_paclets))

a. if (clock ==input_packet)

i. insert intoQueue

b. if (clock % 5 == 0)

i. Remove paclet fromQueue

6. End

Source Code:

```
#include<stdio.h>
#include<strings.h>
#include<stdio.h>
int min(int x,int y)
{
if(x<y)
return x;
else
return y;
}
int main()
{
int drop=0,mini,nsec,cap,count=0,i,inp[25],process;
syste("clear");
```

```
printf("Enter The Bucket Size\n");
scanf("%d",&cap);
printf("Enter The Operation
Rate\n"); scanf("%d",&process);
printf("Enter The No. Of Seconds You Want To Stimulate\n");
scanf("%d",&nsec);
for(i=0;i<nsec;i++)
{
printf("Enter The Size Of The Packet Entering At %d
sec\n",i+1);
scanf("%d",&inp[i]);
}
printf("\nSecond|Packet Recieved|Packet Sent|Packet
Left|Packet Dropped|\n");
printf("
---------\n");
for(i=0;i<nsec;i++)
{
count+=inp[i];
if(count>cap)
{
drop=count-cap;
count=cap;
}
printf("%d",i+1);
printf("\t%d",inp[i]);
mini=min(count,process);
printf("\t\t%d",mini);
count=count-mini;
printf("\t\t%d",count);
printf("\t\t%d\n",drop);
drop=0;
}
for(;count!=0;i++)
{
if(count>cap)
{
drop=count-cap;
count=cap;
}
printf("%d",i+1);
```

```
printf("\t0");
mini=min(count,process);
printf("\t\t%d",mini);
count=count-mini;
printf("\t\t%d",count);
printf("\t\t%d\n",drop);
}}
```

Output:
Compile and run
$ cc –o Congestion Congestion.c
$ ./Congestion
Enter The Bucket Size
5
Enter The Operation Rate
2
Enter The No. Of Seconds You Want To Stimulate
3
Enter The Size Of The Packet Entering At 1 sec
5
Enter The Size Of The Packet Entering At 1 sec
4
Enter The Size Of The Packet Entering At 1 sec
3
Second|Packet Recieved|Packet Sent|Packet Left|Packet Dropped|

Second	Packet Recieved	Packet Sent	Packet Left	Packet Dropped
1	5	2	3	0
2	4	2	3	2
3	3	2	3	1
4	0	2	1	0
5	0	1	0	0

ExperimentNo:6

Distance Vector Algorithm

Aim: C Program for Distance Vector Algorithm to find suitable path for transmition

Distance Vector Algorithm is a decentralized routing algorithm that requires that each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement. To find the shortest path, Distance Vector Algorithm is based on one of two basic algorithms: the Bellman-Ford and the Dijkstra algorithms.

Routers that use this algorithm have to maintain the distance tables (which is a one-dimension array -- "a vector"), which tell the distances and shortest path to sending packets to each node in the network. The information in the distance table is always upd by exchanging information with the neighboring nodes. The number of data in the table equals to that of all nodes in networks (excluded itself). The columns of table represent the directly attached neighbors whereas the rows represent all destinations in the network. Each data contains the path for sending packets to each destination in the network and distance/or time to transmit on that path (we call this as "cost"). The measurements in this algorithm are the number of hops, latency, the number of outgoing packets, etc.

The starting assumption for distance-vector routing is each node knows the cost of the link of each of its directly connected neighbors. Next, every node sends a configured message to its directly connected neighbors containing its own distance table. Now, every node can learn and up its distance table with cost and next hops for all nodes network. Repeat exchanging until no more information between theneighbors.

Consider a node A that is interested in routing to destination H via a directly attached neighbor J. Node A's distance table entry, $D_x(Y,Z)$ is the sum of the cost of the direct-one hop link between A and J, $c(A,J)$, plus neighboring J's currently known minimum-cost path (shortest path) from itself(J) to H. That is $D_x(H,J) = c(A,J) + min_w\{D_j(H,w)\}$ The minw is taken over all the J's This equation suggests that the form of neighbor-to-neighbor communication that will take place in the DV algorithm - each node must know the cost of each of its neighbors' minimum-cost path to each destination. Hence, whenever a node computes a new minimum cost to some destination, it must inform its neighbors of this new minimumcost.

