

# **SRM UNIVERSITY DELHI-NCR, SONEPAT, HARYANA**

**Plot No.39, Rajiv Gandhi Education City, P.S. Rai, Sonapat, Haryana – 131029**

(Established under Haryana Private University Act, 2006 as amended by Act No. 8 of 2013)



## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **MOBILE COMPUTING LAB (CS-4112) LABORATORY RECORD**

<b>STUDENT NAME</b>	<b>Adeep Sri Narayana</b>
<b>REGISTRATION NUMBER</b>	<b>11018210002</b>
<b>YEAR/ SEMESTER</b>	<b>IV Year / VIII Semester</b>
<b>DEPARTMENT / PROGRAMME</b>	<b>B. Tech Computer Science and Engineering</b>
<b>PROGRAMME SECTION</b>	<b>Data Science and AI/Section - C</b>

## **Bonafide Certificate**

Student Name	<b>Adeep Sri Narayana</b>
Registration No	<b>11018210002</b>
Year/Semester	<b>IV year/ VIII Semester</b>
Department/ Programme	<b>B. Tech Computer Science and Engineering</b>
Programme Section	<b>Data Science and AI/Section - C</b>
Subject code/Subject Name	<b>CS 4112 Mobile Computing Lab</b>

Certified that this is the Bonafide record of practical work done by the aforesaid student in the **CS 4112 Mobile Computing Lab** during the academic year **2021 -2022**.

**Faculty (In-charge)**

**(Head of the Department)**

Submission for the Practical Examination held on.....

