**WIRELESS COMMUNICATIONS LAB**

**(ETCS – 463)**

**Faculty name:** Mr. Ashish Sharma **Student name:** Ayush Pandey

**Roll No.:** 45014802718

**Semester:** 7th Semester

**Group:** 7-C-8



Maharaja Agrasen Institute of Technology

PSP Area, Sector – 22, Rohini, New Delhi – 110085



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**VISION**

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

**MISSION**

**The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:**

**Engineering Hardware – Software Symbiosis**

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

**Life – Long Learning**

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

**Liberalization and Globalization**

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

**Diversification**

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

**Entrepreneurship**

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**

**VISION**

To produce “Critical Thinkers of Innovative Technology”.

**MISSION**

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real life problems and fulfil societal needs.

**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Outcome Based Learning**

**Course Outcomes (Revision)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Subject:** | **WIRELESS COMMUNICATIONS** | **Max Marks Internal** | **60** |
| **Subject Code** | ETCS 463 |  |  |
| **Total Credit** | 1 | Evaluation Scheme | |
| Contact Hours |  | Theory/Lab | Lab |
|  |  |  |

**Course Objectives:**

The course will introduce radio propagation and transmission principles used in different wireless communication systems .The course will discuss radio channel characteristics such as fading, interference and diversity to develop a good understanding of the radio engineering area. The course will discuss different techniques used to support voice, data and video communication in wireless systems. The course will also discuss the wireless networks and their basic design thereof from both theoretical and practical points of view.

|  |  |
| --- | --- |
| **S. No.** | **Course Outcomes** |
| C463.1 | Calculate various parameters for cellular wireless systems using Scilab. |
| C463.2 | Understand various concepts, protocols, technologies related to wireless Communication systems by simulating them using Scilab or NS3. |
| C463.3 | Analyze and apply mathematical models of different wireless networks concepts using NS3 or Scilab. |
| C463.4 | Create and Configure Wi-fi networks, Plot using 2D/3D graphs generation and reception of signals in Wireless Cellular networks. |
| C463.5 | Compare and analyze data transmission using different modulation techniques and BER performance of different channels in wireless communication networks. |
| C463.6 | Do R and D in Latest Trends in Wireless Communication, document it and publish research papers in reputed conference or journals |

**WIRELESS COMMUNICATIONS LAB**

**Paper Code: ETCS-463 L T/P C**

**Paper: WIRELESS COMMUNICATIONS LAB 0 2 1**

**List of Experiments:**

1. **Write a program in Scilab to Calculate Frequency Reuse Distance ,Co-Channel Interference reduction factor, Cellular System Capacity, S/I Ratio for a given variables.**
2. **Write a Program in Scilab to calculate maximum traffic intensity and maximum no. of users accommodated in Erlang B and Erlang C system for given no of channels.**
3. **Write a Program in Scilab to calculate Bit Error rate performance of BPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both.**
4. **Program in Scilab to Generate Walsh Codes and then spread the user information using it.**
5. **Program in Scilab to Generate PN Sequence for CDMA Systems.**
6. **Write a Program in NS3 to connect WIFI TO BUS (CSMA) Network.**
7. **Write a Program in NS3 to create WIFI Network in SIMPLE INFRASTUCTURE MODE (of nodes).**
8. **Write a Program in NS3 to create a wireless mobile ad-hoc network between three nodes.**

**PRACTICAL RECORD**

**PAPER CODE : ETCS-463**

**Name of the student : Ayush Pandey**

**University Roll No. : 45014802718**

**Branch : CSE**

**Group : 7C-8**

**PRACTICAL DETAILS**

1. Experiments according to WC lab syllabus prescribed by GGSIPU

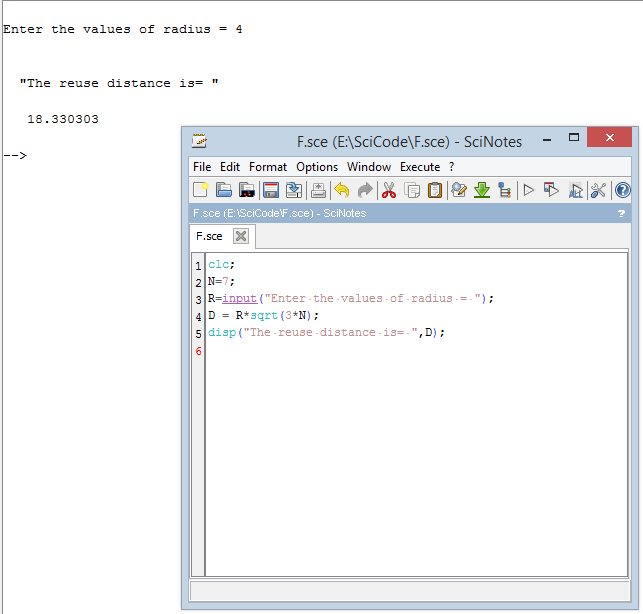
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Exp. No.** | **Experiment Name** | **Marks** | | | | | **Total Marks** | **Signature with Date** |
| **R1** | **R2** | **R3** | **R4** | **R5** |  |  |
| 1. | WAP in Scilab to Calculate Frequency Reuse Distance ,Co-Channel Interference reduction factor, Cellular System Capacity, S/I Ratio for a given variables. |  |  |  |  |  |  |  |
| 2. | WAP in Scilab to calculate max traffic intensity and max no. of users accommodated in Erlang B and Erlang C system for given no of channels. |  |  |  |  |  |  |  |
| 3. | Write a Program in Scilab to calculate Bit Error rate performance of BPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both. |  |  |  |  |  |  |  |
| 4. | Program in Scilab to Generate Walsh Codes and then spread the user information using it. |  |  |  |  |  |  |  |
| 5. | Program in Scilab to Generate PN Sequence for CDMA Systems. |  |  |  |  |  |  |  |
| 6. | Write a Program in ns3 to connect wifi to bus(csma) network. |  |  |  |  |  |  |  |
| 7. | Write a Program in ns3 to create wifi network in simple infrastructure mode (of nodes). |  |  |  |  |  |  |  |
| 8. | Write a Program in NS3 to create a wireless mobile ad-hoc network between three nodes. |  |  |  |  |  |  |  |