To	A	I	H	K		Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K

JA delay is 8   JI delay is 10   JH delay is 12   JK delay is 6

New routing table for J

Vectors received from J's four neighbors

(b)
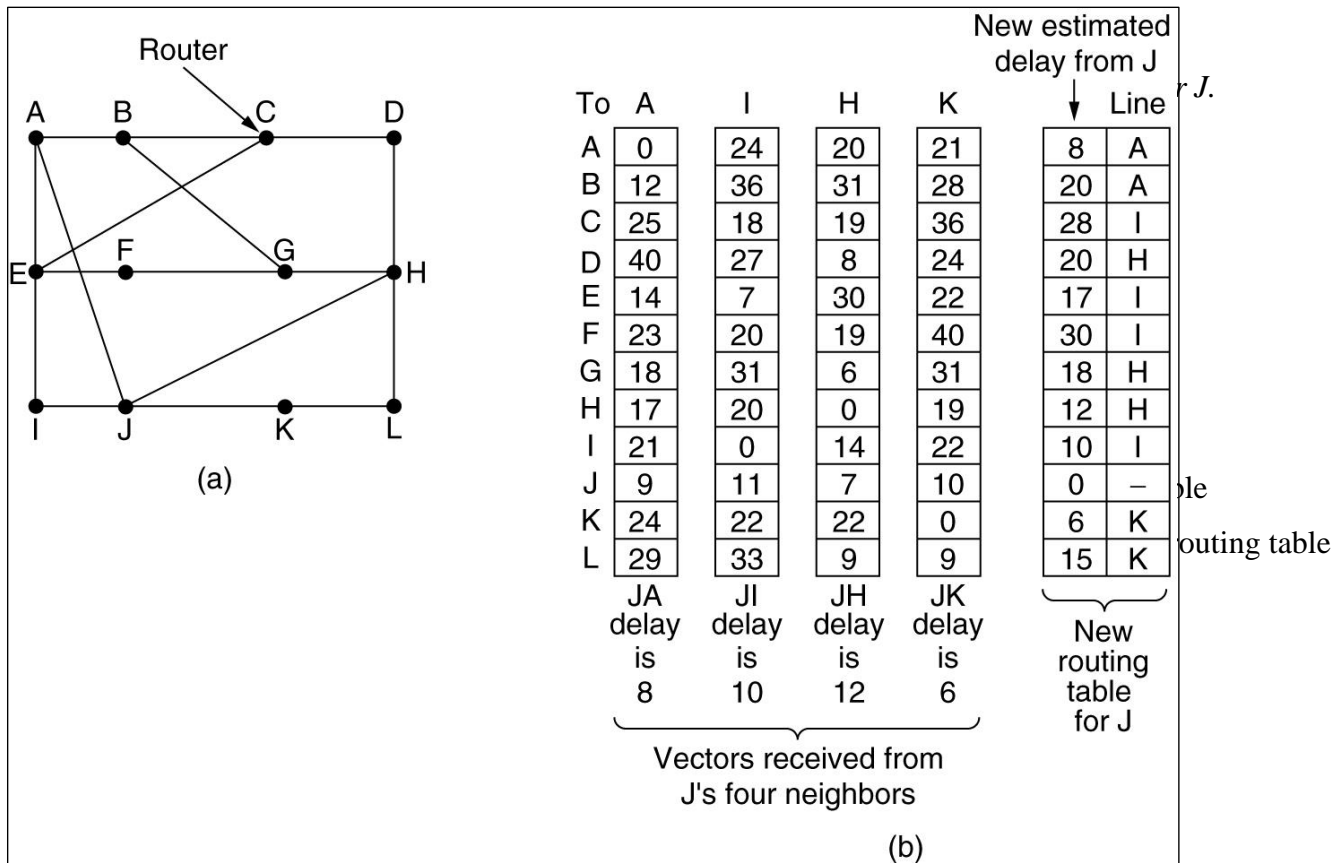
```c
#include<stdio.h>
#include<stdlib.h>

void rout_table();
int d[10][10],via[10][10];
int i,j,k,l,m,n,g[10][10],temp[10][10],ch,cost;

int main()
{
        system("clear");
    printf("enter the value of no. of nodes\n");
    scanf("%d",&n);

    rout_table();
    for(i=0;i<n;i++)
    for(j=0;j<n;j++)
            temp[i][j]=g[i][j];
    for(i=0;i<n;i++)
    for(j=0;j<n;j++)
            via[i][j]=i;
    while(1)
    {
```

```
            for(i=0;i<n;i++)
             for(j=0;j<n;j++)
                   if(d[i][j])
            for(k=0;k<n;k++)
                   if(g[i][j]+g[j][k]<g[i][k])
                   {
                           g[i][k]=g[i][j]+g[j][k];
                           via[i][k]=j;
                   }
            for(i=0;i<n;i++)
            {
                   printf("table for router %c\n" ,i+97);
                   for(j=0;j<n;j++)
                           printf("%c:: %d via %c\n" ,j+97,
                                                 g[i][j],via[i][j]+97);
                   }
            Break;
            }
        }
}


    void rout_table()
    {
        printf("\nEnter the routing table : \n");
            printf("\t|");
        for(i=1;i<=n;i++)
            printf("%c\t",i+96);
        printf("\n");
        for(i=0;i<=n;i++)
            printf(" ------ ");
        printf("\n");
        for(i=0;i<n;i++)
        {
            printf("%c      |",i+97);
            for(j=0;j<n;j++)
            {
                scanf("%d",&g[i][j]);
                    if(g[i][j]!=999)
                    d[i][j]=1;
```

```
                }
            }
    }
```

Output:
[root@localhost]# cc prg3.c
[root@localhost]# ./a.out

enter the value of no. of nodes
4
Enter the routing table :
```
        |a    b    c    d
-----------------------
a    |0    5    1    4
b    |5    0    6    2
c    |1    6    0    3
d    |4    2    3    0
```

table for router a
a:: 0 viaa
b:: 5 via a
c:: 1 via a
d:: 4 via a

table for router b
a:: 5 viab
b:: 0 via b
c:: 5 via d
d:: 2 via b

table for router c
a:: 1 viac
b:: 5 via d
c:: 0 via c
d:: 3 via c

table for router d
a:: 4 viad
b:: 2 via d
c:: 3 via d
d:: 0 via d

do you want to change the cost(1/0)

1

enter the vertices which you want to change the cost

1 3

enter the cost

2

table for router a

a:: 0 viaa

b:: 5 via a

c:: 2 via a

d:: 4 via a

table for router b

a:: 5 viab

b:: 0 via b

c:: 5 via d

d:: 2 via b

table for router c

a:: 2 viac

b:: 5 via d

c:: 0 via c

d:: 3 via c

table for router d

a:: 4 viad

b:: 2 via b

c:: 3 via d

d:: 0 via d

 do you want to change the cost(1/0)

0

ExperimentNo:7

RSA Algorithm to Encrypt and Decrypt the Data

Aim: C Program for Simple RSA Algorithm to encrypt and decrypt the data

The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

The RSA algorithm's efficiency requires a fast method for performing the modular exponentiation operation. A less efficient, conventional method includes raising a number (the input) to a power (the secret or public key of the algorithm, denoted *e* and *d*, respectively) and taking the remainder of the division with *N*. A straight-forward implementation performs these two steps of the operation sequentially: first, raise it to the power and second, apply modulo.

A very simple example of RSA encryption

This is an extremely simple example using numbers you can work out on a pocket calculator (those of you over the age of 35 can probably even do it by hand on paper).

