Mahatma Gandhi Mission’s

**MGM’s College of Engineering and Technology (MGMCET), Navi Mumbai**

***Department of Computer Engineering***

**Lab Manual**

**Final year Year Semester-VIII**

**Sub:** Adhoc Wireless Networks

**Even Semester**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| 1. | List of Experiments | 8 |
| 2. | Course objective, Course outcome &Experiment Plan | 9,10 |
| 3. | CO-PO Mapping | 11 |
| 4. | Study and Evaluation Scheme | 12 |
| 5. | Experiment No. 1 | 13 |
| 6. | Experiment No. 2 | 16 |
| 7. | Experiment No. 3 | 19 |
| 8. | Experiment No. 4 | 22 |
| 9. | Experiment No. 5 | 25 |
| 10. | Experiment No. 6 | 29 |
| 11. | Experiment No. 7 | 34 |
| 12. | Experiment No. 8 | 39 |
| 13. | Experiment No. 9 | 43 |
| 14. | Experiment No. 10 | 46 |
| 15. | Experiment No. 11 | 49 |

List of Experiments

|  |  |
| --- | --- |
| **Sr. No.** | **Experiments Name** |
| 1 | Installation of NS2 & NS3 in Linux. |
| 2 | Basics of Network simulation & a simple wireless network simulation |
| 3 | Measuring Network performance. |
| 4 | Simulating Wi-Fi network. |
| 5 | Implementation wireless network in NS-3. |
| 6 | Implementing IEEE 802.11 wireless LAN in Ad-Hoc Mode using NS2. |
| 7 | Implementation of Bluetooth network for data transfer from one node to other node in NS-2. |
| 8 | Develop a simple wireless adhoc network using aodv, aomdv & tora. Also compare the performance using NS-2. |
| 9 | Explore and use security tools like WEP, WPA and WPA2. Compare the Performance. |
| 10 | Simulation of Sumo on Ubuntu. |
| 11 | Simulation of Veins on Ubuntu. |
| 12 | Experiment No. 12-Mini Project |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Objective, Course Outcome & Experiment Plan  **Course Objective:**   |  |  | | --- | --- | | 1 | To Identify the major issues associated with ad-hoc networks. | | 2 | To identify the requirements for protocols for wireless ad-hoc networks as compared to the protocols existing for wired network. | | 3 | To explore current ad-hoc technologies by researching key areas such as algorithms, protocols, hardware, and applications. To introduce various techniques for representation of the data in the real world. | | 4 | To Provide hands-on experience through real-world programming projects | | 5 | To provide advanced in–depth networking materials to graduate students in networking research. | |
| |  |  | | --- | --- | | **Lab Outcomes:** | | | CO1 | | Identify the characteristics and features of Adhoc Networks. | | | CO2 | | Understand the concepts & be able to design MAC protocols for Ad Hoc networks. | | | CO3 | | Implement protocols / Carry out simulation of routing protocols of Adhoc Networks. | | | CO4 | | Interpret the flow control in transport layer of Ad Hoc Networks. | | | CO5 | | Analyze security principles for routing of Ad Hoc Networks. | | | CO6 | | Utilize the concepts of Adhoc Networks in VANETs. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module**  **No.** | **Week**  **No.** | **Experiments Name** | **Course**  **Outcome** | **Weightage** |
| 1 | W1 | Installation of NS2 & NS3 in Linux. | CO1 | 1 |
| 2 | W2 | Basics of Network simulation & a simple wireless network simulation | CO1 | 6 |
| 3 | W3 | Measuring Network performance. | CO1 | 4 |
| 4 | W4 | Simulating Wi-Fi network. | CO2 | 1 |
| 5 | W5 | Implementation wireless network in NS-3. | CO3 | 2 |
| 6 | W6 | Implementing IEEE 802.11 wireless LAN in Ad-Hoc Mode using NS2. | CO2 | 2 |
| 7 | W6 | Implementation of Bluetooth network for data transfer from one node to other node in NS-2. | CO2 | 2 |
| 8 | W7 | Develop a simple wireless adhoc network using aodv, aomdv&tora. Also compare the performance using NS-2. | CO3 | 2 |
| 9 | W8 | Explore and use security tools like WEP, WPA and WPA2. Compare the Performance. | CO5 | 10 |
| 10 | W9 | Simulation of Sumo on Ubuntu. | CO6 | 10 |
| 11 | W10 | Simulation of Veins on Ubuntu | CO6 | 10 |
| 12 | W11 | Mini Project | CO4 | 10 |

Mapping Course Outcome – Program Outcome

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Subject Weight** | **Course Outcomes** | | **Contribution to Program outcomes PO’S** | | | | | | | | | | | |
|  |  | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **THEORY 20** | CO1 | Identify the characteristics and features of Adhoc Networks. |  | 2 | 2 | 1 | 2 |  |  |  | 1 |  | 1 | 1 |
| CO2 | Understand the concepts & be able to design MAC protocols for Ad Hoc networks. | 1 | 2 | 2 | 1 | 1 |  |  |  | 1 |  | 1 | 1 |
| **CO3** | Implement protocols / Carry out simulation of routing protocols of Adhoc Networks. | 1 | 2 | 2 | 1 | 1 |  |  |  | 1 |  | 1 | 1 |
| **CO4** | Interpret the flow control in transport layer of Ad Hoc Networks. | 1 | 2 | 1 | 1 | 1 |  |  |  | 1 |  | 1 | 1 |
| **CO5** | Analyze security principles for routing of Ad Hoc Networks. | 1 | 1 | 2 | 1 | 1 | 1 |  |  | 1 |  | 2 | 1 |
| **CO6** | Utilize the concepts of Adhoc Networks in VANETs. | 1 | 2 | 2 | 2 | 1 |  |  |  | 1 |  |  | 1 |
| **Subject Weight** | **Course Outcomes** | | **Contribution to Program outcomes PO’S** | | | | | | | | | | | |
|  |  | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **PRATICAL**  **80** | CO1 | Explore the knowledge of NS2 and NS3 by installing it and make it ready |  | 2 | 2 | 1 | 2 |  |  |  | 1 |  | 1 | 1 |
| CO2 | Synthesize a simulation and evaluate the performance of WLAN 802.11 and Bluetooth | 1 | 2 | 2 | 1 | 1 |  |  |  | 1 |  | 1 | 1 |
| **CO3** | Analyze and implement MAC & Network layer protocols using open source and synthesis as well as evaluate its performance | 1 | 2 | 2 | 1 | 1 |  |  |  | 1 |  | 1 | 1 |
| **CO4** | Implement Transport layer protocols / Carry out simulation of routing protocols of Adhoc Networks | 1 | 2 | 2 | 1 | 1 |  |  |  | 1 |  | 1 | 1 |
| **CO5** | Describe and interpret the use security routines and evaluate its performance | 1 | 1 | 2 | 1 | 1 | 1 |  |  | 1 |  | 2 | 1 |
| **CO6** | Explore and understand the capability of SUMO and MOVE as well as Nessi by installing it and analyze it by applying on various scenarios | 1 | 2 | 2 | 2 | 1 |  |  |  | 1 |  |  | 1 |

Study and Evaluation Scheme

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Course Code | Course Name | Examination Scheme | | |
| DLO8013 | Adhoc Wireless Networks Lab | Term Work | Practical & Oral | Total |
| 50 | 25 | 75 |

**Term Work:**

1. A case study should be conducted using a Mini Project by taking a good problem definition and complete the following phases. a. Decomposing the problem into modules b. Identifying the best suited data structure for solving the sub problems with justification c. Define algorithms for various identified functions d. Implement the modules
2. The final certification and acceptance of term work ensures that satisfactory performance of laboratory work and minimum passing marks in term work.
3. Term Work: Total 50 Marks.

**Practical/Experiments:**

1. Term work should consist of at least 10 experiments.
2. Journal must include at least 2 assignments.
3. The final certification and acceptance of term work ensures that satisfactory performance of laboratory work and minimum passing marks in term work.
4. Practical exam will be based on the above syllabus.

**Experiment No. : 1**

**Installation of NS2 & NS3 in Linux.**

Experiment No. 1

1. **Aim:** Installation of NS2&NS3
2. **Objectives:**

* To provide students overview of data communication in network
* To provide experience in installation of network simulation software
* To provide experience in designing and managing a communication protocol.