**Date: 30/09/2021**

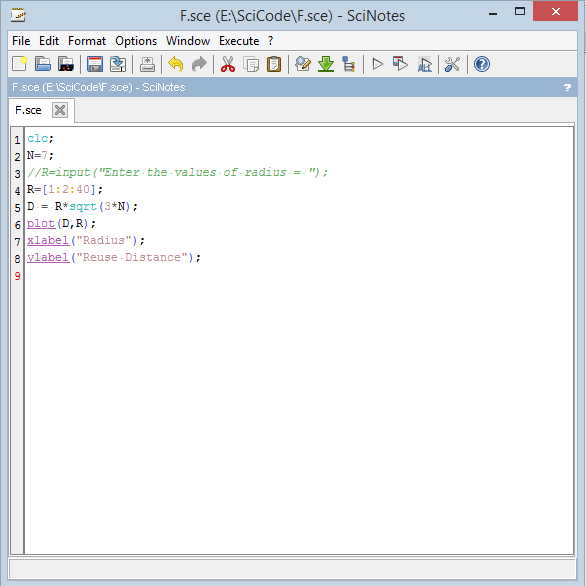
**Experiment-1**

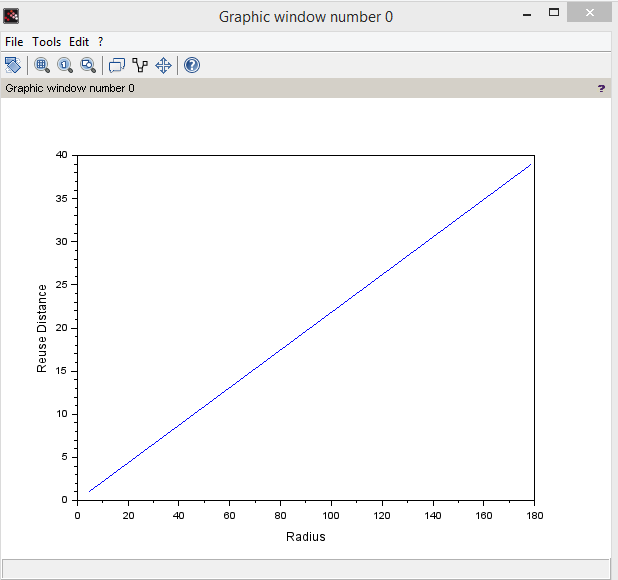
**Aim:** Write a program in Scilab to Calculate Frequency Reuse Distance ,Co-Channel Interference reduction factor, Cellular System Capacity, S/I Ratio for a given variables.