1. Select primes p = 11, q = 3.

2. n = pq = 11.3 = 33
   phi = (p-1)(q-1) = 10.2 = 20

3. Choosee=3
   Check gcd(e, p-1) = gcd(3, 10) = 1 (i.e. 3 and 10 have no common factors except 1),
   and check gcd(e, q-1) = gcd(3, 2) = 1
   therefore gcd(e, phi) = gcd(e, (p-1)(q-1)) = gcd(3, 20) = 1

4. Compute d such that ed ≡ 1 (modphi)
   i.e. compute d = e^{-1} mod phi = 3^{-1} mod 20
   i.e. find a value for d such that phi divides (ed-1)
   i.e. find d such that 20 divides 3d-1.
   Simple testing (d = 1, 2, ...) gives d = 7
   Check: ed-1 = 3.7 - 1 = 20, which is divisible by phi.

5. Public key = (n, e) = (33, 3)
   Private key = (n, d) = (33,7).

   This is actually the smallest possible value for the modulus n for which the RSA algorithm works.

   Now say we want to encrypt the message m = 7, c = m^e mod n = 7^3 mod 33 = 343 mod 33 = 13. Hence the ciphertext c = 13.

To check decryption we compute

m' = c$^{\wedge d}$ mod n = 13$^{\wedge 7}$ mod 33 = 7.

Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that a = bc mod n = (b mod n).(c mod n) mod n so we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the finalvalue.

One way of calculating m' is as follows:-

m' = 13$^{\wedge 7}$ mod 33 = 13$^{\wedge(3+3+1)}$ mod 33 = 13$^{\wedge 3}$.13$^{\wedge 3}$.13 mod 33

= (13$^{\wedge 3}$ mod 33).(13$^{\wedge 3}$ mod 33).(13 mod 33) mod 33

= (2197 mod 33).(2197 mod 33).(13 mod 33) mod 33

= 19.19.13 mod 33 = 4693 mod 33

= 7.

Now if we calculate the cipher text c for all the possible values of m (0 to 32), we get

m 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

c 0 1 8 27 31 26 18 13 17 3 10 11 12 19 5 94

m 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

c 29 24 28 14 21 22 23 30 16 20 15 7 2 6 25 32

Note that all 33 values of m (0 to 32) map to a unique code c in the same range in a sort of random manner. In this case we have nine values of m that map to the same value of c - these are known as *unconcealed messages*. m = 0 and 1 will always do this for any N, no matter how large. But in practice, higher values shouldn't be a problem when we use large values forN.

If we wanted to use this system to keep secrets, we could let A=2, B=3, ..., Z=27. (We specifically avoid 0 and 1 here for the reason given above). Thus the plaintext message "HELLOWORLD" would be represented by the set of integers m1, m2, ...

{9,6,13,13,16,24,16,19,13,5}

Using our table above, we obtain ciphertext integers c1, c2, ...

{3,18,19,19,4,30,4,28,19,26}

Note that this example is no more secure than using a simple Caesar substitution cipher, but it serves to illustrate a simple example of the mechanics of RSA encryption.

Remember that calculating m^e mod n is easy, but calculating the inverse c^-e mod n is very difficult, well, for large n's anyway. However, if we can factor n into its prime factors p and

q, the solution becomes easy again, even for large n's. Obviously, if we can get hold of the secret exponent d, the solution is easy, too.

Key GenerationAlgorithm

1. Generate two large random primes, p and q, of approximately equal size suchthat their product n = pq is of the required bit length, e.g. 1024 bits. [See note1].
2. Compute n = pq and (φ) phi =(p-1)(q-1).
3. Choose an integer e, 1 < e < phi, such that gcd(e, phi) = 1. [See note 2].
4. Compute the secret exponent d, 1 < d < phi, such that
   ed ≡ 1 (mod phi). [See note3].
5. The public key is (n, e) and the private key is (n, d). The values of p, q, andphi should also be keptsecret.

- n is known as the*modulus*.
- e is known as the *public exponent* or *encryptionexponent*.
- d is known as the *secret exponent* or *decryptionexponent*.

Encryption

Sender A does the following:-

1. Obtains the recipient B's public key (n,e).
2. Represents the plaintext message as a positive integer m [see note4].
3. Computes the ciphertext $c = m^e$ modn.
4. Sends the ciphertext c toB.

Decryption

Recipient B does the following:-

1. Uses his private key (n, d) to compute $m = c^d$ modn.
2. Extracts the plaintext from the integer representativem.

Source Code:

```
#include<math.h>
#include<stdlib.h>
#include<stdio.h>

int gcd(long m,long n)
{
        while(n!=0)
        {
```

```
        }              long r=m%n;
           return m; m=n;
     }                 n=r;



     int rsa(char message[50])
     {       longp=0,q=0,n=0,e=0,d=0,phi=0;
             longnummes[100]={0};
             longencrypted[100]={0},decrypted[100]={0};
             longi=0,j=0,nofelem=0;

             printf("\nenter value of p and q\n");
             scanf("%d%d",&p,&q);
             n=p*q;
             phi=(p-1)*(q-1);

             for(i=2;i<phi;i++)
             if(gcd(i,phi)==1) break;
             e=i;

             for(i=2;i<phi;i++)
             if((e*i-1)%phi==0)break;
             d=i;
             for(i=0;i<strlen(message);i++)
             nummes[i]=message[i]-96;
             nofelem=strlen(message);

             for(i=0;i<nofelem;i++)
             {
                     encrypted[i]=1;
                     for(j=0;j<e;j++)
                     encrypted[i] =(encrypted[i]*nummes[i])%n;
             }
             printf("\n Encrypted message\n");
             for(i=0;i<nofelem;i++)
             {
                     printf(" %ld ",encrypted[i]);
                     printf("%c",(char)(encrypted[i])+96);
             }

             for(i=0;i<nofelem;i++)
```

```
        {
                decrypted[i]=1;
                for(j=0;j<d;j++)
                decrypted[i]=(decrypted[i]*encrypted[i])%n;
        }

        printf("\n Decrypted message\n ");
        for(i=0;i<nofelem;i++)
        printf("%c",(char)(decrypted[i]+96));
        return 0;
}
int main()
{       char msg[];
        clrscr();
        printf("Enter The Message To Be Encrypted\n");
        scanf("%s",msg);
        rsa(msg);
        return 0;
}
```