1. **Outcomes:**

* Understand, design and analyse problem in installation of the software
* Install and Configure open source tool NS2
* Exploit gained skills and knowledge
* Engage in higher studies and lifelong studies.

1. **Hardware / Software Required :**NS2, NS3 Ubuntu
2. **Theory:**

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool

that has proved useful in studying the dynamic nature of communication networks. Simulation of

wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP)

can be done using NS2. In general, NS2 provides users with a way of specifying such network

protocols and simulating their corresponding behaviours.

Download NS2 from terminal using command

wget http://downloads.sourceforge.net/project/nsnam/allinone/ns-allinone-2.35/ns-allinone-

2.35.tar.gz

or directly using

http://downloads.sourceforge.net/project/nsnam/allinone/ns-allinone-2.35/ns-allinone-

2.35.tar.gz

•Unpack ns-allinone-2.35.tar.gz to your home directory.

tar -zxvf ns-allinone-2.35.tar.gz -C /home/yourusername

•Next you need to edit a file. Go to /home/yourusername/ns-allinone-2.35/ns-2.35/linkstate/

directory and open ls.h file in a text editor. Line 137 will be

void eraseAll() { erase(baseMap::begin(), baseMap::end()); }

Change it to

void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }

•Now install dependencies.

sudo apt-get install tcl8.5-dev tk8.5-dev

sudo apt-get install build-essential autoconfautomake

sudo apt-get install perlxgraphlibxt-dev libx11-dev libxmu-dev g++ xorg-dev

sudo apt-get install libperl4-corelibs-perl

•Change your directory

cd /home/yourusername/ns-allinone-2.35/

•Run the “install script” by. /install command. [This may take few minutes]

•After installation modify .bashrc file located in your home directory.

gedit /home/yourusername/.bashrc

Add the following lines to the end of the file.

Export PATH=$PATH:/home/yourusername/ns-allinone2.35/bin:/home/yourusername/nsallinone-2.35/tcl8.5.10/unix:/home/yourusername/ns-allinone-2.35/tk8.5.10/unix

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/home/yourusername/ns-allinone-

2.35/otcl-1.14:/home/yourusername/ns-allinone-2.35/lib

export TCL\_LIBRARY=$TCL\_LIBRARY:/home/yourusername/ns-allinone-

2.35/tcl8.5.10/library

Close gedit.

•To enable the path setting immediately, use.

source ~/.bashrc

•Go to directory /home/yourusername/ns-allinone-2.35/ns-2.35

cd /home/yourusername/ns-allinone-2.35/ns-2.35

and type: ./validate

Validations tests are performed. This may also take few minutes.

•Now check ns2 working by entering the command ns. We should get a “%” prompt.

•For testing create a sample.tcl file in your home directory.

geditsample.tcl

and type the following sample code

set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec namout.nam&

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns at 3.0 “finish”

$ns run

•Save and close sample.tcl and run using

ns sample.tcl

It will display a graph with two nodes and a link between them in the NAM

GUI window

NS3

OS to be used: - Ubuntu or Linux Mint (LTS Version

Support)

• Install the Required Libraries

• sudo apt-get update

• sudo apt-install build-essential autoconf

automakelibxmu-dev ia32-libs NetAnim

wiresharkpygraphviz qt4-default p7zip-full bzr

cmake mercurial git cvs

• Download ns-allinone-3.28.1.tar.bz2 from here

• Create a folder “NS3” inside home folder

• Copy ns-allinone-3.28.1.tar.bz2 in “/home/NS3”

folder

• Execute these commands from the “/home/NS3”

folder

• tar jxvf ns-allinone-3.28.1.tar.bz2

• cd ns-allinone-3.28.1/

• Execute this command for building ns3

• ./build.py –enable-examples –enable-tests

• It will take some time for setting up environment

depending on the Memory and CPU.

• To check if build completed successfully

cd ns-allinone-3.28.1/ns-3.28.1/

• ./waf --run hello-simulator

• (This will print “Hello Simulator” in the

terminal ,which indicates build completed

successfully)

Steps to run programs in ns3:-

❖Your Programs can be executed within a folder

called ~ ns-3.28.1/scratch/

❖You have to write your programs in C or Python

within scratch folder .

❖You can run the examples using

• cd ns-allinone-3.28.1/ns-3.28.1/

• ./waf --run scratch/file (No extension

for .cc files)

• ./waf --pyrun scratch/file.py

• Steps For Building NetAnim:-

• cd ..

• cd netanim-3.108

• make clean

• qmake NetAnim.pro• make

make

• Your NetAnim is ready to use

• ./NetAnim

• “./NetAnim” command will open the Program, select

xml file and run it.

1. **Conclusion:** Thus we will install and configure open source tool NS-2 and It is aligned with the simulation needs of modern networking research. It encourages community contribution, peer review, and validation of the software
2. **Viva Questions:**

* What platforms does NS-2 run on and what kind of hardware do I need?
* What protocols does NS-2 support?
* What is the Full form on NS-2??

1. **References:**

* Ekram Hossain and TeerawatIssariyakul, ―Introduction to Network Simulator NS-2, SpringerSecond Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No. : 2**

**Basics of Network simulation & Configure a simple wireless network**

Experiment No. 2

1. **Aim:** learn how to use NS-2, to get acquainted with the simulated objects and understand the operations of network simulation
2. **Objectives:**

* To know the fundamental tools of NS2

1. **Outcomes:**

* Learn the basic concepts about open source network simulator NS-2 and work with NS-2
* Defining the different agents and their applications like TCP, FTP over TCP, UDP, CBR over UDP
* Identifying and solving typical errors encountered during installation of NS-2

1. **Hardware / Software Required :** Virtual lab, NS2, Ubuntu
2. **Theory:**

Network Simulator is based on two languages: C++ and OTcl. OTcl is the object oriented version of Tool Command Language. While the core of NS-2 is written in C++, one uses OTcl to write simulation scripts. C++ helps in the following way:

It helps to increase the efficiency of simulation.

Its is used to provide details of the protocols and their operation.

It is used to reduce packet and event processing time.

OTcl helps in the following way:

With the help of OTcl we can describe different network topologies

It helps us to specify the protocols and their applications

It allows fast development

Tcl is compatible with many platforms and it is flexible for integration

Tcl is very easy to use and it is available in free

And of course, there is a linkage between C++ and OTcl, which allows us to run the simulation scripts.

**Basics of Tcl Programming for NS-2**

Network simulation with NS-2 would involve the following general steps:

1. Initialization and termination aspects of network simulator object
2. Defining the network topology: nodes, links, queues, mobility of nodes, if any
3. Defining the network traffic: creating agents and their applications
4. Setting trace for Network Animator (NAM) [optional]
5. Tracing

In this section, we provide a brief overview of the most commonly used features of NS-2. This summary has been prepared based on various tutorials on, and the manual for, NS-2. See the References section for some of the different tutorials available.

**Initialization**

To create a new simulator we write

set ns [new Simulator]

From the above command we get that a variable ns is being initialized by using the set command. Here the code [new Simulator] is a instantiation of the class Simulator which uses the reserved word new. So we can call all the methods present inside the class simulator by using the variable 'ns'.

Creating the output files

# Create the trace files

set tracefile [open out.tr w]

$ns trace-all $tracefile

# Create the nam files

set namfile [open out.nam w]

$ns namtrace-all $namfile

In the above we create a output trace file 'out.tr' and a NAM visualization file 'out.nam'. But in the Tcl script they are not called by their names declared, while they are called by the pointers initialized for them such as 'tracefile' and 'namfile' respectively.The line which starts with # are commented. The next line opens the file 'out.tr' which is used for writing is declared 'w'. The next line uses a simulator method trace-all by which we will trace all the events in a particular format.

The termination program is done by using a 'finish' procedure

# Defining the 'finish' procedure'

proc finish {} {

global ns tracefilenamfile

$ns flush-trace

close $tracefile

close $namfile

exit 0

}

In the above, the keyword proc is used to declare a procedure called 'finish'. The keyword global is used to tell what variables are being used outside the procedure.

flush-trace is a simulator method that dumps the traces on the respective files. The command close is used to close the trace files and the command exec is used to execute the NAM visualization. The command exit closes the application and returns zero as default for clean exit.