* **Frequency Reuse Distance**

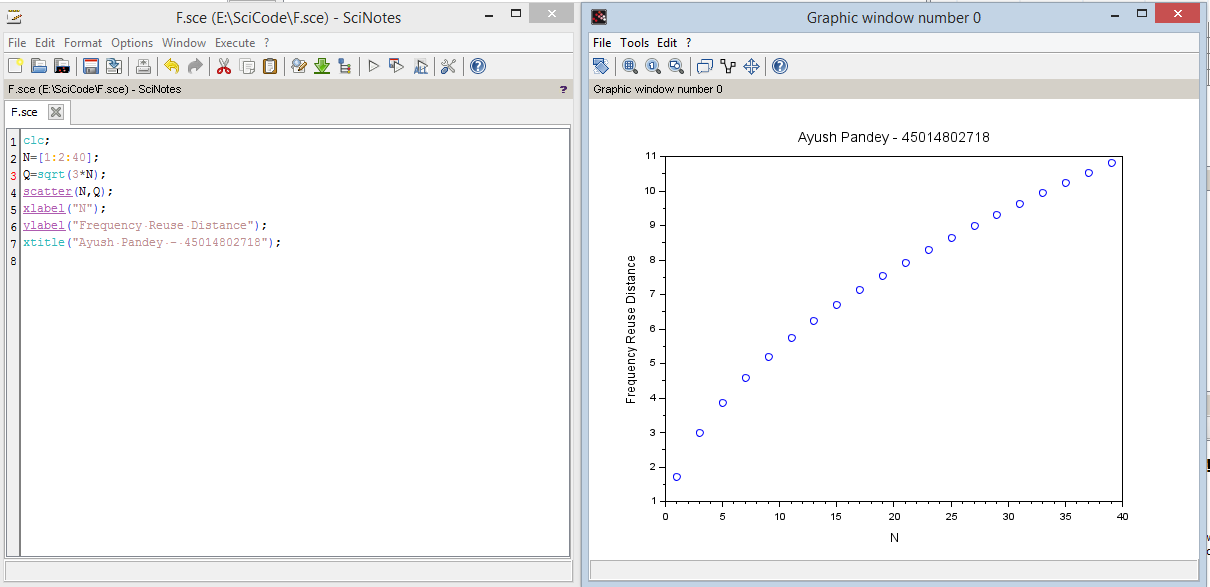


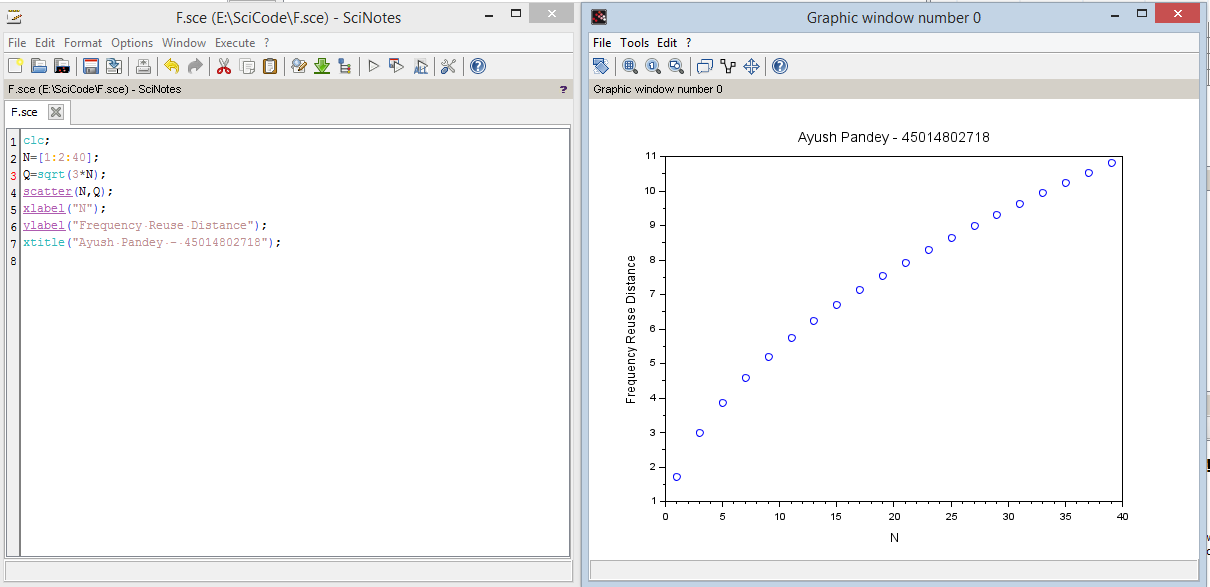
* **Radius vs Reuse Distance Graph**



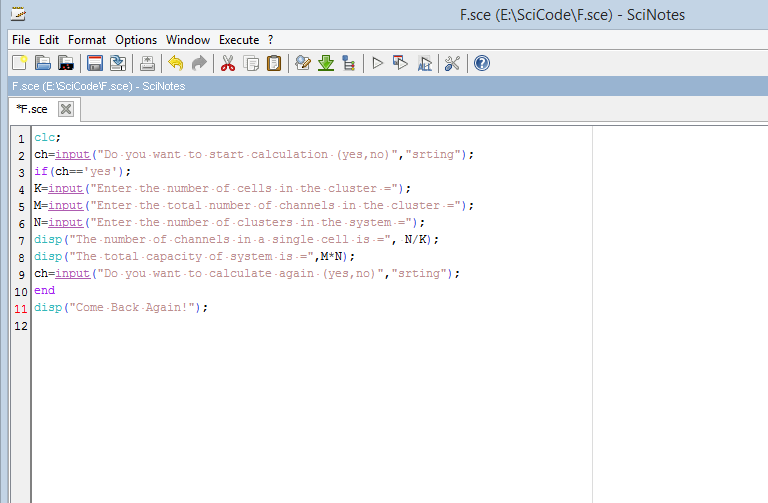


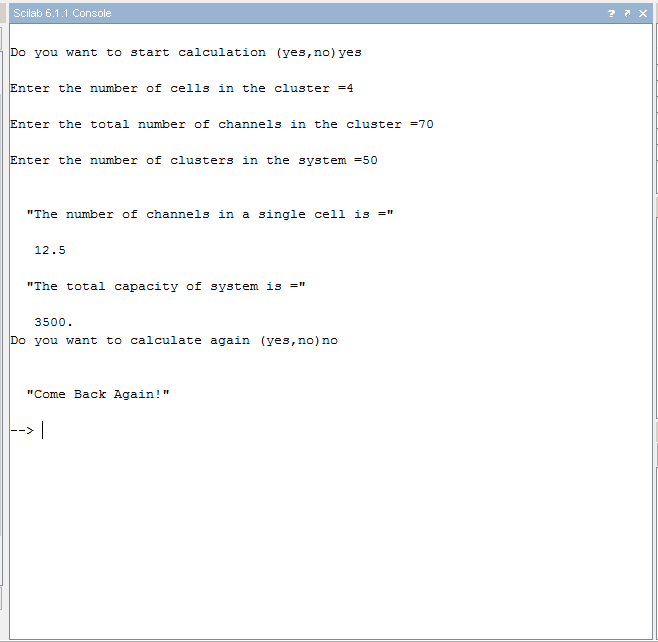
* **N vs Reuse Distance Scatter Graph**

****

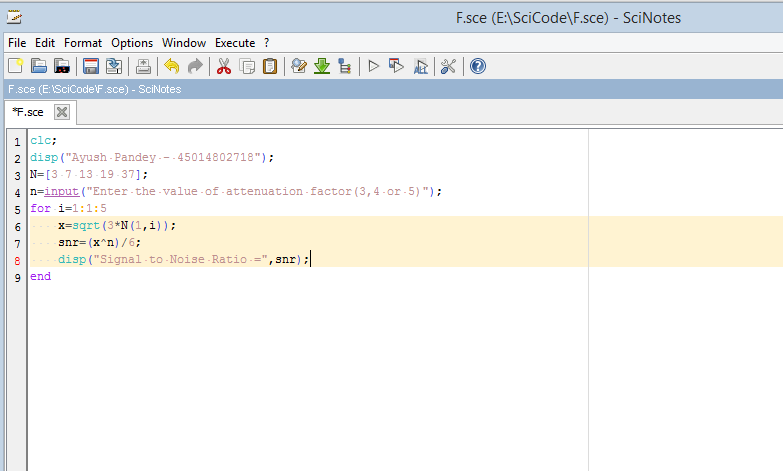
****

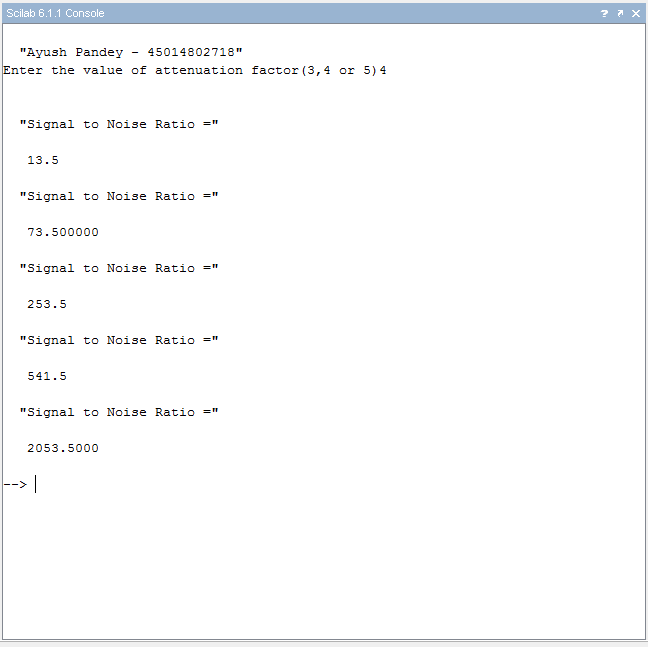
* **Channels in a single cell & Total Capacity**