Examiner

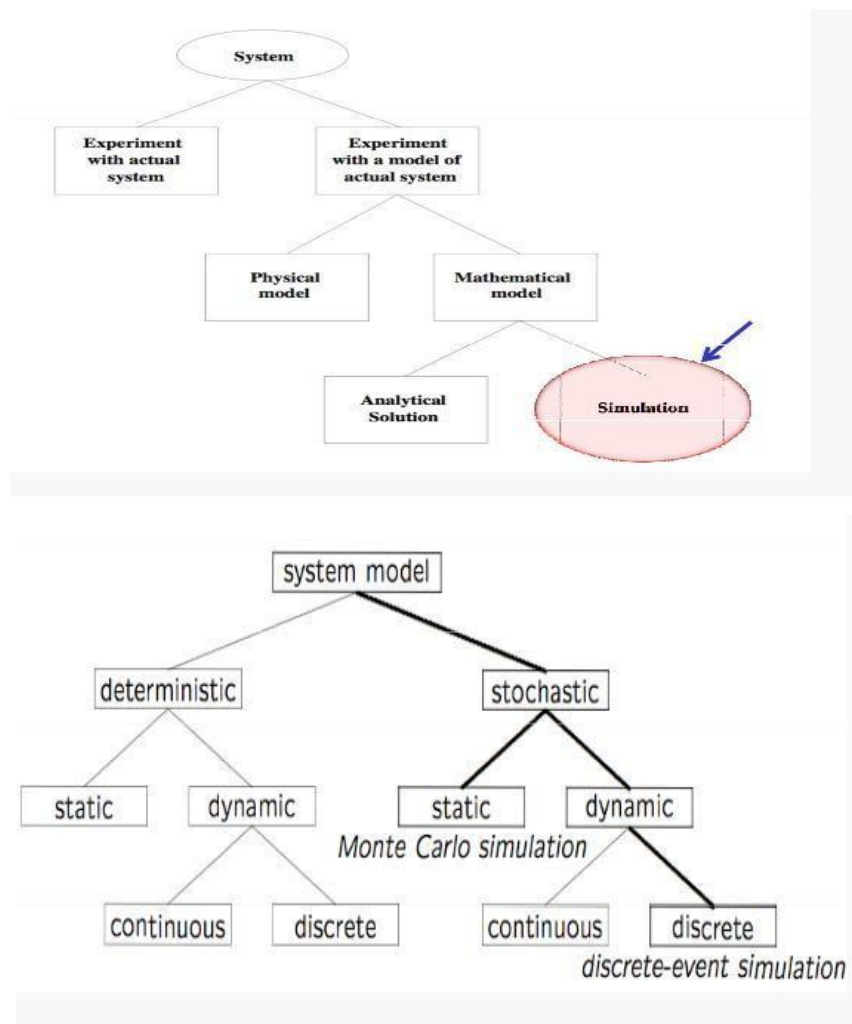
## **Table of Contents**

<b>Experiment No.</b>	<b>Date</b>	<b>Name of the Experiment</b>	<b>Page No.</b>
1.	07-03-2022	Introduction to discrete event simulation	4-5
2.	14-03-2022	Installation of ns3	6-7
3.	25-03-2022	To connect two nodes in ns3	8-9
4.	01-04-2022	To install NetAnim to simulate two nodes in ns3	10-11
5.	08-04-2022	To implement star topology in ns3	12-14
6.	06-05-2022	To implement bus topology in ns3	15-17
7.	13-05-2022	To implement AODV algorithm	18-22
8.	20-05-2022	To implement DSR algorithm	23-31

**Faculty (In-charge)**

### Discrete System

State variables change instantaneously at a separate point in time, e.g., a bank, since state variables - number of customers, change only when a customer arrives or when a customer finish being served and departs.



### Discrete Event Simulation

A discrete-event simulation models a system whose state may change only at discrete point models a system whose state may change only at discrete point in time.

**System:**

is composed of objects called entities that have certain properties called attributes.

**State:**

a collection of attributes or state variables that represent the entities of the system.

**Event:**

an instantaneous occurrence in time that may alter the state of the system.

**Simulation**

Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of a system.

**Need for Simulation**

- Many systems are highly complex, precluding the possibility of analytical solutions.
- The analytical solutions are extraordinarily complex, requiring vast computing resources.
- Thus, such systems should be studied by means of simulation numerically exercising the model for inputs in question to see how they affect the output measures of performance.

NS3 is a powerful network simulation program whose simulations are built on C++ It is available on [www.nsnam.org](http://www.nsnam.org).

## Steps to install NS3

Following are the basic steps to be completed in order to install ns3 on Ubuntu virtual machine.

1. Install prerequisite packages
2. Download ns3 codes
3. Build ns3
4. Validate ns3

## Detailed steps are as follows

1. `sudo apt-get update / dnf update`
2. `sudo apt-get upgrade / dnf upgrade`
3. Once ubuntu/fedora is installed run the following command opening the terminal(ctrl+alt+T) window.
4. To install prerequisite dependency packages- Type the following command in the terminal window.

```
sudo apt-get/ dnf install gcc g++ python python-dev mercurial bzr gdb valgrind gsl-bin  
libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev  
libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphviz imagemagick texlive texlive-latex-  
extra texlive-generic-extra texlive-generic-recommended texinfo dia texlive texlive-latex-  
extra texlive-extra- utils texlive-generic-recommended texi2html python-pygraphviz python-  
kiwi python- pygoocanvas libgoocanvas-dev python-pygccxml
```

After downloading NS3 on the drive, extract all the files in the NS3 folder,

which you have created.

5. Then you can find build.py along with other files in the NS3 folder. Then to build the examples in ns-3 run:

```
./build.py --enable-examples --enable-tests
```

If the build is successful then it will give output "Build finished successfully".

6. Now run the following command on the terminal window, to configure with waf(build tool)

```
./waf -d debug --enable-examples --enable-tests configure
```

To build with waf(optional)

```
./waf
```

7. To test everything alright run the following command on the terminal window,

```
./test.py
```

If the tests are ok the installation is done.

8. Now after installing ns3 and testing it run some programs first to be ns3 user:  
make sure you are in directory where waf script is available then run.

## Output:

The screenshot shows the ns-3 website's releases page for version 3.36. The page has a dark header with the ns-3 logo and navigation links: About, Consortium, Research, Education, Documentation, Develop, Support, and a search bar. A left sidebar lists releases from ns-3.36 down to ns-3.20, with ns-3.36 highlighted. The main content area for ns-3.36 includes a breadcrumb trail (Home > Releases > ns-3.36), the version number 'ns-3.36', and a note about a minor release update to ns-3.36.1. It details the release date (April 30, 2022) and update date (May 23, 2022), mentioning contributions from thirty authors and changes to the build system (Waf to CMake) and propagation loss model. A 'Download' section provides a link to the source archive and explains it contains additional tools (bake, netanim, pybindgen). A 'Documentation' section lists various formats and links to changes, release notes, and upgrade patches. At the bottom, a file manager window shows the downloaded file 'ns-allinone-3.36.tar.bz2' with a size of 6.9/41.6 MB and 3 minutes left.