#***
RESULT
[root@localhost]# cc prg7.c
[root@localhost]# ./a.out

Enter the text:
hello
Enter the value of P and Q :
5
7
Encrypted Text is: 8 h 10 j 17 q 17 q 15 o
Decrypted Text is: hello

ExperimentNo:8

Client-server socket programming

Aim: *Using TCP/IP Sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.*

Sockets are a protocol independent method of creating a connection between processes. Sockets can be either

- *Connection based or connectionless*: Is a connection established before communication or does each packet describe thedestination?
- *Packet based or streams based*: Are there message boundaries or is it onestream?
- *Reliable or unreliable:* Can messages be lost, duplicated, reordered, orcorrupted?

Socket characteristics

Sockets are characterized by their domain, type and transport protocol. Common domains are:

- AF_UNIX: address format is UNIX pathname
- AF_INET: address format is host and portnumber

Common types are:

- *virtual circuit*: received in order transmitted andreliably
- *datagram*: arbitrary order,unreliable

Each socket type has one or more protocols. Ex:

- TCP/IP (virtualcircuits)
- UDP(datagram)

Use of sockets:

- Connection–based sockets communicate client-server: the server waits for aconnection from theclient
- Connectionless sockets are peer-to-peer: each process issymmetric.

Socket APIs

- *socket*: creates a socket of a given domain, type, protocol (buy aphone)
- *bind*: assigns a name to the socket (get a telephonenumber)
- *listen*: specifies the number of pending connections that can be queued for aserver socket. (call waitingallowance)
- *accept*: server accepts a connection request from a client (answerphone)
- *connect*: client requests a connection request to a server(call)
- *send, sendto*: write to connection(speak)
- *recv, recvfrom*: read from connection(listen)
- *shutdown*: end thecall

Connection-based communication

Server performs the following actions

- *socket*: create thesocket
- *bind*: give the address of the socket on theserver
- *listen*: specifies the maximum number of connection requests that can be pending forthis process
- *accept*: establish the connection with a specificclient
- *send, recv*: stream-based equivalents of read and write(repeated)
- *shutdown*: end reading orwriting
- *close*: release kernel datastructures

Client performs the following actions

- *socket*: create thesocket
- *connect*: connect to aserver
- *send, recv*:(repeated)
- *shutdown*
- *close*

TCP-based sockets
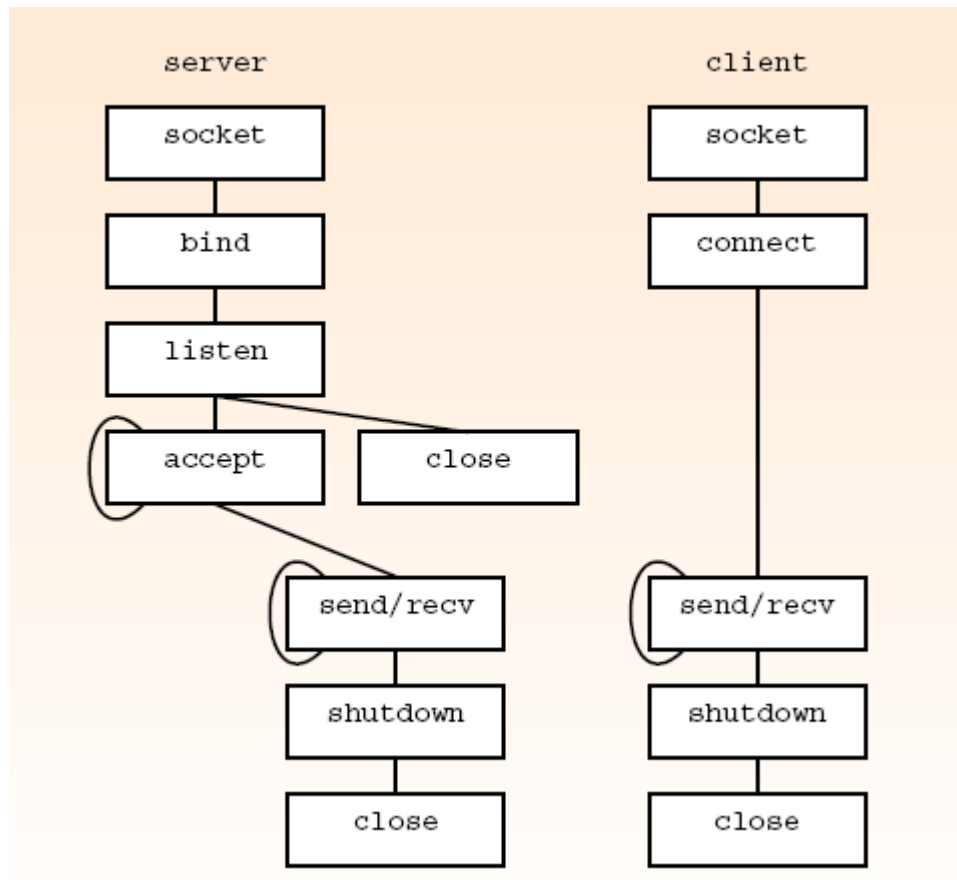
socket API

#include<sys/types.h>

#include<sys /socket.h>

int socket(int domain, int type, int protocol) ;

Returns a file descriptor (called a socket ID) if successful, -1 otherwise. Note that the socket

returns a socket descriptor which is the same as a file descriptor.

The *domain* is AF_INET.