****

****

* **S/I Ratio**

****

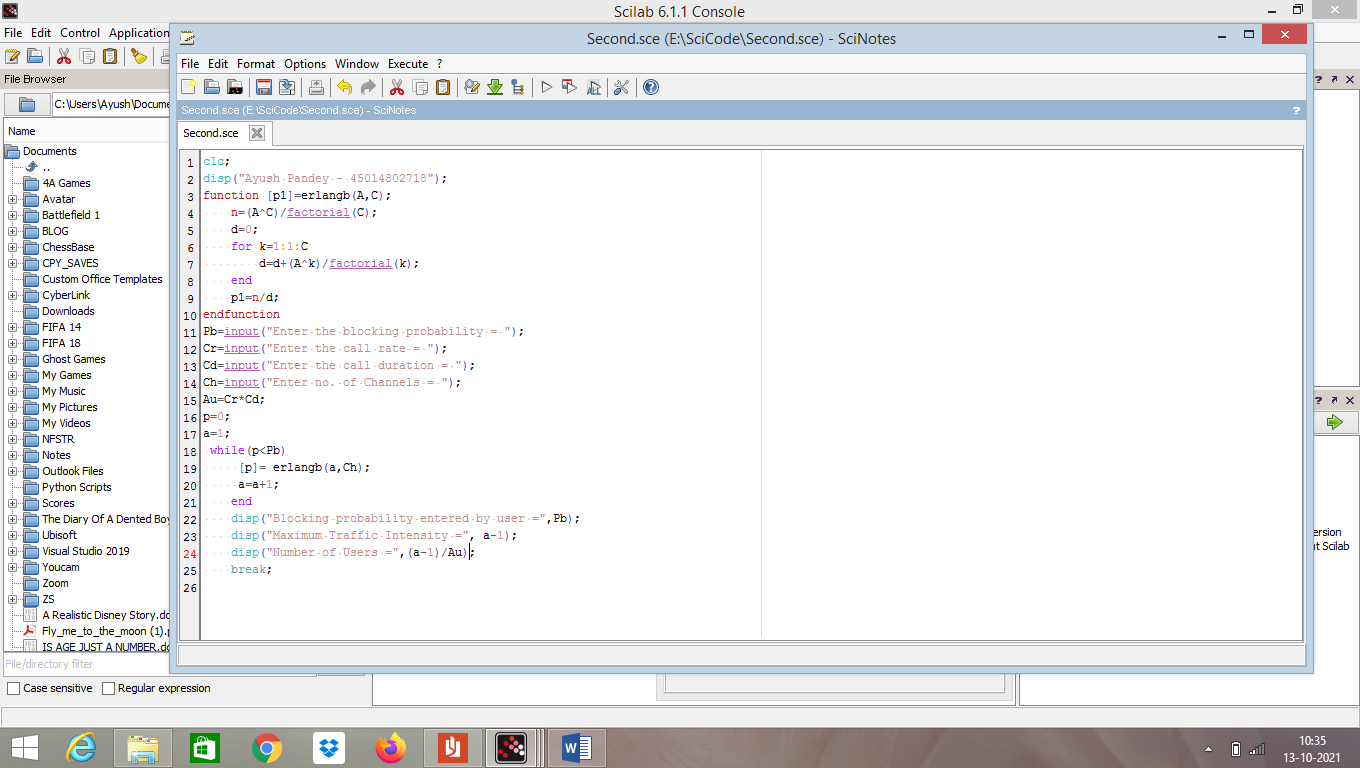
****

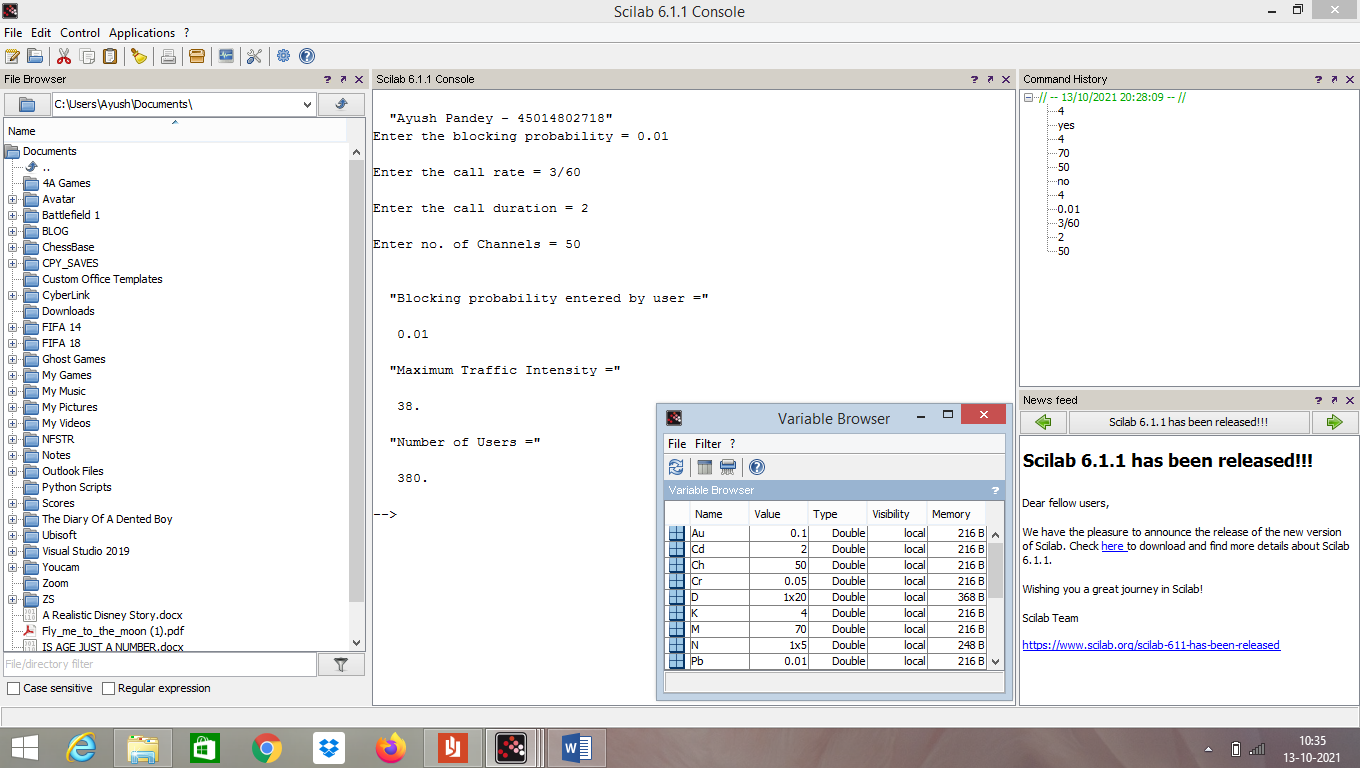
**Date: 07/10/2021**

**Experiment-2**

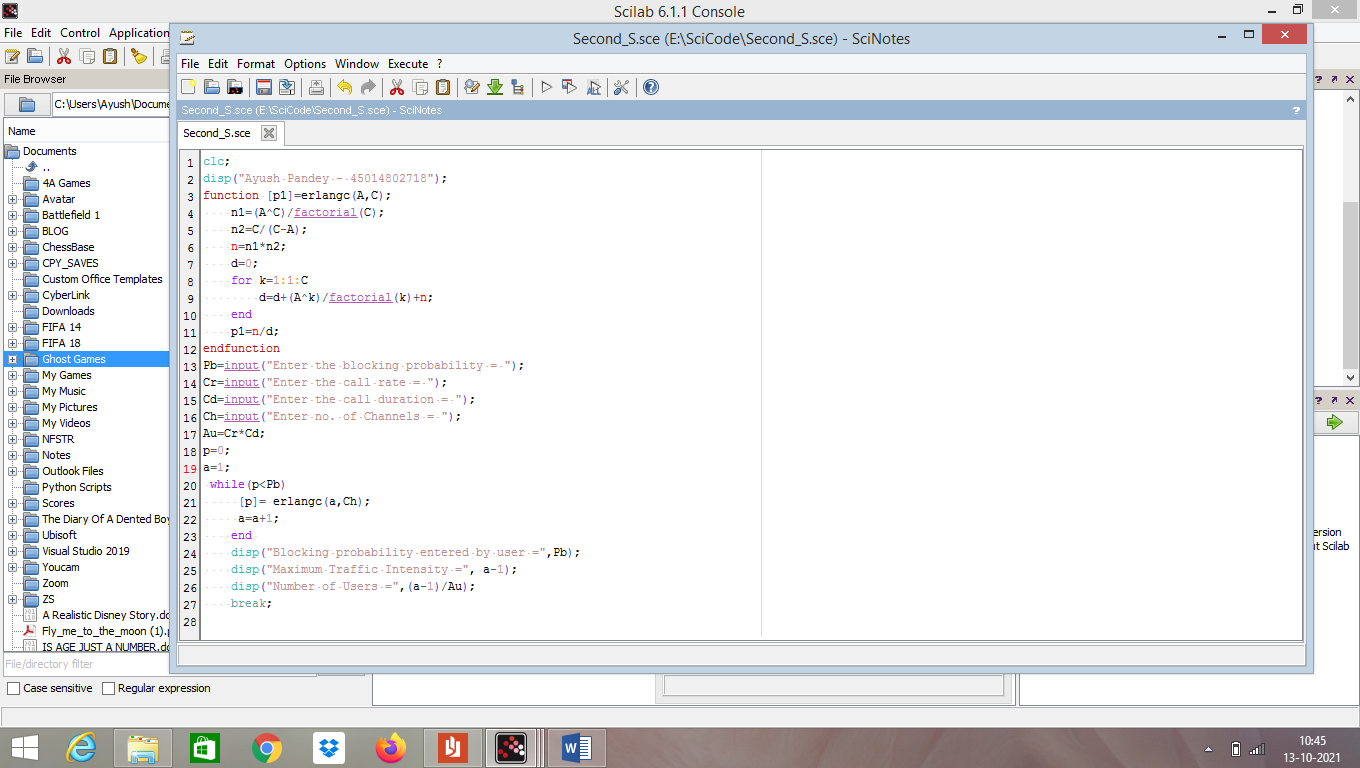
**Aim:** Write a Program in Scilab to calculate maximum traffic intensity and maximum no. of users accommodated in Erlang B and Erlang C system for given no of channels.