In ns we end the program by calling the 'finish' procedure

# End the program

$ns at 125.0 "finish"

Thus the entire operation ends at 125 seconds.To begin the simulation we will use the command

# Start the the simulation process

$ns run

**Defining nodes, links, queues (topology)**

Way to create a node:

set n0 [$ns node]

In the above we created a node that is pointed by a variable n0. While referring the node in the script we use $n0. Similarly we create another node n2. Now we will set a link between the two nodes.

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

So we are creating a bi-directional link between nodes n0 and n2 with a capacity of 10 Mb/sec and a propagation delay of 10 ms.

In NS an output queue of a node is implemented as a part of a link whose input is that node to handle the overflow at the queue. If the buffer capacity of the output queue is exceeded then the last packet arrived is dropped and here we will use a 'DropTail' option. There are other queue types such as RED (Random Early Discard) mechanism, FQ (Fair Queuing), DRR (Deficit Round Robin), SFQ (Stochastic Fair Queuing) also available.

Now we will define the buffer capacity of the queue related to the above link

# Set queue size of the link

$ns queue-limit $n0 $n2 20

So, if we summarize the above three things we get

# Create 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]

# Create links between the nodes

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

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

$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail

$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail

$ns duplex-link $n0 $n2 0.5Mb 40ms DropTail

$ns duplex-link $n0 $n2 0.5Mb 40ms DropTail

# Set queue-size of the link (n2-n3) to 20

$ns queue-limit $n2 $n3 20

Agents and applications

TCP

TCP is used to provide reliable transport of packets from one host to another host by sending acknowledgements on proper transfer or loss of packets. Thus, TCP requires bi-directional links in order for acknowledgements to return to the source.

Now we will show how to set up tcp connection between two nodes

# Setting a TCP connection

set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n4 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

$tcp set packetSize\_ 552

The command set tcp [new Agent/TCP] gives a pointer called 'tcp' to the TCP agent object of ns. The command $ns attach-agent $n0 $tcp defines the source node of TCP connection. Next the command set sink [new Agent/TCPSink] defines the destination of TCP by a pointer called 'sink'. The next command $ns attach-agent $n4 $sink defines the destination node as n4. Next, the command $ns connect $tcp $sink makes the TCP connection between the source and the destination i.e n0 and n4. When we have several flows (such as TCP, UDP) in a network, to identify these flows we set their flow ID by using the command $tcp set fid\_1. In the last line we set the packet size of TCP as 552 byte. The default packet size of TCP is 1000 B.

FTP over TCP

File Transfer Protocol (FTP) is a standard mechanism provided by the Internet for transferring files from one host to another. FTP differs from other client server applications in that it establishes two connections between the client and the server. One connection is used for data transfer and other one is used for providing control information. FTP uses the services of the TCP. The well Known port 21 is used for control connections and the other port 20 is used for data transfer.

Here we will learn in how to run a FTP connection over a TCP:

# Initiating FTP over TCP

Set ftp [new Application/FTP]

$ftp attach-agent $tcp

In above,the command set ftp [new Application/FTP] gives a pointer called 'ftp' which indicates the FTP application. Next, we attach the ftp application with tcp agent as FTP uses the services of TCP.

UDP

The User datagram Protocol is one of the main protocols of the Internet protocol suite. UDP helps the host to send send messages in the form of datagrams to another host which is present in a Internet protocol network without any kind of requirement for channel transmission setup. UDP provides a unreliable service and the datagrams may arrive out of order, appear duplicated, or go missing without notice. UDP assumes that error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

Now we will learn how to create a UDP connection in network simulator.

# 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

The command set udp [new Agent/UDP] gives a pointer called 'udp' which indicates the udp agent which is a object of ns. Then the command $ns attach-agent $n1 $udp defines the source node of UDP connection. Next the command set null [new Agent/Null] defines the destination of udp by a pointer called null. The next command $ns attach-agent $n5 $null defines the destination node as n5. Next, the command $ns connect $udp $null makes the UDP connection between the source and the destination i.e n1 and n5. To identify a particular flow we mark it using the command $udp set fid\_2.

Constant Bit Rate (CBR)

Constant Bit Rate (CBR) is a term used in telecommunications, relating to the quality of service.When referring to codecs, constant bit rate encoding means that the rate at which a codec's output data should be consumed is constant. CBR is useful for streaming multimedia content on limited capacity channels since it is the maximum bit rate that matters, not the average, so CBR would be used to take advantage of all of the capacity. CBR would not be the optimal choice for storage as it would not allocate enough data for complex sections (resulting in degraded quality) while wasting data on simple sections.

CBR over UDP Connection

# Setup CBR over UDP

et cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetSize\_ 1000

$cbr set rate\_ 0.01Mb

$cbr set random\_ false

In the above we define a CBR connection over a UDP one. Well we have already defined the UDP source and UDP agent as same as TCP. Instead of defining the rate we define the time interval between the transmission of packets in the command $cbr set rate\_ 0.01Mb. Next, with the help of the command $cbr set random\_ false we can set random noise in cbr traffic. We can keep the noise by setting it to false or we can set the noise on by the command $cbr set random\_ 1. We can set by packet size by using the command $cbr set packetSize\_. The packet size is specified in bytes.

Scheduling Events

In ns the tcl script defines how to schedule the events or in other words at what time which event will occur and stop. This can be done using the command $ns at time event. Here in our program we will schedule when the ftp and cbr traffic should start and stop.

# Scheduling the events

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 124.0 "$ftp stop"

$ns at 124.5 "$cbr stop"

Network Animator (NAM)

When we will run the above program in ns then we can can visualize the network in the NAM. But instead of giving random positions to the nodes, we can give suitable initial positions to the nodes and can form a suitable topology. So, in our program we can give positions to the nodes in NAM in the following way

# Give position to the nodes (for NAM)

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

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

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

$ns simplex-link-op $n3 $n2 orient-left

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

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

We can also define the color of CBR and TCP packets for identification in NAM. For this we use the following command

# Marking the flows (for NAM)

$ns color1 Blue

$ns color2 Red

To view the network animator we need to type the command: nam

Network Animator could only be run on a desktop. This Virtual Lab does not provide any option to visualize the NAM output (apart from a few screenshots). Henceforth, we would skip creating NAM trace files in our code.

1. **Conclusion:** Thus the students will be able to run ns to view the simulation and get the required outputs.
2. **Viva Questions:**

* What is the code to view the simulation of the network in NAM ?
* What is the function of trace-all ?
* At what point of program the 'finish' procedure should be called ?

1. **References:**

* Ekram Hossain and TeerawatIssariyakul, ―Introduction to Network Simulator NS-2, SpringerSecond Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No. : 3**

**Measuring Network performance.**

Experiment No. 3

**1. Aim:** Measuring Network performance.

**2. Objectives:**

* To understand different types of metrics used for measuring the network performance.
* To analyze the performance by different types of curve used in different scenario.

**3. Outcomes:**

* Get familiar with the concept of network performance evaluation, and different related metrics
* Get an overview on bandwidth sharing by multiple traffic flows
* Identify bottlenecks in a network

**4. Hardware / Software Required :** Virtual lab , NS2, Ubuntu

**5. Theory:**

**Network Performance Evaluation**

For a given network, one might be interested to know how well it is performing. One might also wish to know what could be done to further improve the performance, or if the network is giving the peak performance. Thus, one needs to do a comparative study of the network by considering different options. This performance evaluation helps the user to determine the suitable network configuration that serves him best.

For example consider a new startup organization which has setup its own web portal. As the portal gradually becomes popular then network traffic increases which would degrade its performance. Therefore, one should have a well configured network with proper load balancing capabilities.

**Performance Evaluation Metrics**

Before we can proceed with performance evaluation, we must choose the different metrics that would help us in making comparisons. There could be different metrics to determine the performance like throughput, delay, jitter, packet loss. The choice of metric would depend upon the purpose the network has been setup for. The metrics could be related to the different layers of the network stack. For example, TCP throughput is based on the application layer, whereas IP round trip time is based on the network layer. For example, a network supporting multimedia applications should have minimum delay and jitter.Packet loss might not be a critical issue for such network. However, packet loss might be a considerable factor for networks supporting textual data oriented applications, say someone downloading by FTP.

Once the metrics have been chosen, one goes for their quantitative evaluation by subjecting the network under diverse conditions. For example, one could make step by step increments in bandwidth of the links, which in turn improve the throughput. However, the throughput might get saturated beyond the certain point. That is, further increase in bandwidth would not improve throughput. Thus, the optimum value of bandwidth has been determined.