The *type* argument can be:

> ➢ SOCK_STREAM: Establishes a virtual circuit forstream

> ➢ SOCK_DGRAM: Establishes a datagram forcommunication

> ➢ SOCK_SEQPACKET: Establishes a reliable, connection based, two way communication
> with maximum message size. (This is not available on mostmachines.)

protocol is usually zero, so that type defines the connection within domain.

bind

#include <sys / types.h>

#include<sys / socket.h>

int bind(int sid, struct sockaddr *addrPtr, int len)

Where

> ➢ *sid*: is the socketid
> ➢ *addrPtr*: is a pointer to the address family dependent addressstructure
> ➢ *len*: is the size of*addrPtr

Associates a socket id with an address to which other processes can connect. In internet protocol the address is [ipNumber, portNumber]

sockaddr

For the internet family:

struct sockaddr_in {

   sa_family_t   sin_family;   // = AF INET

  in_port_t   sin_port;   // is a port number

  structin_addr sin_addr;   // an IPaddress

}

listen

#include <sys / types.h>

#include <sys / socket.h>

int listen (int sid, int size) ;

Where size it the number of pending connection requests allowed (typically limited by Unix kernels to 5). Returns the 0 on success, or -1 if failure.

accept

#include <sys / types.h>

#include <sys / socket.h>

int accept(int sid ,struct sockaddr *addrPtr , int *lenPtr)

Returns the socketId and address of client connecting to socket.

if lenPtr or addrPtr equal zero, no address structure is returned.

lenPtr is the maximum size of address structure that can be called, returns the actual value.

Waits for an incoming request, and when received creates a socket for it.

send

#include <sys / types.h>

#include <sys / socket.h>

int send(int sid ,const char *bufferPtr ,int len ,intflag)

Send a message. Returns the number of bytes sent or -1 if failure.

(Must be a bound socket).

flag is either

- ➢ 0:default
- ➢ MSG OOB: Out-of-band high prioritycommunication

recv

#include <sys / types.h>

#include <sys / socket.h>

int recv (int sid , char *bufferPtr , int len , int flags)

Receive up to len bytes in bufferPtr. Returns the number of bytes received or -1 on failure.

flags can be either

- ➢ 0:default
- ➢ MSG OOB: out-of-boundmessage
- ➢ MSG PEEK: look at message withoutremoving

Shutdown

#include <sys / types.h>

#include <sys / socket.h>

int shutdown (int sid , int how)

Disables sending (how=1 or how=2) or receiving (how=0 or how=2). Returns -1 on failure.

Connect

          -this is the first of the client calls

#include <sys / types.h>

#include <sys / socket.h>

int connect (int sid , struct sockaddr *addrPtr , int len)

Specifies the destination to form a connection with (addrPtr), and returns a 0 if successful, -1 otherwise.

Port usage

Note that the initiator of communications needs a fixed port to target communications.

This means that some ports must be reserved for these ‒well known‖ ports.

Port usage:

➢ 0-1023: These ports can only be binded to byroot

➢ 1024-5000: well knownports

➢ 5001-64K-1: ephemeralports

Source Code:

Client Side:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#define SERV_TCP_PORT 6880
#define SERV_HOST_ADDR "127.0.0.1"

int main()
{       int sockfd;
        struct sockaddr_in serv_addr,cli_addr;
        char filename[100],buf[1000];
        int n;
        serv_addr.sin_family=AF_INET;
        serv_addr.sin_addr.s_addr=inet_addr(SERV_HOST_ADDR);
        serv_addr.sin_port=htons(SERV_TCP_PORT);

        if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
                printf("Client:cant open stream socket\n");
        else
```

```
        printf("Client:stream socket opened successfully\n");

        if(connect(sockfd,(struct sockaddr *)&serv_addr,

        sizeof(serv_addr))<0)
        printf("Client:cant connect toserver\n");
        else
        printf("Client:connected to server successfully\n");
        printf("\n Enter the file name to be displayed :");
        scanf("%s",filename);
        write(sockfd,filename,strlen(filename));
        printf("\n filename transfered to server\n");
        n=read(sockfd,buf,1000);
        if(n < 0)
        printf("\n error reading from socket");
        printf("\n Client : Displaying file content of %s\n",filename);
        fputs(buf,stdout);
        close(sockfd);
        exit(0);
}
```

Output:
AT CLIENT SIDE
[root@localhost]# cc tcpc.c
[root@localhost]# ./a.out
Data Sent
File Content....

Sockets are a mechanism for exchanging data between processes. These processes can either be on the same machine, or on different machines connected via a network. Once a socket connection is established, data can be sent in both directions until one of the endpoints closes the connection.

I needed to use sockets for a project I was working on, so I developed and refined a few C++ classes to encapsulate the raw socket API calls. Generally, the application requesting the data is called the client, and the application servicing the request is called the server. I created two primary classes, ClientSocket and ServerSocket, that the client and server could use to exchange data.

SERVER SIDE:

```c
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#define SERV_TCP_PORT 6880
#define SERV_HOST_ADDR "127.0.0.1"

int main()
{       intsockfd,newsockfd,clilen;
        struct sockaddr_in cli_addr,serv_addr;
        char filename[25],buf[1000];
        int n,m=0;
        int fd;

        if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
                printf("server:cant open stream socket\n");
        else
        printf("server:stream socket opened successfully\n");

        serv_addr.sin_family=AF_INET;
        serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
        serv_addr.sin_port=htons(SERV_TCP_PORT);

        if((bind(sockfd,(struct sockaddr *)
        &serv_addr,sizeof(serv_addr)))<0)
        printf("server:cant bind local address\n");
        else
        printf("server:bound to local address\n");
        listen(sockfd,5);
        printf("\n SERVER : Waiting for client...\n");
        for(;;)
        {
                clilen=sizeof(cli_addr);
                newsockfd=accept(sockfd,(struct sockaddr *)
        &cli_addr,&clilen);
```