ns-3

About Consortium Research Education Documentation Develop Support Search

Releases

ns-3.36

Authors

Documentation

Download

ns-3.35

ns-3.34

ns-3.33

ns-3.32

ns-3.31

ns-3.30

ns-3.29

ns-3.28

ns-3.27

ns-3.26

ns-3.25

ns-3.24

ns-3.23

ns-3.22

ns-3.21

ns-3.20

Older releases

Home > Releases > ns-3.36

## ns-3.36

**Note:** Please see below about the ns-3.36.1 minor release update of ns-3.36.

**ns-3.36** was released on April 30, 2022, and updated by **ns-3.36.1** on May 23, 2022, due to contributions from [thirty authors](#). This release includes a change of the ns-3 build system from [Waf](#) to [CMake](#). A new phased array spectrum propagation loss model has been added to better support different 4G/5G MIMO models that can exploit the multiple subarray concept, and multiple PhasedArray antenna models can be supported per device. LTE handover now works with carrier aggregation configurations. This was mainly a maintenance and bug-fixing release cycle for the Wi-Fi module, but the default Wi-Fi standard has been upgraded to 802.11ax. Many additional improvements and bug fixes are listed in the [RELEASE\\_NOTES](#).

## Download

The ns-3.36.1 release download is available from [this link](#). This download is a source archive that contains some additional tools (bake, netanim, pybindgen) in addition to the ns-3.36.1 source. The ns-3 source code by itself can also be checked out of our Git repository by referencing the tag 'ns-3.36.1'.

## Documentation

The documentation is available in several formats from [this link](#).

- What has changed since ns-3.35? Consult the [changes](#) and [RELEASE\\_NOTES](#) pages for ns-3.36.
- What has changed between ns-3.36 and the ns-3.36.1 release? Consult the [changes](#) and [RELEASE\\_NOTES](#) pages for ns-3.36.1.
- A patch to upgrade from ns-3.35 to ns-3.36 can be found [here](#)
- A patch to upgrade from ns-3.36 to ns-3.36.1 can be found [here](#)
- Errata containing any late-breaking information about the release can be found [here](#)

ns-allinone-3.36.tar.bz2  
6.9/41.6 MB, 3 mins left

Show all

**Source Code –**

```
#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

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

{

    Time::SetResolution (Time::NS);

    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO); LogComponentEnable
("UdpEchoServerApplication", LOG_LEVEL_INFO);

    NodeContainer nodes; nodes.Create (2);

    PointToPointHelper pointToPoint;

    pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

    NetDeviceContainer devices;

    devices = pointToPoint.Install (nodes);

    InternetStackHelper stack; stack.Install (nodes);

    Ipv4AddressHelper address;

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

    Ipv4InterfaceContainer interfaces = address.Assign (devices); UdpEchoServerHelper echoServer (9);
```



```

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1)); serverApps.Start (Seconds
(1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9); echoClient.SetAttribute
("MaxPackets", UIntegerValue (1)); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (nodes.Get (0)); clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Simulator::Run ();

Simulator::Destroy ();

return 0;

}

```

## Output

```

ns3@ubuntu:~$ cd Desktop
ns3@ubuntu:~/Desktop$ ls
ns-allinnone-3.27  ns-allinnone-3.28  parse_groceries.awk
ns3@ubuntu:~/Desktop$ cd ns-allinnone-3.28
ns3@ubuntu:~/Desktop/ns-allinnone-3.28$ ls
bake          constants.pyc  pybindgen-0.17.0.post58+ngcf00cc0  util.pyc
build.py      netanin-3.100  README
constants.py  ns-3.28       util.py
ns3@ubuntu:~/Desktop/ns-allinnone-3.28$ cd ns-3.28
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28$ ls
AUTHORS      DLTxPhyStats.txt  testpy-output      utils.pyc
bindings     doc               testpy.suppl       VERSION
build         examples          UInterferenceStats.txt  waf
CHANGES.html  LICENSE          ULPdcpStats.txt     waf.bat
contrib       Makefile          ULPdcpStats.txt     waf-tools
different.pcap  README           ULRlcStats.txt      wscript
DLMacStats.txt  RELEASE_NOTES    ULrInrStats.txt     wutils.py
DLPdcpStats.txt  scratch          ULTxPhyStats.txt    wutils.pyc
DRLcStats.txt   src              utils
DLRsrpInrStats.txt  test.py         utils.py
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28$ cd scratch
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28/scratch$ ls
first.cc  myfirst.cc  scratch-simulator.cc  subdir
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28/scratch$ ./waf --run scratch/firstbash: ./waf: No such file or directory
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28/scratch$ ./waf --run scratch/first
bash: ./waf: No such file or directory
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28/scratch$ cd ..
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28$ ./waf --run scratch/first
waf: Entering directory '/home/ns3/Desktop/ns-allinnone-3.28/ns-3.28/build'
waf: Leaving directory '/home/ns3/Desktop/ns-allinnone-3.28/ns-3.28/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (30.905s)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
ns3@ubuntu:~/Desktop/ns-allinnone-3.28/ns-3.28$