The table below shows different metrics of evaluation, and categories they are appropriate for

| **Category** | **Metric** | **Unit** |
| --- | --- | --- |
| Productivity | Throughput | Bytes per second |
| Responsiveness | Delay, jitter | seconds |
| Utilization | Channel utilization | Percentage of time busy |
| Loss | Packet drops, Retransmission count | Number |
| Buffer space | Queue size, overflow or underflow rate | Bytes |

It might not be always possible or feasible to obtain best performance from a network due to various factors like high cost,complexity, compatibility. In such cases one would like to obtain optimum performance by balancing different factors.

**Following are some of the performance measurement metrics:**

**Latency**: It can take a long time for a packet to be delivered across intervening networks. In reliable protocols where a receiver acknowledges delivery of each chunk of data, it is possible to measure this as round-trip time.

**Packet loss**: In some cases, intermediate devices in a network will lose packets. This may be due to errors, to overloading of the intermediate network, or to intentional discarding of traffic in order to enforce a particular service level.

**Retransmission**: When packets are lost in a reliable network, they are retransmitted. This incurs two delays: First, the delay from re-sending the data; and second, the delay resulting from waiting until the data is received in the correct order before forwarding it up the protocol stack.

**Throughput:** The amount of traffic a network can carry is measured as throughput, usually in terms such as kilobits per second. Throughput is analogous to the number of lanes on a highway, whereas latency is analogous to its speed limit.

**Parameters Affecting the Performance of Networks**

Different parameters can together or independently determine how well a network would perform. A few such are mentioned below:

**Bandwidth**: Its is the maximum data transfer rate which a link allows.It is expressed in bits per second(bps).

**Propagation Delay:** It is the amount of time required to for a packet to travel from one node to another.If the propagation delay is high then throughput will be low i.e they are inversely proportional to each other.

Queue type and queue size: The queue of a node is implemented as a part of a link whose input is that node to handle the overflow at the queue.But if the buffer capacity of the output queue is exceeded then the last packet arrived is dropped.We do set the buffer capacity by using queue size.

**Performance Evaluation Techniques**

Before starting with tuning the performance of a network one must remember that the performance, to some extent, depends on the workload as well as the topology. A given topology might give different throughputs under CBR and exponential traffic. Keeping this in mind, one can go for studying an actual network. Otherwise one can simulate its performance using suitable parameters. these simulations would largely depend on queuing theory.

**Network Performance Evaluation using NS-2**

In this section we discuss how to evaluate performance of a network by simulating it with ns2.

Choose and generate a network topology to be used throughout the simulation. This could be a wired network, in which case the topology remains fixed. However, for a wireless network with mobile nodes the topology would change with time, or randomly.

Once the topology has been generated, traffic source(s) and destination(s) are fixed. Assign suitable traffic sources to the source nodes, and traffic sinks to the destination nodes.

Some of the parameters that can be used for comparative study of performance of the network are: link bandwidth, propagation delay, node queue type. For example: In ns2 we do create a link by using this code:

1

$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail

In this code there could be three parameters namely bandwidth, propagation delay and queue type.

We can vary these parameters and could possibly obtain different throughputs. From there we can determine the conditions that provide higher throughput values.In general, we can alter different parameters and study their effects on one or more performance metrics and thereby filter out the combination of parameters that gives best performance.

Performance of the network can be determined by considering different metrics for example 'Throughput'

We can vary these parameters and could possibly obtain different throughputs, which can be plotted using xgraph

From there we can determine the conditions that provide higher throughput values

Make suitable combinations with the parameters that wil bring some changes in the throughput

Use the best combination of parameters which will bring the best throughput and implement it

We are considering only one performance metric i.e throughput in our experiment.Other metrics like packet loss,latency,retransmission can measured to evaluate the performance of a network in a more accurate way which will help us to setup the network in a proper way.

1. **Conclusion:**
2. **Viva Questions:**

* What is Propagation Delay ?
* What is throughput?

1. **References:**

* Ekram Hossain and TeerawatIssariyakul, ―Introduction to Network Simulator NS-2, SpringerSecond Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No. : 4**

**Simulating Wi-Fi network.**

Experiment No. 4

1. **Aim:** Simulate wifi network mode
2. **Objectives:**

* learn the different standards and the simulation of Wi-Fi network.
* Understand the concept of hidden node and exposed node problem and solve these issues.

1. **Outcomes:**

* Understand about Wi-Fi network, different standards, and related protocols
* Analyze the Wi-Fi communication range in the presence of the access point (AP) and the base station (BS)
* Learn about hidden node and exposed node problems, and their possible solutions

1. **Hardware / Software Required :** Virtual lab, NS 2 , Ubuntu
2. **Theory:**

Wi-Fi Networks

Wi-Fi (Wireless Fidelity) uses the IEEE 802.11 standard. Wi-Fi has some other extensions like 802.11a, 802.11b, and 802.11g. Wi-Fi technology operating at a frequency of 2.4 GHz and uses radio communication.

IEEE 802.11 Standards

Following are the different standards for Wi-Fi

802.11 is the wireless local area networks (WLANs) standard. Supports 1- 2 Mbps.

802.11a is a high speed WLANs standard for 5 GHz band. It uses an orthogonal frequency division multiplexing (OFDM) encoding scheme.

802.11b is a wireless standard for 2.4 GHz band. It supports 11 Mbps. It uses only DSSS (Direct Sequence Spread Spectrum).

802.11d is a international roaming. This automatically configures devices to meet local radio frequency (RF) regulations.

802.11e address the quality of service (QoS) requirements for all IEEE wireless radio interfaces.

802.11f defines inter-access point communications for multiple vendor-distributed WLANs.

802.11g establishes an additional modulation technique for 2.4 GHz band. This supports speeds up to 54 Mbps.

802.11h supports the spectrum management of the 5 GHz band.

802.11i define the current security weaknesses for both encryption and authentication protocols.

802.11n supports more throughput improvements. Also provides speeds up to 500 Mbps.

The basic difference between 802.11a,802.11b and 802.11g are given below:

| **Parameter** | **802.11a** | **802.11b** | **802.11g** |
| --- | --- | --- | --- |
| **Standard approved** | Sept 1999 | Sept 1999 | June 2003 |
| **Available bandwidth** | 300MHz | 83.5MHz | 83.5MHz |
| **No. of overlapping channel** | 4 | 3 | 3 |
| **Frequency** | 5GHz | 2.4GHz | 2.4GHz |
| **Typical Data Rate** | 23 Mbit/s | 4.5 Mbit/s | 19 Mbit/s |
| **Maximum Data Rate** | 54 Mbit/s | 11 Mbit/s | 54 Mbit/s |
| **Range** | 115 feet | 115 feet | 125 feet |
| **Compatibility** | None | None | backward compatible with b |
| **Advantages** | fast maximum speed, regulated frequencies prevent signal interference from other devices | lowest cost, signal range is good and not easily obstructed | fast maximum speed, signal range is good and not easily obstructed |
| **Limitations** | highest cost, shorter range signal that is more easily obstructed | slowest maximum speed, home appliances may interfere on the unregulated frequency band | costs more than 802.11b, appliances may interfere on the unregulated signal frequency |

Hardware Requirements for Wi-Fi

The following hardware devices are required for connecting the Wi-Fi Network.

Access Point

Access Point (AP) acts as a bridge between the wired network and wireless devices. It allows multiple devices to connect through it for accessing the network. An AP can also act as a router through which the data transmission can be possible from one AP to another.

Wireless Network Card

A wireless network card (WNC) is required on each device on a wireless network. A desktop computer would need an internal card, which will usually have a small antenna or an external antenna on it. These antennas are optional on most equipment and they help to increase the signal on the card.

Transmitter

Transmitter is used for emitting the wireless signals and it also receive the connection requests where a wireless client will send the requests and receives the replies from the transmitter. In this case, the transmitter is the wireless router.

How to connect to the Wi-Fi Networks?

Wi-Fi Network is easy to connect. Suppose, we will think about our laptop with any operating systems, then we can easily connect to a Wi-Fi network for accessing or we can share different files on a network.

Once we have acquired the necessary wireless networking hardware then, connect it all together to form a network and allow each device to communicate. The instructions below will act as basic guidelines of what needs to be done.

The distance between each computer should be below 100 meters

Each computer should be on the same floor

Plug the AP into the power outlet and existing Ethernet jack on the network

Configure the access point (usually through a web browser)

Configure the client computers with the appropriate network settings required to be able to communicate with the AP.