```
        if(newsockfd<0)
        printf("server:accept error\n");
        else
        printf("server:accepted\n");
        n=read(newsockfd,filename,25);
        filename[n]='\0';
        printf("\n SERVER : %s is
        found and ready to transfer
        \n",filename);
        fd=open(filename,O_RDONLY);
        n=read(fd,buf,1000);
        buf[n]='\0';
        write(newsockfd,buf,n);
        printf("\n transfer success\n");
        close(newsockfd);
        exit(0)
}
}
```

Output: [root@localhost
]# cc tcps.c [root@localhost]#
./a.out Received the file name :
data.txt File contentsent

ExperimentNo:9

Demonstrate Deauthentication Attack using Kali Linux

Use following commands:

1) Use methods to list out all of the Access Points.
2) Use appropriate wifi bands parameters to list out all access points under 2.5 and 5ghz frequency
3) Use targeted sniffing to select particular access point and devices/node connected to that access points
4) Use **aireplay-ng to disauthenticate device form network**

ExperimentNo:10

Demonstrate Detection of ARP poisioning using Kali linux
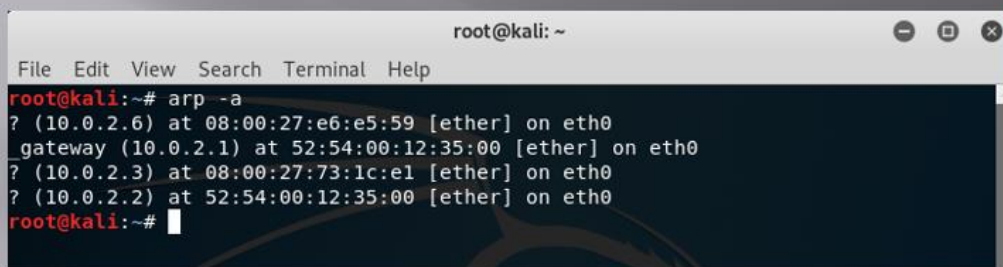
Following points to consider for this Program.

1) Need to have windows virtual machine, which can be downloaded from below link
   https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/
2) Install windows virtual machine in virtual box

   https://www.youtube.com/watch?v=kF-8JijHMEU
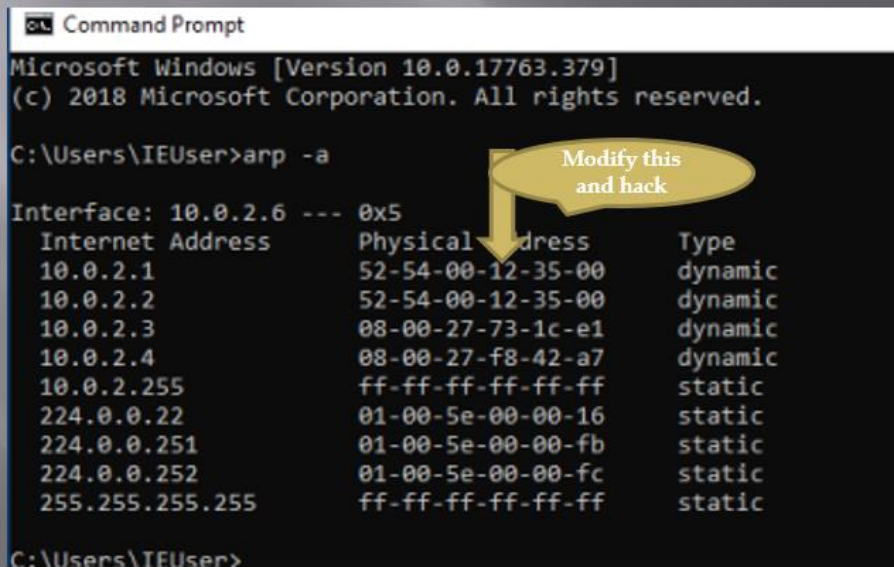
Procedure:

1) **Command to know the ARP table:**
   In terminal of windows as well as kali type **arp-a** which lists ARP table in windows machine as well as kali machine.

2) Use **arpspoof command** 2 times in 2 different terminals in kali machine to modify ARP table in windows and router
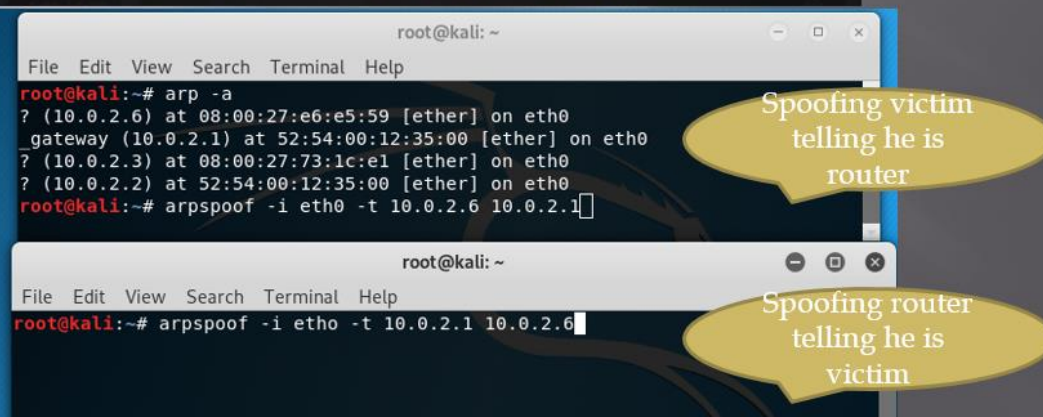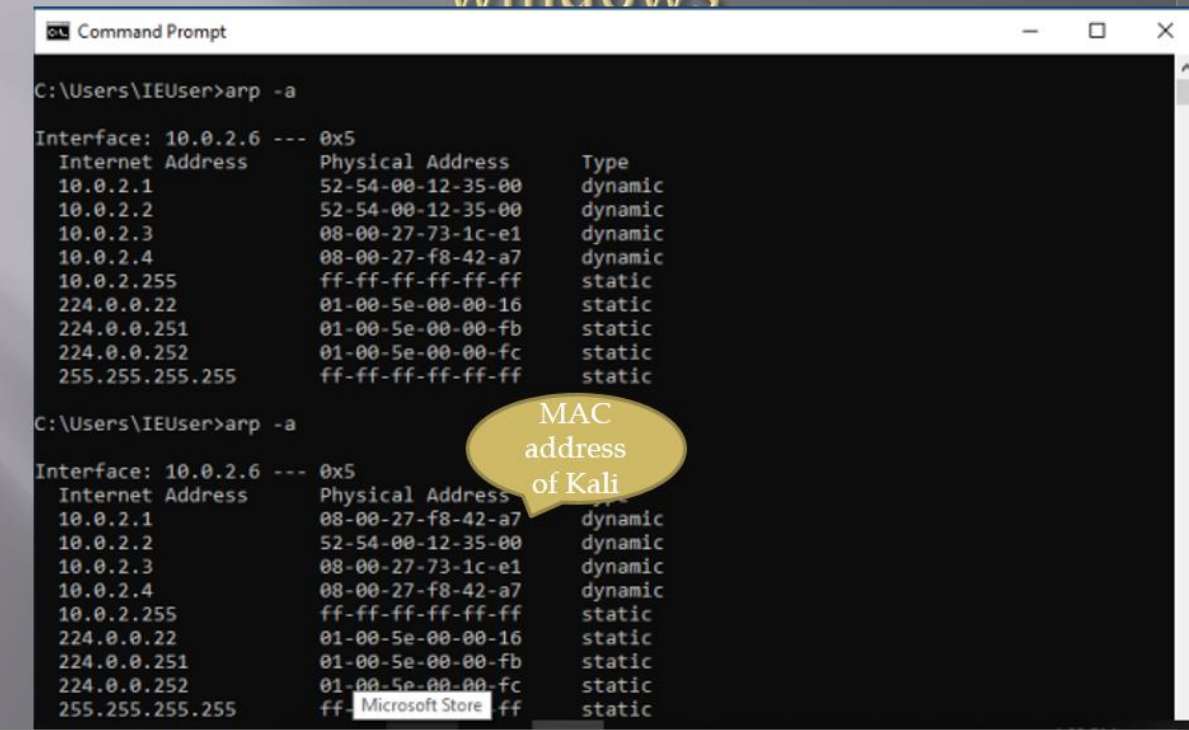
Procedure to detect ARP Poisoning in widows/linux