```

## Installing NetAnim

To install netanim, install the following prerequisite packages

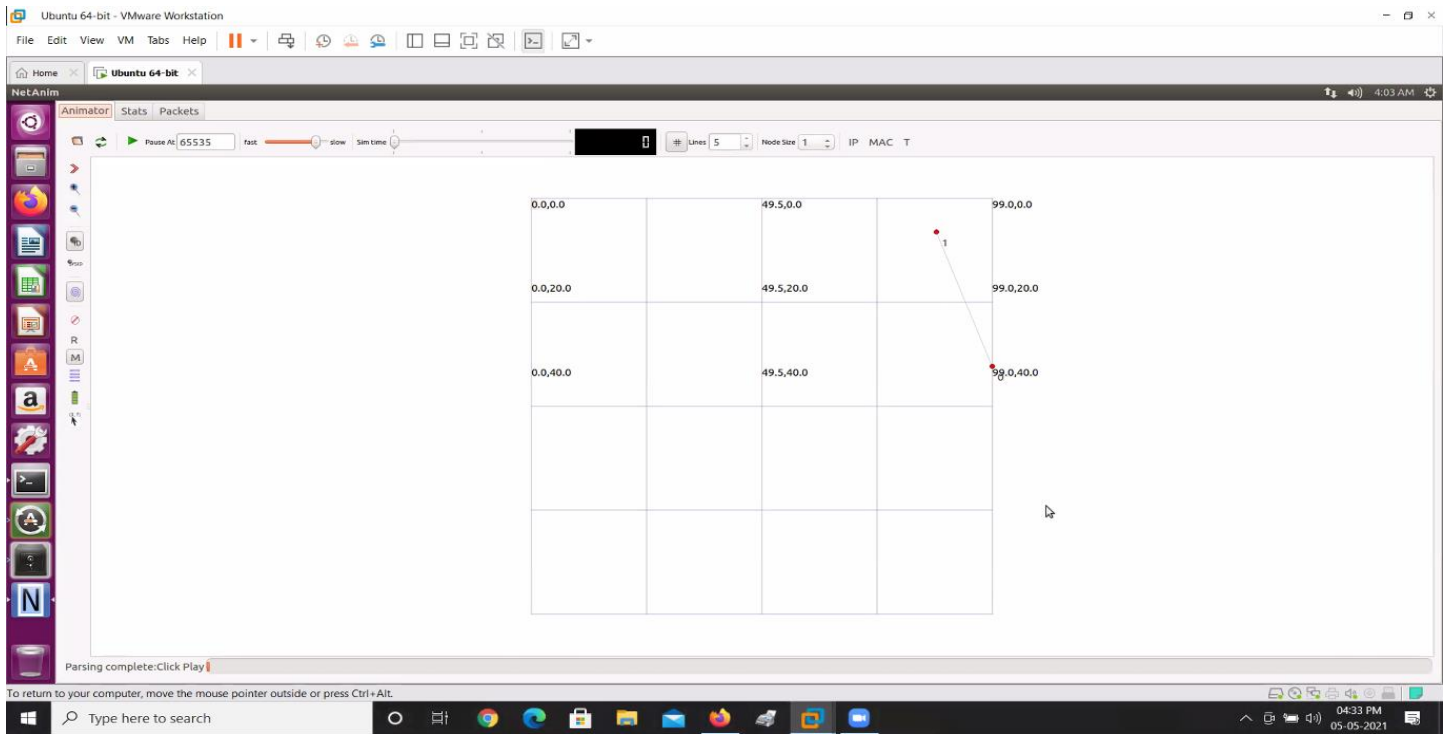
1. Install mercurial –  
apt-get/dnf install mercurial
2. Install qt4 development packages  
Apt-get/dnf install qt4-dev tools
3. Download NetAnim from <http://code.nsnam.org/netanim>
4. Build NetAnim  
cd NetAnim  
make clean qmake  
NetAnim.pro make