Advantages of Wi-Fi

Following are the different benefits of Wi-Fi Networks

In wireless ad-hoc network mode, devices like consumer electronics and gaming applications can directly connect and exchange data with each other.

Digital images can be transferred wirelessly from cameras and other devices.

All connected devices within the range have access to Internet and inter-networking.

Wi-Fi enables wireless voice-applications (Vo WLAN or WVOIP).

Wi-Fi provides a secure computer networking gateway, firewall, DHCP server and an intrusion detection system along with other features.

Cost of cabling and network deployment of Local Area Networks is significantly reduced.

Can be used at placed where wiring and cable lay-out is not feasible.

Due to its cost effective nature, it can be used widely in different educational campuses and industries.

Wi-Fi device can function in any type of geographical location.

Limitations

Like any other types of technology, Wi-Fi has its set of drawbacks that are listed as follows:

Global inconsistency of spectrum assignments and operational limitations.

Overlapping of channels.

Limited range of equivalent isotropically radiated power in some areas.

Greater power consumption compared to lower bandwidth standards.

Limited battery life due to range and reach requirements.

Wi-Fi network range is also limited.

MAC Protocols

The 802.11 standards use a MAC layer known as CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance).

In CSMA/CA a Wireless node that wants to transmit & performs the following sequence:

1. Listen on the desired channel.

2. If channel is idle (no active transmitters) it sends a packet.

3. If channel is busy then, the node waits until the transmission end then a contention period where minimum time a host must transmit before it can be sure that the no other host’s packet has collided with its transmission.

4. If the channel is still idle at the end of the contention period, then the node transmits its packet otherwise it repeats the process defined in step-3 above until it gets a free channel.

Use of RTS/CTS to Exchange Data

Step 1:

At first the sender check whether the medium is idle or not, if so, after the Distributed Inter Frame Space (DIFS will check the status and sense before transmitting the data in the wireless medium) units of time, it will broadcasts a Request-to-Send (RTS) frame to the receiver address.

Step 2:

If the receiver is within the range, then it will wait for Short Inter Frame Space (SIFS is the small time interval between the data frame and its acknowledgment) unit of time, then only it will respond to the sender with a Clear-to-Send (CTS) frame.

Step 3:

If the sender receive the CTS frame, then it will wait for another SIFS unit of time before sending the data frame to the receiver.

Step 4:

Finally, when the receiver will successfully receive the data frame, it will wait for SIFS unit of time and also send an Acknowledgement (ACK) message return to the sender.

Following figure-02 shows how data exchanges using RTC/CTS

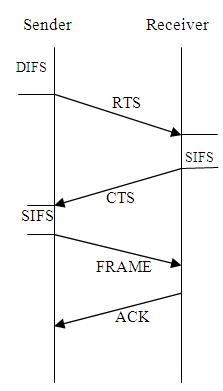


Figure-02:How Data exchanges using RTC/CTS

**Issues in Wi-Fi Networks**

Wi-Fi suffers from two well known problems:

* Hidden Terminal Problem
* Exposed Terminal Problem

**The Hidden Terminal Problem**

The hidden node/ terminal problem found at a point to multipoint network and it is defined as being one in which three or more nodes are present. Let there are three nodes :node A, node B and node C.

A and C cannot hear each other.

A sends to B, C cannot receive A.

C wants to send B, C senses a free medium.

Collision occurs at B.

A cannot receive the collision.

A is hidden for C.

Figure-03 illustrates the Hidden Terminal Problem using node A,B and C

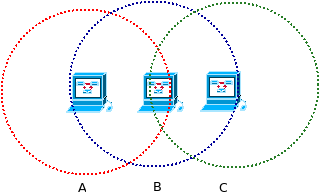


Figure-03: Hidden Terminal Problem

Solution of Hidden Terminal Problem

The solution of hidden terminal problem is as follows.

When A wants to send a packet to B , A first sends a Request-to-send (RTS) to B.

On receiving RTS, B responds by sending Clear-to-Send (CTS).

When C overhears a CTS, it keeps quiet for the duration of the transfer.

Transfer duration is included in both RTS and CTS.

RTS and CTS are short frames, reduces collision chance.

The other methods that can be employed to solve hidden terminal problem are :

- Increase transmitting power from the nodes.

- Use unidirectional antennas.

- Remove obstacles.

- Move the node.

- Use protocol enhancement software.

- Use antenna diversity.

Effect of Hidden Terminal Problem

If one node hidden to another then the re-transmission will increase. It also increase the delay and decrease the throughput.

Exposed Terminal Problem

Suppose there are four nodes: node A, node B, node C and node D.

Here -

B can send to both A and C .

C can send to D, but not to A or B.

A and C cannot hear each other.

Now the Problem as follows :

- When B transmits to A, C detects the transmission using the carrier sense mechanism.

- So C defers transmitting to D.

- But C could have sent to D, so blocked unnecessarily.

The following figure-04 shows the Exposed Terminal Problem using node A,B,C and D

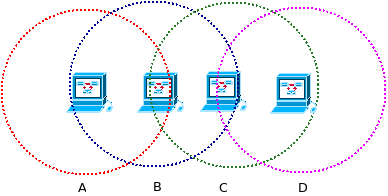


Figure-04: Exposed Terminal Problem

Exposed Terminal Problem

Solution to the Exposed Terminal Problem

Exposed terminal problems cannot be mitigated with RTS/CTS. This can be explained with the following scenario.

Suppose B sends RTS to A.

A sends CTS to B.

C hears RTS, but not CTS, assumes it is ok to send to D.

1. **Conclusion:**
2. **Viva Questions:**

* What is hidden terminal problem?
* What is exposed terminal problem?

1. **References:**

* Ekram Hossain and TeerawatIssariyakul, ―Introduction to Network Simulator NS-2, SpringerSecond Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No. : 5**

**Implementation of wireless network in NS-3.**

Experiment No. 5

1. **Aim:** Implementation wireless network in NS-3.
2. **Objectives:**

* To design a wireless network using NS3

1. **Outcomes:**

* Student will be able to design a wireless network in NS3
* Student will be able to run adhoc protocols in NS3

1. **Hardware / Software Required :** Turbo C
2. **Theory:** 
   1. Setup a 2-nodes wireless adhoc network. Place the nodes at a fixed distance in a 3dscenario.
   2. Install all the relevant network stacks, up to and including UDP.
   3. Setup a CBR transmission between the nodes, one acting as a server and one as a client. Take the iperf [1] behaviour as an example.
   4. Setup counters and outputs for packets sent and received.
   5. Schedule the simulation to run for enough time to obtain statistically relevant results (suggestion: analyze some test results and reduce the simulation time accordingly).
   6. Repeat the simulation varying the distance between the nodes from a minimum of1meter to the point where the nodes can't transmit/receive anymore.
   7. Repeat the above varying the channel models and the transmission/receive parameters like node's position above the ground, transmission power, etc.
   8. Show the differences between the various channel models, and comment them.

Identify the channel model that is more appropriate for each case (indoor, outdoor, LoS, NLoS, etc.).

Ns-3 provides a set of 802.11 models that attempt to provide an accurate MAC-level implementation of the 802.11specification and a “not-so-slow” PHY-level model of the 802.11a specification.

As we know both point-to-point and CSMA topology helper objects when constructing point-to-point topologies.

We can set nWifi to control how many STA (station) nodes are created in the simulation. There will always be one AP (accesspoint) node on the wireless network. By default there are three “extra” CSMA nodes and three wireless STA nodes.

The code begins by loading module include files. There are a couple of new includes corresponding to the Wifi module and the mobility module which we will discuss below.

#include "ns3/core-module.h"

#include "ns3/simulator-module.h"

#include "ns3/node-module.h"

#include "ns3/helper-module.h"

#include "ns3/wifi-module.h"

#include "ns3/mobility-module.h"

The ns-3 namespace is used and a logging component is defined.