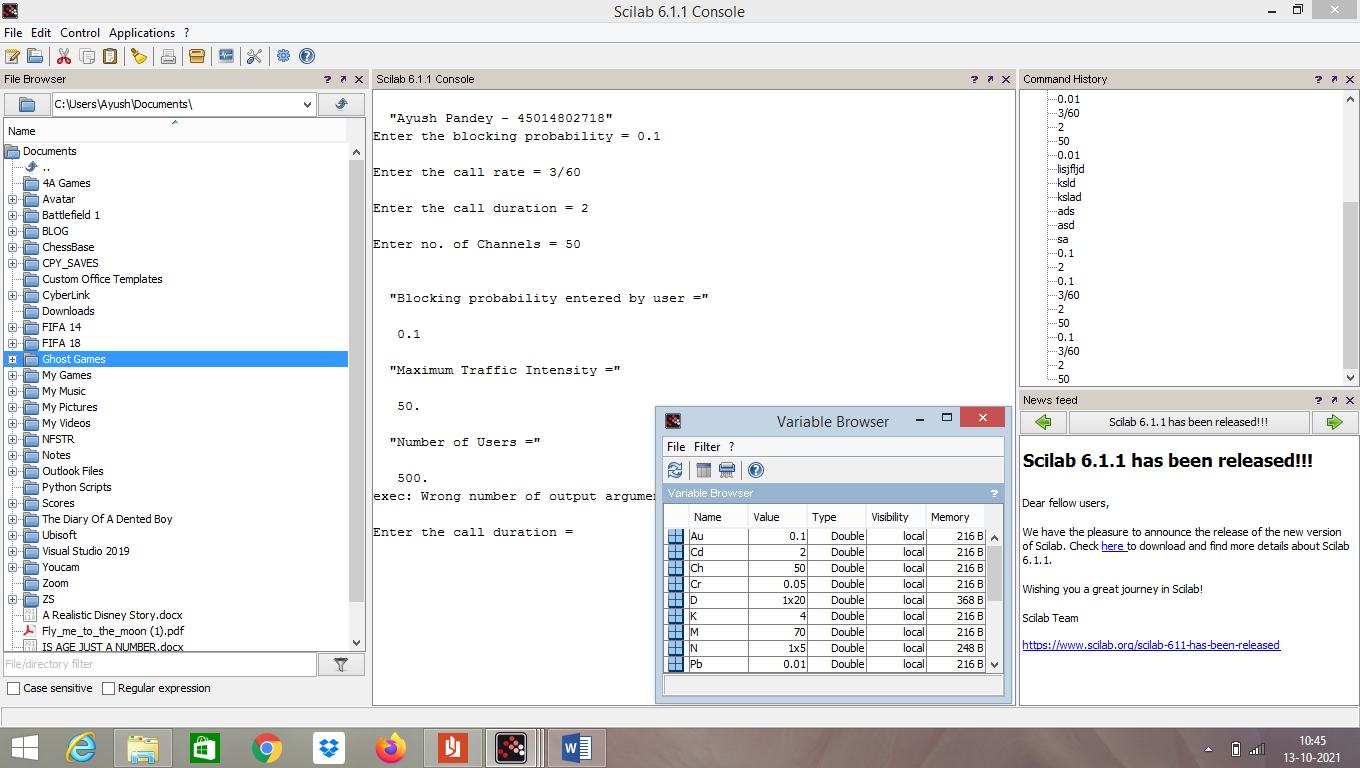
* **Erlang B**

****

****

* **Erlang C**

****

****

**Date: 14/10/2021**

**Experiment-3**

**Aim:** Write a Program in Scilab to calculate Bit Error rate performance of BPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both.

**Code:**

clc;

clear;

n=10000;

data\_stream=grand(1,n,"uin",0,1);

bpsk\_stream=2\*data\_stream-1; snr=1:20;

l=length(snr);

s\_AWGN=0;

s\_AWGN\_Ray=0;

biterror\_AWGN=[];

biterror\_AWGN\_Rayleigh=[];

for k=1:1:l

h=1/sqrt(2)\*(rand(1,n,'normal')+%i\*(rand(1,n,'normal')));

noise=1/sqrt(2)\*(10^((k/20)))\*(rand(1,length(bpsk\_stream),'normal')+%i\*(rand(1,length(bpsk\_stream),'normal')));

s\_AWGN=s\_AWGN+noise;

s\_AWGN\_RAY=s\_AWGN.\*h+noise;

received\_signal=conj(h).\*s\_AWGN\_RAY;

recdata\_AWGN=[]; recdata\_AWGN\_Rayleigh=[];

for i=1:1:n

if (real(s\_AWGN(i))>=0)

output\_AWGN=1;

else

output\_AWGN=0; end

recdata\_AWGN(i) = output\_AWGN; end

for i=1:1:n

if (real(s\_AWGN\_RAY(i))>=0)

output\_AWGN\_Rayleigh=1;

else

output\_AWGN\_Rayleigh=0; end

recdata\_AWGN\_Rayleigh(i) = output\_AWGN\_Rayleigh; end

err\_AWGN = 0;

err\_AWGN\_Rayleigh = 0;

for i=1:1:n

if recdata\_AWGN[i] ~= bpsk\_stream(i)

err\_AWGN = err\_AWGN + 1;

end

end

for i=1:1:n

if recdata\_AWGN\_Rayleigh(i) ~= bpsk\_stream(i)

err\_AWGN\_Rayleigh = err\_AWGN\_Rayleigh + 1;

end end

biterror\_AWGN(k) = err\_AWGN/n;

biterror\_AWGN\_Rayleigh(k) = err\_AWGN\_Rayleigh/n;

end

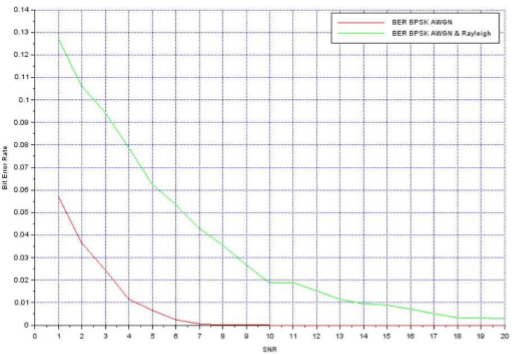
subplot(2,1,1);

plot(snr,biterror\_AWGN); xgrid()

subplot(2,1,2);

plot(snr,biterror\_AWGN\_Rayleigh); xgrid()