Now we have to make the following changes in the first.cc file which are as follows

- Include the following header file along with the other included header files  
#include “netanim-module.h”
- Below just before the **Simulator::Run();** part, include the following code snippet  
AnimationInterface anim(“anim1.xml”);  
Anim.SetConstantPosition(nodes.Get(0), 1.0, 2.0)  
Anim.SetConstantPosition(nodes.Get(1), 2.0, 3.0)
- The number of positions will depend on the number of nodes created, in this case we have created 2 nodes, so two positions will be created.
- Save the changes and run the following file.

## Output

```
Waf: Leaving directory '/home/ns3/Desktop/ns-allinone-3.28/ns-3.28/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (16.069s)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28$ cd scratch
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28/scratch$ ls
first.cc myfirst.cc scratch-simulator.cc subdir
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28/scratch$ gedit myfirst.cc
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28/scratch$ cd .
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28$ ./waf --run scratch/myfirst
Waf: Entering directory '/home/ns3/Desktop/ns-allinone-3.28/ns-3.28/build'
[ 961/2746] Compiling scratch/myfirst.cc
[2725/2746] Linking build/scratch/myfirst
Waf: Leaving directory '/home/ns3/Desktop/ns-allinone-3.28/ns-3.28/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (19.089s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28$ cd
ns3@ubuntu:~$ clear
ns3@ubuntu:~$ cd Desktop
ns3@ubuntu:~/Desktop$ ls
ns-allinone-3.27 ns-allinone-3.28 parse_groceries.awk
ns3@ubuntu:~/Desktop$ cd ns-allinone-3.28
ns3@ubuntu:~/Desktop/ns-allinone-3.28$ ls
bake build.py constants.py constants.pyc netanim-3.108 ns-3.28 pybindgen-0.17.0.post58+ngcf00cc0 README util.py util.pyc
ns3@ubuntu:~/Desktop/ns-allinone-3.28$ cd netanim-3.108
ns3@ubuntu:~/Desktop/ns-allinone-3.28/netanim-3.108$ ./NetAnim
```



**Source Code –**

```
#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/netanim-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Star");

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

{

    Config::SetDefault ("ns3::OnOffApplication::PacketSize", UIntegerValue (137));

    Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("14kb/s"));

    uint32_t nSpokes = 8;

    CommandLineCmd;

    cmd.AddValue ("nSpokes", "Number of nodes to place in the star", nSpokes); cmd.Parse (argc, argv);

    NS_LOG_INFO ("Build star topology.");

    PointToPointHelper pointToPoint;

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

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

    PointToPointStarHelper star (nSpokes, pointToPoint);
```

```

NS_LOG_INFO ("Install internet stack on all nodes.");

InternetStackHelper internet; star.InstallStack (internet);


NS_LOG_INFO ("Assign IP Addresses.");

star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));

NS_LOG_INFO ("Create applications.");

uint16_t port = 50000;

Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port)); PacketSinkHelper
packetSinkHelper ("ns3::TcpSocketFactory", hubLocalAddress); ApplicationContainer hubApp =
packetSinkHelper.Install (star.GetHub ());

hubApp.Start (Seconds (1.0));

hubApp.Stop (Seconds (10.0));

OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ()); onOffHelper.SetAttribute
("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]")); onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));

ApplicationContainer spokeApps;


for (uint32_t i = 0; i < star.SpokeCount (); ++i)
{

AddressValue remoteAddress (InetSocketAddress (star.GetHubIpv4Address (i), port));
onOffHelper.SetAttribute ("Remote", remoteAddress);

spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));

}

spokeApps.Start (Seconds (1.0));

spokeApps.Stop (Seconds (10.0));

NS_LOG_INFO ("Enable static global routing.");