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE ("ThirdScriptExample");

The main program begins by adding some command line parameters for enabling or disabling logging components and for changingthe number of devices created.

bool verbose = true;

uint32\_t nCsma = 3;

uint32\_t nWifi = 3;

CommandLinecmd;

cmd.AddValue (``nCsma'', ``Number of \"extra\" CSMA nodes/devices'', nCsma);

cmd.AddValue (``nWifi'', ``Number of wifi STA devices'', nWifi);

cmd.AddValue (``verbose'', ``Tell echo applications to log if true'', verbose);

cmd.Parse (argc,argv);

if (verbose)

{

LogComponentEnable(``UdpEchoClientApplication'', LOG\_LEVEL\_INFO);

LogComponentEnable(``UdpEchoServerApplication'', LOG\_LEVEL\_INFO);

}

the next step is to create two nodes that we will connect via the point-to-point link.

NodeContainer p2pNodes;

p2pNodes.Create (2);

Next, We instantiate a PointToPointHelper and set the associated default Attributes so that we create a five megabit persecond transmitter on devices created using the helper and a two millisecond delay on channels created by the helper. We then Installthe devices on the nodes and the channel between them.

PointToPointHelperpointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

Next, we declare another NodeContainer to hold the nodes that will be part of the bus (CSMA) network.

NodeContainercsmaNodes;

csmaNodes.Add (p2pNodes.Get (1));

csmaNodes.Create (nCsma);

The next line of code Gets the first node (as in having an index of one) from the point-to-point node container and adds it to the container of nodes that will get CSMA devices. The node in question is going to end up with a point-to-point device and a CSMA device. We then create a number of “extra” nodes that compose the remainder of the CSMA network.

We then instantiate a CsmaHelper and set its Attributes as we did in the previous example. We create a NetDeviceContainer to keep

track of the created CSMA net devices and then we Install CSMA devices on the selected nodes.

Csma Helper csma;

Csma .Set Channel Attribute ("DataRate", StringValue ("100Mbps"));

csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainercsmaDevices;

csmaDevices = csma.Install (csmaNodes);

Next, we are going to create the nodes that will be part of the Wifi network. We are going to create a number of “station” nodes as

specified by the command line argument, and we are going to use the “leftmost” node of the point-to-point link as the node for the

access point.

Node Container wifi StaNodes;

Wifi StaNodes.Create (nWifi);

NodeContainerwifiApNode = p2pNodes.Get (0);

The next bit of code constructs the wifi devices and the interconnection channel between these wifi nodes. First, we configure the

PHY and channel helpers:

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();

YansWifiPhyHelperphy = YansWifiPhyHelper::Default ();

For simplicity, this code uses the default PHY layer configuration and channel models which are documented in the API doxygen

documentation for the YansWifiChannelHelper::Default and YansWifiPhyHelper::Default methods. Once these objects are created,

we create a channel object and associate it to our PHY layer object manager to make sure that all the PHY layer objects created bythe YansWifiPhyHelper share the same underlying channel, that is, they share the same wireless medium and can communicationand interfere:

phy.SetChannel (channel.Create ());

Once the PHY helper is configured, we can focus on the MAC layer. Here we choose to work with non-Qos MACs so we use aNqosWifiMacHelper object to set MAC parameters.

WifiHelperwifi = WifiHelper::Default ();

wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

NqosWifiMacHelper mac = NqosWifiMacHelper::Default ();

The SetRemoteStationManager method tells the helper the type of rate control algorithm to use. Here, it is asking the helper to usethe AARF algorithm details are, of course, available in Doxygen.Next, we configure the type of MAC, the SSID of the infrastructure network we want to setup and make sure that our stations don’tperform active probing:

1. **Conclusion:**
2. **Viva Questions:** 
   * What is the difference between wired and wireless networks?
   * Which protocol is used for routing?
3. **References:**

* Ekram Hossain and TeerawatIssariyakul, ―Introduction to Network Simulator NS-2, SpringerSecond Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No. : 6**

Implementing IEEE 802.11 wireless LAN in Ad-Hoc Mode using NS2.

Experiment No. 6

1. **Aim:** Simulating IEEE 802.11 wireless LAN using NS2(Ad-hoc Mode).
2. **Objectives:**

* To understand the basic concepts of wireless LAN (WLAN).

1. **Outcomes:**

* Students will be able to explore the evolution of mobile generation and understand the core concepts of Wireless LAN

1. **Hardware / Software Required:** NS2
2. **Theory:**

A wireless local-area network (LAN) uses radio waves to connect devices such as laptops to the Internet and to your business network and its applications. When you connect a laptop to a WiFi hotspot at a cafe, hotel, airport lounge, or other public place, you're connecting to that business's wireless network.

**Ad hoc mode:** An Ad-hoc network allows each device to communicate directly with each other. There is no central Access Point controlling device communication. Ad-hoc networks are only able to communicate with other Ad-hoc devices, they are not able to communicate with any Infrastructure devices or any other devices connected to a wired network. In addition, Ad-hoc mode security is less sophisticated compared to an Infrastructure mode network.

**Infrastructure mode:** An Infrastructure mode network requires the use of an Access Point. The Access Point controls Wireless communication and offers several important advantages over an Ad-hoc network. For example, a Infrastructure based network supports increased levels of security, potentially faster data transmission speeds and integration with a wired network.

**Historical Profile:**

Heinrich Herz discovered and first produced radio waves in 1888 and by 1894 the modern way to send a message over telegraph wires was first conducted. Marconi sent and received signals up to two miles using radio waves. Marconi became known as the “father of radio”. By 1899, Marconi sent a signal nine miles across the Bristol Channel and 31 miles across the English Channel to France. In 1901 he was able to transmit across the Atlantic Ocean.

During World War II, the United States Army first used radio signals for data transmission. This inspired a group of researchers in 1971 at the University of Hawaii to create the first packet based radio communications network called ALOHNET. ALOHNET was the very first wireless local area network (WLAN). This first WLAN consisted of 7 computers that communicated in a bi-directional star topology.

The first generation of WLAN technology used an unlicensed band (902-928 MHz ISM), which later became crowded with interference from small appliances and industrial machinery. A spread spectrum was used to minimize this interference, which operated at 500 kilobits per second. The second generation of WLAN technology was four times faster and operating at 2Mbps per second. Third generation WLAN technology operates on the same band as the second generation and we currently use it today.

In 1990, the IEEE 802 Executive Committee established the 802.11 Working Group to create a wireless local area network (WLAN) standard. The standard specified an operating frequency in the 2.4GHz ISM band. In 1997 the group approved IEEE 802.11 as the world's first WLAN standard with data rates of 1 and 2 Mbps.

**New Concepts to be learned:**

NS-2isaneventdrivennetworksimulatordevelopedatUCBerkeleythatsimulatesvarietyofIPnetworks.Itisanobjectoriented,eventdrivennetworksimulatorwritteninC++withOTclasfront-end.Sincetwodifferenttasks–simulationofprotocols&networkresearch-aretobeperformedNS-2supportstwolanguagesC++&OTcl.

ItcanbeusedtoimplementnetworkprotocolssuchasTCPandUDP,trafficsourcebehaviorsuchasFTP,Telnet,Web,CBRandVBR,routerqueuemanagementmechanismsuchasDropTail,REDandCBQ,routing algorithmssuchasDijkstra, andmore.NSprovidessubstantialsupportforsimulationofTCP,routing,andmulticastprotocolsoverwiredandwireless(localandsatellite)Networks.NSalsoimplementsmulticastingandsomeoftheMAClayerprotocols(802based)forLANsimulations.

NSsupportssimulationofmobilenetworkbasedonwirelessMAC.Followingisthescenariointhenswirelessmodel:

* Mobile-nodeatcoreofmobilitymodel
* Mobilenodescanmoveinagiventopology,receive/transmitsignalsfrom/towirelesschannels
* WirelessnetworkstackconsistsofLL,ARP,MAC,IFQetc.
* Allowssimulationsofmulti-hopadhocnetworks,wirelessLANs,sensornetworksetc.

1. **Conclusion:**

Students have successfully understood basic concepts of Wireless LAN (WLAN), and they have used the knowledge to simulate a basic WLAN in Ad-hoc mode using NS2.

1. **Viva Questions:**

* What is WLAN?
* What is NS2?

**8 .References:**

* Ekram Hossain and Teerawat Issariyakul, ―Introduction to Network Simulator NS-2, Springer Second Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No.: 7**

Implementation of Bluetooth network for data transfer from one node to other node in NS-2.