**OUTPUT:**

****

**Date: 21/10/2021**

**Experiment-4**

**Aim:** Program in Scilab to Generate Walsh Codes and then spread the user information using it.

**Code:**

clc;

disp("Ayush Pandey : 45014802718");

w = [0 0;0 1];

disp('original walsh code matrix = ',w);

function [​w\_inv​]=​compliment​(​w​)

for i=1:1:length(​w​(1,:))

for j=1:1:length(​w​(1,:))

if ​w​(i,j)== 0

​w\_inv​(i,j)=1;

else

​w\_inv​(i,j)=0;

end

end

end

endfunction

comp = ​compliment​(w);

disp('compliment of walsh code',comp);

w = [w w; w comp];

disp("New Matrix",w);

len=length(w(2,:));

disp("Length of new matrix :",len);

input1 = [1 0 0 1 0];

input2 = [0 1 1 1 0];

input3 = [1 0 1 1 0];

disp("input1",input1);

disp("input2",input2);

disp("input3",input3);

Wcode1 = w(2,:);

Wcode2 = w(3,:);

Wcode3 = w(4,:);

spread = []

spread1 = []

spread2 = []

spread3 = []

for i=1:1:length(input1)

for j=1:1:length(Wcode1)

variable\_xor1 = ​bitxor​(input1(1,i),Wcode1(1,j));

spread1 = [spread1 variable\_xor1];

end

end

disp("Code1 Spread",spread1);

for i=1:1:length(input2)

for j=1:1:length(Wcode2)

variable\_xor2 = ​bitxor​(input2(1,i),Wcode2(1,j));

spread2 = [spread2 variable\_xor2];

end

end

disp("Code2 Spread",spread2);

for i=1:1:length(input3)

for j=1:1:length(Wcode3)

variable\_xor3 = ​bitxor​(input3(1,i),Wcode3(1,j));

spread3 = [spread3 variable\_xor3];

end

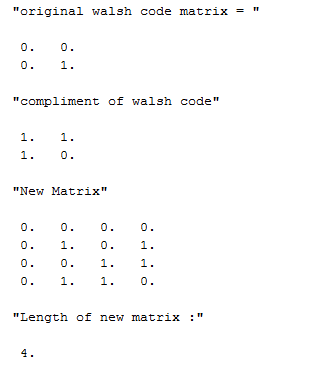
end

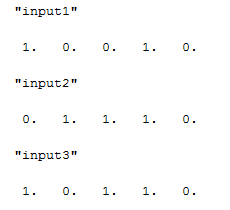
disp("Code3 Spread",spread3);

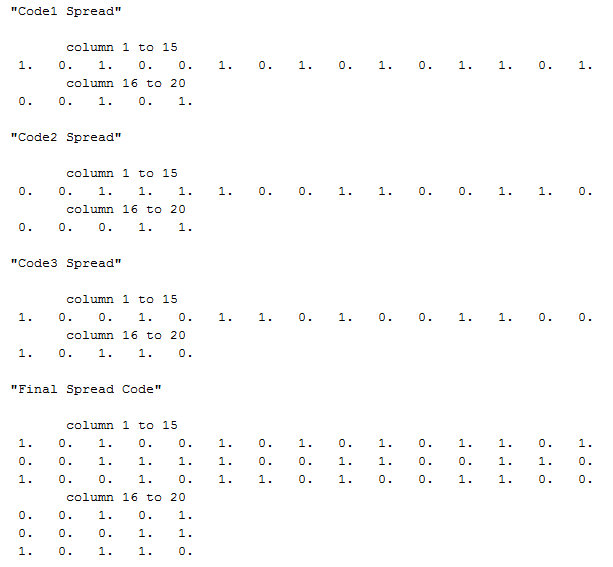
spread = [spread1;spread2;spread3];

disp("Final Spread Code",spread);

**OUTPUT:**

****

****

****

**Date: 28/10/2021**

**Experiment-5**

**Aim:** Program in Scilab to Generate PN Sequence for CDMA Systems.

**Code:**

clc;

disp("Ayush Pandey : 45014802718")

r(1)= 1; r(2)= 0; r(3)= 1; r(4)= 0;

R = [r(1) r(2) r(3) r(4)];

PN = [];

len = length(r);

disp('lemgth of input',len);

disp('initial bit pattern of flip flops',R);

for i=1:1:((2^len)-1)

temp1 = r(1); temp2 = r(2); temp3 = r(3); temp4 = r(4);

PN = [PN r(4)];

temp1 = bitxor(temp3,temp4);

r(4) = r(3); r(3) = r(2); r(2) = r(1); r(1) = temp1;

R = [r(1) r(2) r(3) r(4)];

disp('current bit pattern of flip flops',R);

end

disp('15 bit pattern',PN);

for i=1:1:((2^len)-1)

if(PN(i)==0)

PN(i) = -1;

end

end

disp('After replacing 0 with -1 the 15 bit pattern',PN);

info = [1 -1 -1 1];

leninfo = length(info);

disp("length of data ",length(info));

spread = [];

for i=1:1:leninfo

for j=1:1:length(PN)

x = info(1,i)\*PN(1,j);

spread = [spread x];

end

end

disp("spread ",spread);

len\_spread = length(spread);

disp("length of spreaded data", len\_spread);

PN = [PN PN PN PN];

disp("Updated PN : ",PN);

despread = [];

for i=1:1:length(spread)

x = spread(1,i)\*PN(1,i);

despread = [despread x];

end

disp("multiplied output",despread);

sum\_col = [sum(despread(1:15)) sum(despread(16:30)) sum(despread(31:45)) sum(despread(46:60))];

disp("sum total",sum\_col);

received\_signal = [];

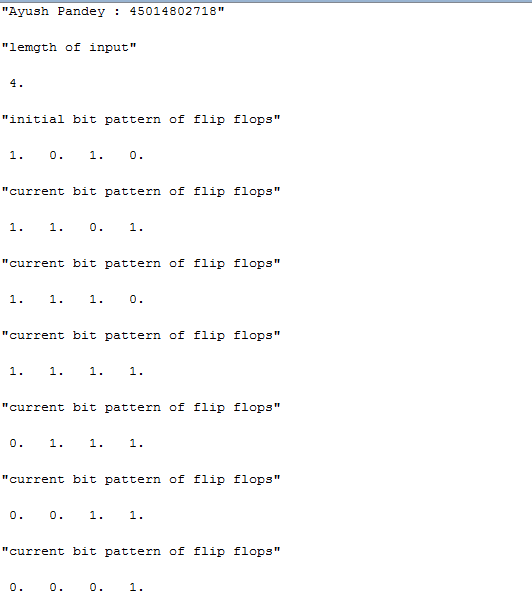
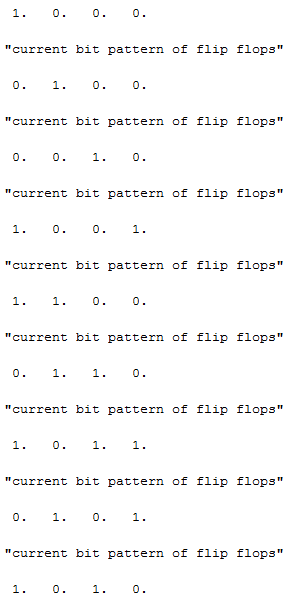
for i=1:1:length(sum\_col)