```

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

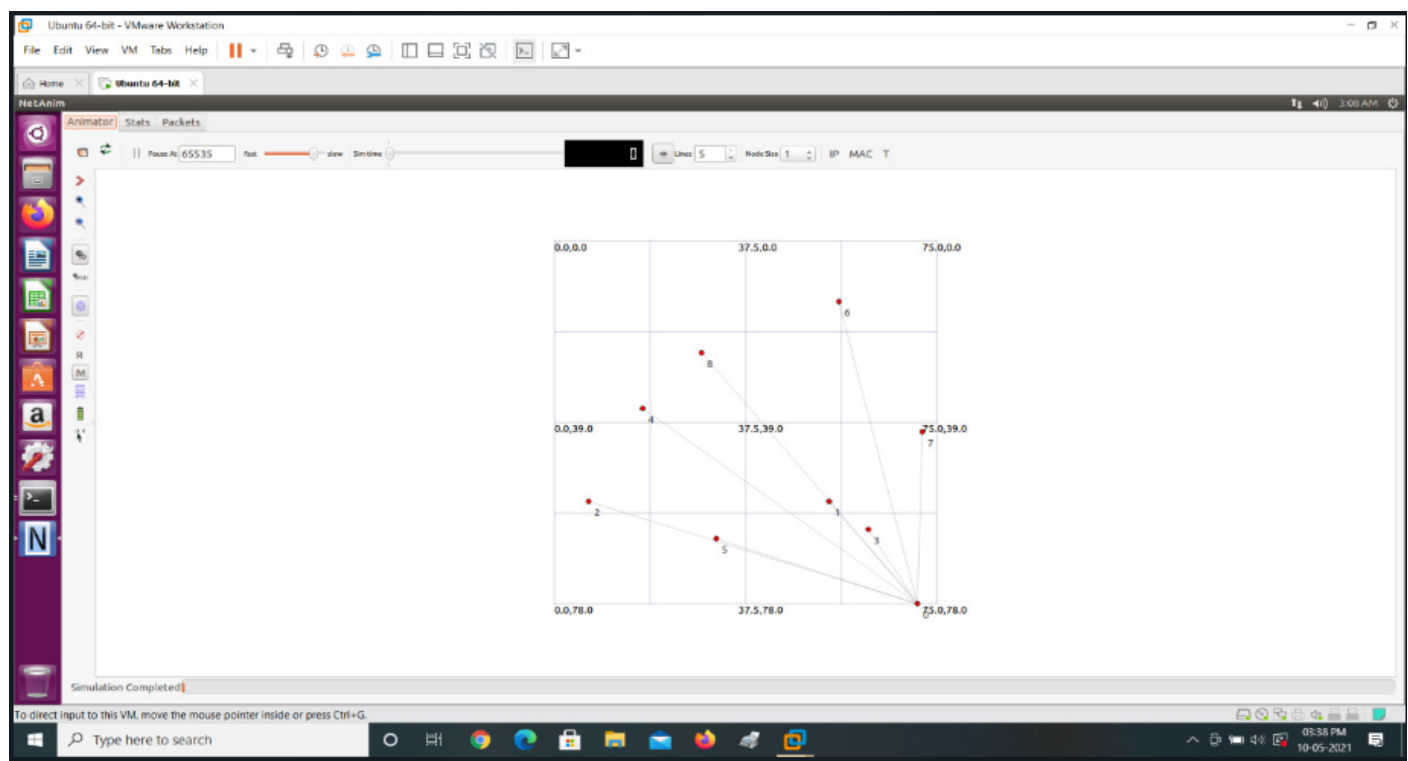
```
NS_LOG_INFO ("Enable pcap tracing.");
```

```
pointToPoint.EnablePcapAll ("star");
```

```
NS_LOG_INFO ("Run Simulation."); Simulator::Run (); Simulator::Destroy (); NS_LOG_INFO  
("Done.");
```

```
return 0;
```