Experiment No. 7

1. **Aim:** Implementation of Bluetooth network for data transfer from one node to other node in NS-2.
2. **Objectives:**

* To understand the working of bluetooth
* To know the difference piconet & scatternet

1. **Outcomes:**

* Understand about Bluetooth networks and how it is differentiate from WiFi networks
* Simulating the Bluetooth networks with NS2 by using UCBT extensions

1. **Hardware / Software Required :** Turbo C
2. **Theory:**

The name Bluetooth was named after 10th century Viking king in Denmark Harald Bluetooth who united and controlled Denmark and Norway [i]. Bluetooth is a low power, low cost, and short range radio network standard. It supports unlicensed Industrial, Scientific and Medical (ISM) 2.4 GHz short-range radio frequency band. It is complementary to the Wi-Fi network specified by IEEE 802.11b/g/a standard. Bluetooth uses a frequency hopping scheme to provide robust wireless communication. As a cable replacement, Bluetooth is widely used in cell phone, PDA, laptop, headset, and printer, etc. to form a Personal Area Network (PAN) and provide universal access . Bluetooth uses a radio technology called frequency-hopping spread spectrum

**Networking of Bluetooth**

Bluetooth technology provides both point-to-point and point-to-multipoint connection. In point-to-multipoint connections, the channel is shared among several Bluetooth units. In point-to-point connections, only two units share the connection. Bluetooth protocols assume that a small number of units will participate in communications at any given time. These small groups are called piconets , and they consist of one master unit and up to seven active slave units. The master is the unit that initiates transmissions, and the slaves are responding units. This type of Bluetooth network can have only one master unit. If several piconets overlap a physical area, and members of the various piconets communicate with each other, this new, larger network is known as a scatternet . Any unit in one piconet can communicate in a second piconet as long as it serves as master for only one piconet at a time. The following figure-01 illustrated the Bluetooth Scatternet scenario with slave/slave node.

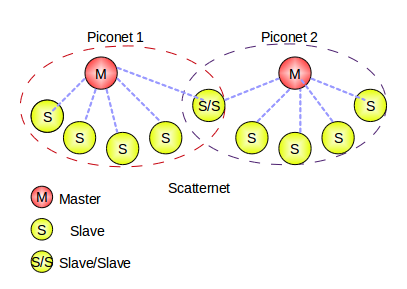


Figure-01: Illustration of Bluetooth Piconets&Scatternet with Slave/Slave node

When the no. of piconet connect with each other with the slave nodes, then the slave nodes, which are connected with both the Master nodes of Piconets are known as Slave/Slave nodes. But when the no. of piconets are connected with each other with another master nodes, then the master nodes, which are Slave to one Piconet and Master for other Piconets are known as Master/Slave nodes.The following figure-02 illustrated the Bluetooth Scatternet with master/slave scenario.

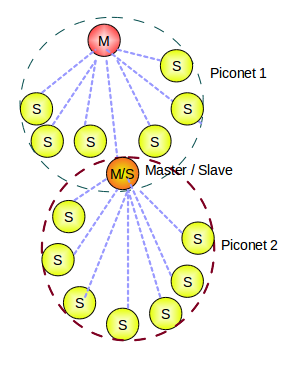


Figure-02: Illustration of Bluetooth Scatternet with Master/Slave node

# Script showing a layer 2 handover in 802.16

# Scenario: Communication between MN and Sink Node with MN attached to BS.

# Notes:- In order to perform layer 2 handover, the BSs must share the same channel

# - Additional mechanisms are required in order to continue a communication after a layer 2 handover This is achieved by updating the MN address to simulate layer 3 handovers and traffic redirection.

# Topology scenario:

#

#

# |-----|

# | MN0 | 2.0.1

# |-----|

#

#

# (^) (^)

# | |

# |--------------| |--------------|

# | Base Station | 2.0.0 | Base Station | 3.0.0

# |--------------| |--------------|

# | |

# | |

# |-----------| |

# | Router |---------------------|

# |-----------| 1.0.0

# |

# |-----------| |

# | Sink node |---------------------|

# |-----------| 0.0.0

#

# Explanations of scenario/information to follow the code:

# 1- Upon link going down (i.e power level decreased and crossed a

# threshold), the MS sends a scan request to its serving BS

# 2- when the WimaxCtrlAgent located in the BS receives the scan

# request, it needs to respond to the SS. In case of no association or

# association without coordination, the BS handles the request locally

# and can respond right away. When there is coordination with other

# neighboring BSs, it cannot do that since it has to consult these

# BSs. That's why the code is commented in case 2, but you will see

# that it is sending synchronization request to the neighbor BSs. Case

# 3 as not been implemented.

# 3- When the MS receives the response, there is a delay as defined in the

# standard (note that the scanning does not start but it is set to

# PENDING). Also it checks for Association with coordination. If there

# is coordination, there are rendez-vous time that were defined by the

# serving BS (in the WimaxCtrlAgent) and we need to schedule timers so

# that we switch to the right channel at the right time.

# After few frames the scanning will start then a serie of

# start/pause/resume/stop scanning depending on the number of scan

# iterations.

# 4- Handover: SS will send MSHO-REQ message to the serving BS:

# message type = MAC\_MOB\_MSHO\_REQ. The data includes, id and mean RSSI

# of current serving BS; id and mean RSSI of neighbor BSs that where

# detected during scanning. The serving BS will response with MSHO-RSP

# to the SS: message type =MAC\_MOB\_BSHO\_RSP, with a recommended target

# BS (the one with the strongest RSSI).

# 5- SS receives the response from serving BS and starts to HO

# with the recommended BS. If the recommended BS is

# different from the current serving BS, it will send a message to

# serving BS.

#check input parameters

if {$argc != 0} {

puts ""

puts "Wrong Number of Arguments! No arguments in this topology"

puts ""

exit

}

# set global variables

set output\_dir .

set traffic\_start 15

set traffic\_stop 250

set simulation\_stop 250

#define debug values

Mac/802\_16 set debug\_ 0

Agent/WimaxCtrl set debug\_ 1

Mac/802\_16 set print\_stats\_ 0

Mac/802\_16 set t21\_timeout\_ 0.02 ;#20

Mac/802\_16/BS set dlratio\_ .66

Mac/802\_16/SS set dlratio\_ .66

Mac/802\_16 set ITU\_PDP\_ 1

Mac/802\_16 set lgd\_factor\_ 10.0 ;#note: the higher the value the earlier the trigger

Mac/802\_16 set rxp\_avg\_alpha\_ 0.2 ;# the higher the value (max=1), the more sensitive the scan trigger

Agent/WimaxCtrl set adv\_interval\_ 1.0

Agent/WimaxCtrl set default\_association\_level\_ 0

Agent/WimaxCtrl set synch\_frame\_delay\_ 0.5

#Physical layer configuration

Phy/WirelessPhy set Pt\_ 0.2

Phy/WirelessPhy set RXThresh\_ 1.90546e-16

Phy/WirelessPhy set CSThresh\_ [expr 0.9\*[Phy/WirelessPhy set RXThresh\_]]

Phy/WirelessPhy set OFDMA\_ 1

# Parameter for wireless nodes

set opt(chan) Channel/WirelessChannel ;# channel type

set opt(prop) Propagation/OFDMA ;# radio-propagation model

set opt(netif) Phy/WirelessPhy/OFDMA ;# network interface type

set opt(mac) Mac/802\_16 ;# MAC type

set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type

set opt(ll) LL ;# link layer type

set opt(ant) Antenna/OmniAntenna ;# antenna model

set opt(ifqlen) 50 ;# max packet in ifq

set opt(adhocRouting) DSDV ;# routing protocol

set opt(x) 2000 ;# X dimension of the topography

set opt(y) 2000 ;# Y dimension of the topography

#defines function for flushing and closing files

proc finish {} {

global ns tfoutput\_dirnb\_mn

$ns flush-trace

close $tf

exit 0

}

#create the simulator

set ns [new Simulator]

$ns use-newtrace

#create the topography

set topo [new Topography]

$topoload\_flatgrid $opt(x) $opt(y)

#open file for trace

set tf [open $output\_dir/out.res w]

$ns trace-all $tf

# set up for hierarchical routing (needed for routing over a basestation)

$ns node-config -addressType hierarchical

AddrParams set domain\_num\_ 4 ;# domain number

lappendcluster\_num 1 1 1 1 ;# cluster number for each domain

AddrParams set cluster\_num\_ $cluster\_num

lappendeilastlevel 1 1 2 2 ;# number of nodes for each cluster

AddrParams set nodes\_num\_ $eilastlevel

puts "Configuration of hierarchical addressing done"