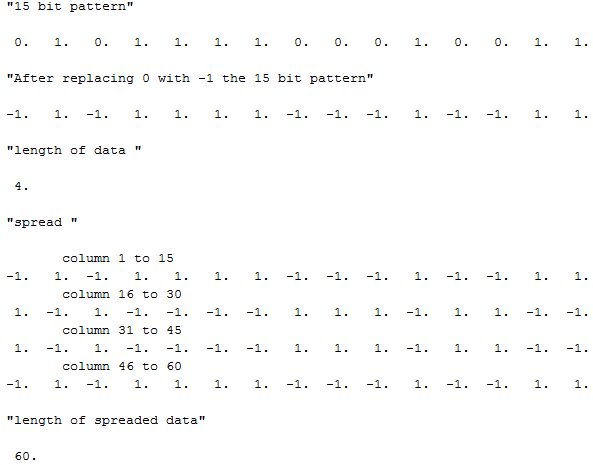
received\_signal =[received\_signal sum\_col(i)/15];

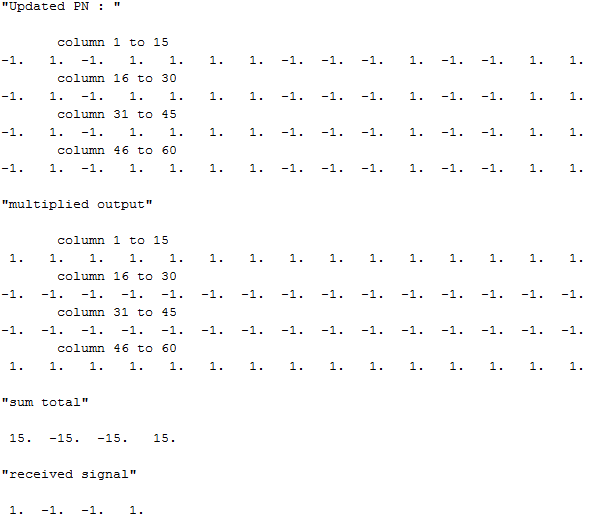
end

disp("received signal",received\_signal);

**OUTPUT:**

****



****

**Date: 04/11/2021**

**Experiment-6**

**Aim:** Write a Program in NS3 to connect WIFI TO BUS (CSMA) Network

**Code:**

#include "ns3/core-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/network-module.h"

#include "ns3/applications-module.h"

#include "ns3/mobility-module.h"

#include "ns3/csma-module.h"

#include "ns3/internet-module.h"

#include "ns3/yans-wifi-helper.h"

#include "ns3/ssid.h"

using namespace ns3;

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

int main (int argc, char \*argv[]){

bool verbose = true;

uint32\_t nCsma = 3;

uint32\_t nWifi = 3;

bool tracing = false;

CommandLine cmd (\_\_FILE\_\_);

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.AddValue ("tracing", "Enable pcap tracing", tracing);

cmd.Parse (argc,argv);

if (nWifi > 18){

std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" << std::endl;

return 1;

}

if (verbose){

LogComponentEnable ("UdpEchoClientApplication", LOG\_LEVEL\_INFO);

LogComponentEnable ("UdpEchoServerApplication", LOG\_LEVEL\_INFO);

}

NodeContainer p2pNodes;

p2pNodes.Create (2);

PointToPointHelper pointToPoint;

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

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

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

NodeContainer csmaNodes;

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

csmaNodes.Create (nCsma);

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));

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

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;

wifiStaNodes.Create (nWifi);

NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();

YansWifiPhyHelper phy;

phy.SetChannel (channel.Create ());

WifiHelper wifi;

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

WifiMacHelper mac;

Ssid ssid = Ssid ("ns-3-ssid");

mac.SetType ("ns3::StaWifiMac","Ssid", SsidValue (ssid),"ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;

staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac","Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;

apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator","MinX", DoubleValue (0.0),"MinY", DoubleValue (0.0),"DeltaX", DoubleValue (5.0),"DeltaY", DoubleValue (10.0),"GridWidth", UintegerValue (3),"LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel","Bounds", RectangleValue(Rectangle (-50, 50, -50, 50))); mobility.Install (wifiStaNodes);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility.Install (wifiApNode);

InternetStackHelper stack;

stack.Install (csmaNodes);

stack.Install (wifiApNode);

stack.Install (wifiStaNodes);

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer p2pInterfaces;

p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer csmaInterfaces;

csmaInterfaces = address.Assign (csmaDevices);

address.SetBase ("10.1.3.0", "255.255.255.0");

address.Assign (staDevices);

address.Assign (apDevices);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue (1));

echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));

echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps =

echoClient.Install (wifiStaNodes.Get (nWifi - 1));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Stop (Seconds (10.0));

if (tracing == true){

pointToPoint.EnablePcapAll ("third");

phy.EnablePcap ("third", apDevices.Get (0));

csma.EnablePcap ("third", csmaDevices.Get (0), true);

}

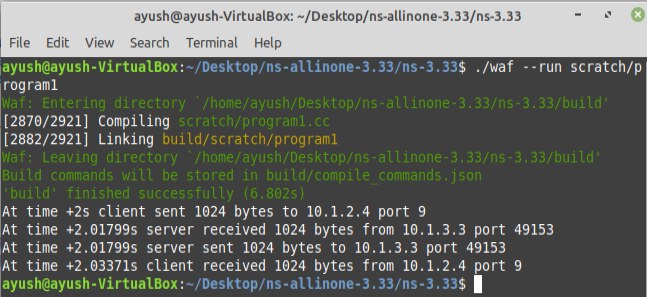
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT:**

****

**Date: 11/11/2021**

**Experiment-7**

**Aim:** Write a Program in NS3 to create WIFI Network in SIMPLE INFRASTUCTURE MODE (of nodes).

**Code:**

#include "ns3/command-line.h"

#include "ns3/config.h"

#include "ns3/double.h"

#include "ns3/string.h"

#include "ns3/log.h"

#include "ns3/yans-wifi-helper.h"

#include "ns3/ssid.h"

#include "ns3/mobility-helper.h"

#include "ns3/ipv4-address-helper.h"

#include "ns3/yans-wifi-channel.h"

#include "ns3/mobility-model.h"

#include "ns3/internet-stack-helper.h"

using namespace ns3;

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

void ReceivePacket (Ptr<Socket> socket){

while (socket->Recv ()){

NS\_LOG\_UNCOND ("Received one packet!");}}

static void GenerateTraffic (Ptr<Socket> socket, uint32\_t pktSize,

uint32\_t pktCount, Time pktInterval ){

if (pktCount > 0){

socket->Send (Create<Packet> (pktSize));

Simulator::Schedule (pktInterval, &GenerateTraffic,

socket, pktSize,pktCount - 1, pktInterval);}

else{

socket->Close ();

}}

int main (int argc, char \*argv[]){

std::string phyMode ("DsssRate1Mbps");

double rss = -80;

uint32\_t packetSize = 1000;

uint32\_t numPackets = 1;

double interval = 1.0;

bool verbose = false;

CommandLine cmd (\_\_FILE\_\_);

cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);

cmd.AddValue ("rss", "received signal strength", rss);

cmd.AddValue ("packetSize", "size of application packet sent", packetSize);

cmd.AddValue ("numPackets", "number of packets generated", numPackets);

cmd.AddValue ("interval", "interval (seconds) between packets", interval);

cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);

cmd.Parse (argc, argv);

Time interPacketInterval = Seconds (interval);

Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",

StringValue (phyMode));

NodeContainer c;

c.Create (2);