1) Download XARP tool and install like normal windows installation
   http://www.xarp.net/#download
2) Run arp-a to check routing table note the original routing table entries
3) Perform ARP Poisoning attack
4) Automatically new changes (poisoned entries) in routing table will be monitored by Xarp tool and it will notify.
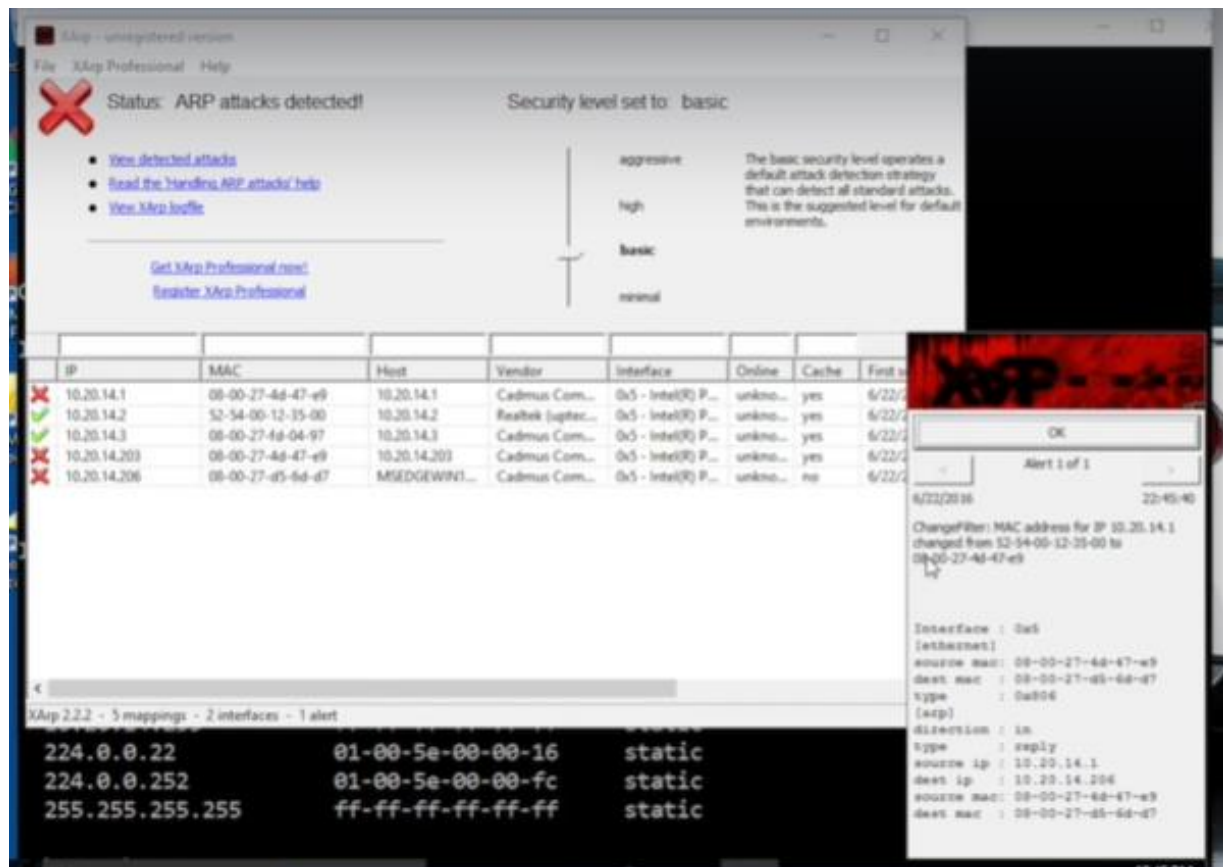
ExperimentNo:11
Demonstrate Prevention of man in the middle attack using Kali Linux

Procedure:

The IP address of the client machine used over LAN for this demo is: 192.168.1.44

And the Attacker IP is: 192.168.1.1

- Open terminal and ping the target machine to verify the IP address you are using and to add it to your arp table
- Type arp in the terminal command line to see your arp table
- For security purposes, IP forwarding is by default disabled in modern Linux systems. For temporarily enabling it, type : echo 1 > /proc/sys/net/ipv4/ip_forward
- For ARP poisoning, the command syntax is: arpspoof -i interface -t target -r host
- Example: arpspoof -i eth0 -t 192.168.1.44 -r 192.168.1.1

  A basic setup is complete and victim network traffic will now pass through the attacker machine. To listen to these packets, we will use Wireshark

- Open up a new terminal and type wireshark. Go to the interface which is capturing all the data flow (here eth0) and start the capture.
- Filter out packets according to what you are looking for. For the purpose of this demo, the user is logging in to a vulnerable website DVWA which uses HTTP instead of the secure version HTTPS. Filter protocol as http and search for required data.
- Right click on the packet and follow TCP stream to open up the data contained within. We can clearly obtain the login credentials of the user, that is the username and password.

ExperimentNo:12

Write a program to illustrate buffer overflow attack.