# Create God

create-god 3 ;#nb\_mn + 2 (base station and sink node)

#creates the sink node in first addressing space.

set sinkNode [$ns node 0.0.0]

puts "sink node created"

set router [$ns node 1.0.0]

puts "router node created"

#create common channel

set channel [new $opt(chan)]

#creates the Access Point (Base station)

$ns node-config -mobileIP ON \

-adhocRouting $opt(adhocRouting) \

-llType $opt(ll) \

-macType Mac/802\_16/BS \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-channel $channel \

-topoInstance $topo \

-wiredRouting ON \

-agentTrace ON \

-routerTrace ON \

-macTraceON \

-movementTrace OFF

#setup channel model

set prop\_inst [$ns set propInstance\_]

$prop\_inst ITU\_PDP PED\_A

#puts "Configuration of base station"

set bstation [$ns node 2.0.0]

$bstation random-motion 0

#provide some co-ord (fixed) to base station node

$bstation set X\_ 50.0

$bstation set Y\_ 50.0

$bstation set Z\_ 0.0

[$bstation set mac\_(0)] set-channel 0

#add MOB\_SCN handler

set wimaxctrl [new Agent/WimaxCtrl]

$wimaxctrl set-mac [$bstation set mac\_(0)]

$ns attach-agent $bstation $wimaxctrl

puts "Base Station 1 created"

set bstation2 [$ns node 3.0.0]

$bstation2 random-motion 0

#provide some co-ord (fixed) to base station node

$bstation2 set X\_ 800.0

$bstation2 set Y\_ 50.0

$bstation2 set Z\_ 0.0

[$bstation2 set mac\_(0)] set-channel 1

#add MOB\_SCN handler

set wimaxctrl2 [new Agent/WimaxCtrl]

$wimaxctrl2 set-mac [$bstation2 set mac\_(0)]

$ns attach-agent $bstation2 $wimaxctrl2

puts "Base Station 2 created"

#Add neighbor information to the BSs

$wimaxctrl add-neighbor [$bstation2 set mac\_(0)] $bstation2

$wimaxctrl2 add-neighbor [$bstation set mac\_(0)] $bstation

# creation of the mobile nodes

$ns node-config -macType Mac/802\_16/SS \

-wiredRouting OFF \

-macTraceON ;# Mobile nodes cannot do routing.

set wl\_node [$ns node 2.0.1] ;# create the node with given @.

$wl\_node random-motion 0 ;# disable random motion

$wl\_node base-station [AddrParams addr2id [$bstation node-addr]] ;#attach mn to basestation

$wl\_node set X\_ 300.0

$wl\_node set Y\_ 40.0

$wl\_node set Z\_ 0.0

set HAaddress [AddrParams addr2id [$bstation node-addr]]

[$wl\_node set regagent\_] set home\_agent\_ $HAaddress

$ns at 15.0 "$wl\_nodesetdest 750.0 40.0 5.0"

$ns at 100.0 "$wl\_nodesetdest 100.0 40.0 5.0"

[$wl\_node set mac\_(0)] set-channel 0

[$wl\_node set mac\_(0)] set-diuc 1

[$wl\_node set mac\_(0)] setflow UL 10000 BE 275 2 0 0.05 15 1 0 0 0 0 0 0 0 0 0 0 ;# setting up static flows

[$wl\_node set mac\_(0)] setflow DL 10000 BE 275 2 0 0.05 15 1 0 0 0 0 0 0 0 0 0 0 ;# setting up static flows

puts "wireless node created ..." ;# debug info

#create source traffic

#Create a UDP agent and attach it to node n0

set udp [new Agent/UDP]

$udp set packetSize\_ 1500

$ns attach-agent $sinkNode $udp

# Create a CBR traffic source and attach it to udp0

set cbr [new Application/Traffic/CBR]

$cbr set packetSize\_ 1000

$cbr set interval\_ 0.5

$cbr attach-agent $udp

#create an sink into the sink node

# Create the Null agent to sink traffic

set null [new Agent/Null]

$ns attach-agent $wl\_node $null

# Attach the 2 agents

$ns connect $udp $null

# create the link between sink node and base station

$ns duplex-link $router $bstation 100Mb 3ms DropTail

$ns duplex-link $router $bstation2 100Mb 3ms DropTail

$ns duplex-link $sinkNode $router 100Mb 3ms DropTail

# Traffic scenario: here the all start talking at the same time

$ns at $traffic\_start "$cbr start"

$ns at $traffic\_stop "$cbr stop"

$ns at $simulation\_stop "finish"

puts "Running simulation"

$ns run

puts "Simulation done."

1. **Conclusion:**
2. **Viva Questions:**

* What is bluetooth?
* What are the differences between bluetooth and Wlan?

1. **References:**

* Ekram Hossain and TeerawatIssariyakul, ―Introduction to Network Simulator NS-2, SpringerSecond Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No. : 8**

**Develop a simple wireless adhoc network using aodv, aomdv & tora. Also compare the performance using NS-2.**

Experiment No. 8

**1. Aim:** Develop a simple wireless adhoc network using aodv, aomdv & tora. Also compare the performance using NS-2.

**2. Objectives:**

* Understand different routing protocol of adhoc networks

**3. Outcomes:**

* Implement different routing protocols
* Differentiate different routing protocols

**4. Hardware / Software Required :** NS2, linux

**5. Theory:**

The step by step process happening in AODV network simulation in NS2

1. In the TCL script, when the user configures AODV as a routing protocol by using the command,

$ns node-config -adhocRouting AODV

the pointer moves to the “start” and this “start” moves the pointer to the Command function of AODVprotocol.

2. In the Command function, the user can find two timers in the “start”

\* btimer.handle((Event\*) 0);

\* htimer.handle((Event\*) 0);

3. Let’s consider the case of htimer, the flow points toHelloTimer::handle(Event\*) function and the user can see the following lines:

agent ->sendHello();

double interval = MinHelloInterval + ((MaxHelloInterval - Min-HelloInterval) \* Random::uniform());

assert(interval -> = 0);

Scheduler::instance().schedule(this, &intr, interval);

These lines are calling the sendHello() function by setting the appropriate interval of Hello Packets.

4. Now, the pointer is in AODV::sendHello() function and the user can seeScheduler::instance().schedule(target , p, 0.0) which will schedule the packets.

5. In the destination node AODV::recv(Packet\*p, Handler\*) is called, but actually this is done after the node is receiving a packet.

6. AODV::recv(Packet\*p, Handler\*) function then calls therecvAODV(p) function.

7. Hence, the flow goes to the AODV::recvAODV(Packet \*p) function, which will check different packets types and call the respective function.

8. In this example, flow can go to case

AODVTYPE HELLO:

recvHello(p);

break;

9. Finally, in the recvHello() function, the packet is received

**TORA (Temporally Ordered Routing Algorithm)** is a source initiated on demand routing protocol.

TORA is a highly adaptive, efficient, loop-free and scalable routing protocol based on link reversal algorithm.

The main objective of TORA is to limit message propagation in the highly dynamic mobile computing environment. It means, it is designed to reduce communication overhead by adapting local topological changes in ad hoc network. Another main feature of TORA routing protocol is the localization of control packets to a small region (set of nodes) near the occurrence of a topological changes due to route break. Hence, each node of the network required to contain its local routing and topology information about adjacent nodes.

TORA supports multiple routes to transmit data packet between source and destination nodes of mobile ad hoc network. In short, TORA exhibits multipath routing capability.

The TORA's operation can be compared to that of water flowing downhill toward a sink node through a grid of tubes that model the routes in the real world network. The tube junctions represent the nodes, the tube themselves represent the route links between the nodes, the tube's water represents the packets flowing between nodes through the route links toward the destination.

**6. Conclusion:**

**7. Viva Questions:**

* What is the difference between all 3 protocols?
* What are the metrics of all 3 protocols?

**8. References:**

* Ekram Hossain and TeerawatIssariyakul, ―Introduction to Network Simulator NS-2, SpringerSecond Edition. (T1)
* Jack L.Burbank,―Introduction to Network Simulator 3,Wiley Publications(T2)
* Siva Ram Murthy and B.S. Manoj, ―Adhoc Wireless Networks Architectures and protocols, 2ndedition, Pearson Education, 2007 (T3)
* Michael Gregg,―Build your own Security lab, Wiley India edition

**Experiment No. : 12**

**Mini Project**