WifiHelper wifi;

if (verbose){

wifi.EnableLogComponents ();

}

wifi.SetStandard (WIFI\_STANDARD\_80211b);

YansWifiPhyHelper wifiPhy;

wifiPhy.Set ("RxGain", DoubleValue (0));

wifiPhy.SetPcapDataLinkType (WifiPhyHelper::DLT\_IEEE802\_11\_RADIO);

YansWifiChannelHelper wifiChannel;

wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");

wifiChannel.AddPropagationLoss ("ns3::FixedRssLossModel","Rss",DoubleValue (rss));

wifiPhy.SetChannel (wifiChannel.Create ()); WifiMacHelper wifiMac; wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",

"DataMode",StringValue (phyMode),"ControlMode",StringValue (phyMode));

Ssid ssid = Ssid ("wifi-default");

wifiMac.SetType ("ns3::StaWifiMac","Ssid", SsidValue (ssid));

NetDeviceContainer staDevice = wifi.Install (wifiPhy, wifiMac, c.Get (0));

NetDeviceContainer devices = staDevice;

wifiMac.SetType ("ns3::ApWifiMac","Ssid", SsidValue (ssid));

NetDeviceContainer apDevice = wifi.Install (wifiPhy, wifiMac, c.Get (1));

devices.Add (apDevice);

MobilityHelper mobility;

Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();

positionAlloc->Add (Vector (0.0, 0.0, 0.0));

positionAlloc->Add (Vector (5.0, 0.0, 0.0));

mobility.SetPositionAllocator (positionAlloc);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility.Install (c);

InternetStackHelper internet;

internet.Install (c);

Ipv4AddressHelper ipv4;

NS\_LOG\_INFO ("Assign IP Addresses.");

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (devices);

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");

Ptr<Socket> recvSink = Socket::CreateSocket (c.Get (0), tid);

InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);

recvSink->Bind (local);

recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

Ptr<Socket> source = Socket::CreateSocket (c.Get (1), tid);

InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"), 80);

source->SetAllowBroadcast (true);

source->Connect (remote);

wifiPhy.EnablePcap ("wifi-simple-infra", devices);

NS\_LOG\_UNCOND ("Testing " << numPackets << " packets sent with receiver rss " << rss );

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),

Seconds (1.0), &GenerateTraffic,

source, packetSize, numPackets, interPacketInterval);

Simulator::Stop (Seconds (30.0));

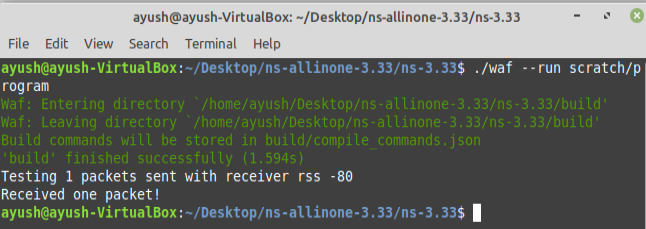
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT:**

****

**Date: 18/11/2021**

**Experiment-8**

**Aim:** Write a Program in NS3 to create a wireless mobile ad-hoc network between three nodes.

**Code:**

#include "ns3/command-line.h"

#include "ns3/config.h"

#include "ns3/double.h"

#include "ns3/string.h"

#include "ns3/log.h"

#include "ns3/yans-wifi-helper.h"

#include "ns3/mobility-helper.h"

#include "ns3/ipv4-address-helper.h"

#include "ns3/yans-wifi-channel.h"

#include "ns3/mobility-model.h"

#include "ns3/internet-stack-helper.h"

using namespace ns3;

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

void ReceivePacket (Ptr<Socket> socket){

while (socket->Recv ()){

NS\_LOG\_UNCOND ("Received one packet!");}}

static void GenerateTraffic (Ptr<Socket> socket, uint32\_t pktSize,

uint32\_t pktCount, Time pktInterval ){

if (pktCount > 0){

socket->Send (Create<Packet> (pktSize));

Simulator::Schedule (pktInterval, &GenerateTraffic,

socket, pktSize,pktCount - 1, pktInterval);}

else{

socket->Close ();}}

int main (int argc, char \*argv[]){

std::string phyMode ("DsssRate1Mbps");

double rss = -80; // -dBm

uint32\_t packetSize = 1000; // bytes

uint32\_t numPackets = 1;

double interval = 1.0; // seconds

bool verbose = false;

CommandLine cmd (\_\_FILE\_\_);

cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);

cmd.AddValue ("rss", "received signal strength", rss);

cmd.AddValue ("packetSize", "size of application packet sent", packetSize);

cmd.AddValue ("numPackets", "number of packets generated", numPackets);

cmd.AddValue ("interval", "interval (seconds) between packets", interval);

cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);

cmd.Parse (argc, argv);

Time interPacketInterval = Seconds (interval);

Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",

StringValue (phyMode));

NodeContainer c;

c.Create (2);

WifiHelper wifi;

if (verbose){

wifi.EnableLogComponents (); // Turn on all Wifi logging

}

wifi.SetStandard (WIFI\_STANDARD\_80211b);

YansWifiPhyHelper wifiPhy;

wifiPhy.Set ("RxGain", DoubleValue (0) );

wifiPhy.SetPcapDataLinkType (WifiPhyHelper::DLT\_IEEE802\_11\_RADIO);

YansWifiChannelHelper wifiChannel;

wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");

wifiChannel.AddPropagationLoss ("ns3::FixedRssLossModel","Rss",DoubleValue (rss));

wifiPhy.SetChannel (wifiChannel.Create ());

WifiMacHelper wifiMac;

wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager","DataMode",StringValue (phyMode),"ControlMode",StringValue (phyMode));

wifiMac.SetType ("ns3::AdhocWifiMac");

NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, c);

MobilityHelper mobility;

Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();

positionAlloc->Add (Vector (0.0, 0.0, 0.0));

positionAlloc->Add (Vector (5.0, 0.0, 0.0));

mobility.SetPositionAllocator (positionAlloc);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility.Install (c);

InternetStackHelper internet;

internet.Install (c);

Ipv4AddressHelper ipv4;

NS\_LOG\_INFO ("Assign IP Addresses.");

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (devices);

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");

Ptr<Socket> recvSink = Socket::CreateSocket (c.Get (0), tid);

InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);

recvSink->Bind (local);

recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

Ptr<Socket> source = Socket::CreateSocket (c.Get (1), tid);

InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"), 80);

source->SetAllowBroadcast (true);

source->Connect (remote);

wifiPhy.EnablePcap ("wifi-simple-adhoc", devices);

NS\_LOG\_UNCOND ("Testing " << numPackets << " packets sent with receiver rss " << rss );

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),Seconds (1.0), &GenerateTraffic, source, packetSize, numPackets, interPacketInterval);

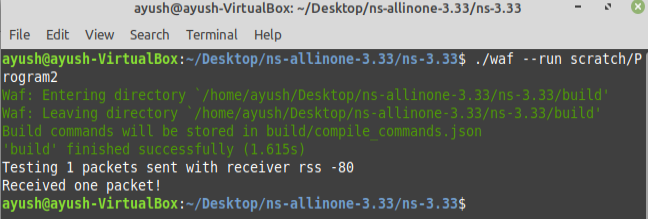
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT:**

****