```c
#include <stdio.h>
#include <string.h>

int main(void)
{
 char buff[15];
 int pass = 0;

printf("\n Enter the password : \n");
 gets(buff);

if(strcmp(buff, "thegeekstuff"))
{
 printf ("\n Wrong Password \n");
 }
 else
{
 printf ("\n Correct Password \n");
 pass = 1;
}
 if(pass)
{
/* Now Give root or admin rights to user*/
 printf ("\n Root privileges given to the user \n");
}

return 0;
}
```

The program above simulates scenario where a program expects a password from user and if the password is correct then it grants root privileges to the user.

Let's the run the program with correct password ie 'thegeekstuff' :

OUTPUT

RUN1

 Enter the password :

thegeekstuff

Correct Password

Root privileges given to the user

This works as expected. The passwords match and root privileges are given.But do you know that there is a possibility of buffer overflow in this program. The gets() function does not check the array bounds and can

even write string of length greater than the size of the buffer to which the string is written. Now, can you even imagine what can an attacker do with this kind of a loophole?

Here is an example :

RUN 2
Enter the password :

hhhhhhhhhhhhhhhhhhhh

Wrong Password

Root privileges given to the user

VIVA QUESTIONS

1. What are functions of differentlayers?

2. Differentiate between TCP/IP Layers and OSILayers

3. Why header isrequired?

4. What is the use of adding header and trailer toframes?

5. What isencapsulation?

6. Why fragmentationrequires?

7. What isMTU?

8. Which layer imposesMTU?

9. Differentiate between flow control and congestioncontrol.

10. Differentiate between Point-to-Point Connection and End-to-Endconnections.

11. What are protocols running in differentlayers?

12. What is ProtocolStack?

13. Differentiate between TCP andUDP.

14. Differentiate between Connectionless and connection orientedconnection.

15. Why frame sorting isrequired?

16. What is meant bysubnet?

17. What is meant byGateway?

18. What is an IP address?

19. What is MACaddress?

20. Why IP address is required when we have MACaddress?

21. What is meant byport?

22. What are ephemerical port number and well known portnumbers?

23. What is asocket?

24. What are the parameters ofsocket()?

25. Describe bind(), listen(), accept(),connect(), send() andrecv().

26. What are system calls? Mention few ofthem.

27. What is IPC? Name threetechniques.

28. Explain mkfifo(), open(), close() with parameters.

29. What is meant by filedescriptor?

30. What is meant by trafficshaping?

31. How do you classify congestion controlalgorithms?

32. Differentiate between Leaky bucket and Tokenbucket.

33. How do you implement Leakybucket?

34. How do you generate bustytraffic?

35. What is the polynomial used inCRC-CCITT?

36. What are the other error detectionalgorithms?

37. What is difference between CRC and Hammingcode?

38. Why Hamming code is called 7,4code?

39. What is odd parity and evenparity?

40. What is meant bysyndrome?

41. What is generatormatrix?

42. What is spanningtree?

43. Differentiate between Prim's and Kruskal'salgorithm.

44. What are Routingalgorithms?

45. How do you classify routing algorithms? Give examples foreach.

46. What are drawbacks in distance vectoralgorithm?

47. How routers update distances to each of itsneighbor?

48. How do you overcome count to infinityproblem?

49. What iscryptography?

50. How do you classify cryptographicalgorithms?

51. What is publickey?

52. What is privatekey?

53. What are key, ciphertext andplaintext?

54. What issimulation?

55. What are advantages ofsimulation?

56. Differentiate between Simulation andEmulation.

57. What is meant byrouter?

58. What is meant bybridge?

59. What is meant byswitch?

60. What is meant byhub?

61. Differentiate between route, bridge, switch andhub.

62. What is ping andtelnet?

63. What isFTP?

64. What isBER?

65. What is meant by congestionwindow?

66. What isESS?

67. What is incoming throughput and outgoingthroughput?

68. What iscollision?

69. How do you generate multiple traffics across different sender-receiverpairs?

70. How do you setup EthernetLAN?

71. What is meant by mobilehost?

72. What is meant byNS2?

73. What are nam and tr files?

74. Name few other Network simulators

75. Differentiate between logical and physicaladdress.

76. Which address gets affected if a system moves from one place to anotherplace?

77. What is ICMP? What are uses of ICMP? Namefew.

78. Which layer implements security fordata?