## Output



**Source Code**

```
#include "ns3/core-module.h";

#include "ns3/network-module.h";

#include "ns3/csma-module.h";

#include "ns3/internet-module.h";

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

#include "ns3/applications-module.h";

#include "ns3/ipv4-global-routing-helper.h";

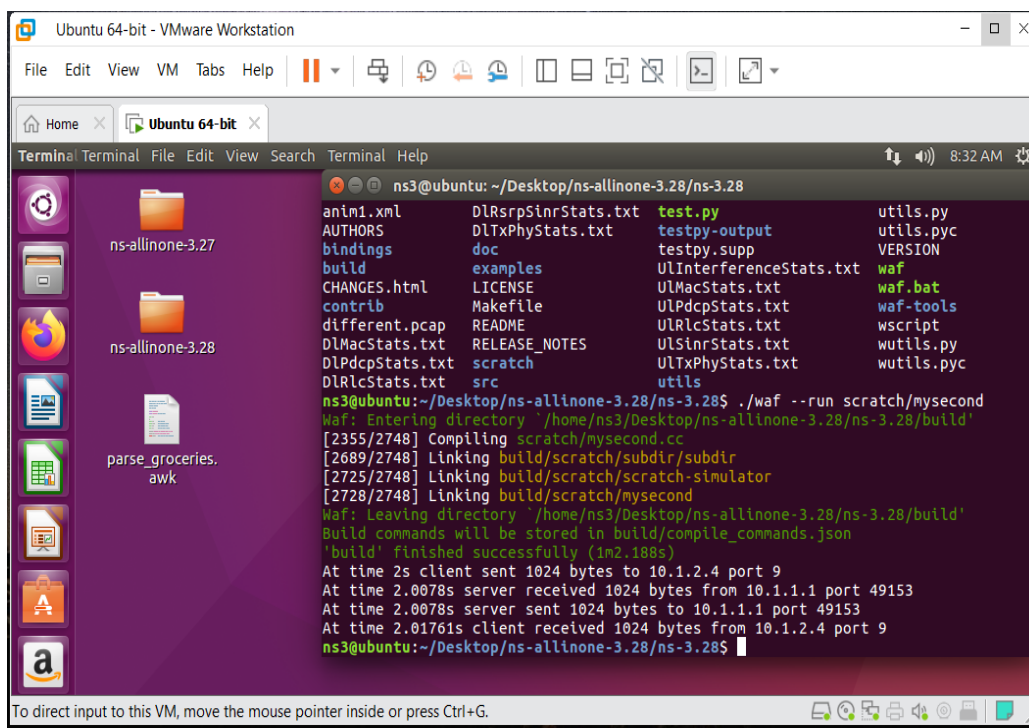
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("SecondScriptExample"); int
main (int argc, char *argv[])
{
bool verbose = true; uint32_t nCsma = 3;
CommandLine cmd;
cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma); cmd.AddValue
("verbose", "Tell echo applications to log if true", verbose); cmd.Parse (argc,argv);
if (verbose)
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO); LogComponentEnable
("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
nCsma = nCsma == 0 ? 1 : nCsma; NodeContainer p2pNodes; p2pNodes.Create (2); NodeContainer
csmaNodes;
csmaNodes.Add (p2pNodes.Get (1)); csmaNodes.Create (nCsma); PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms")); NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes); CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps")); csma.SetChannelAttribute ("Delay",
TimeValue (NanoSeconds (6560))); NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes); InternetStackHelper stack;
stack.Install (p2pNodes.Get (0)); stack.Install (csmaNodes); 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);
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 (p2pNodes.Get (0)); clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0)); Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
pointToPoint.EnablePcapAll ("second"); csma.EnablePcap ("second", csmaDevices.Get (1), true);
Simulator::Run ();
Simulator::Destroy (); return 0;}

```

## Output



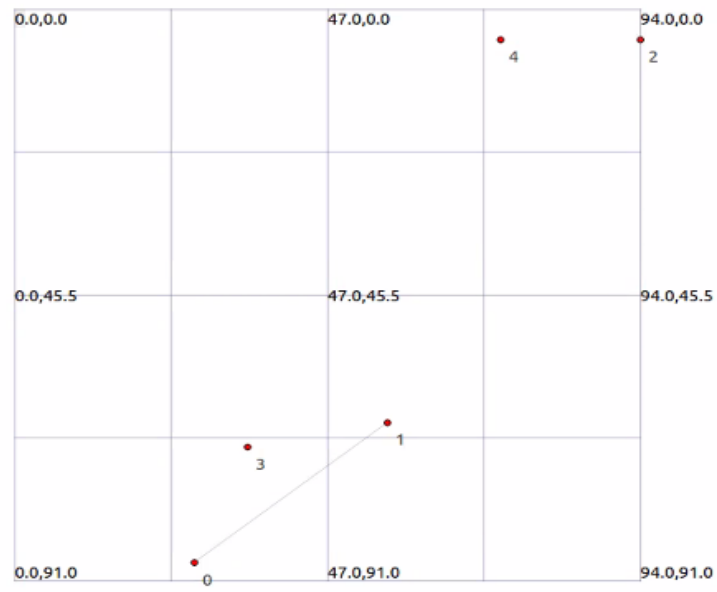
```

ns3@ubuntu: ~/Desktop/ns-allinone-3.28/ns-3.28
anim1.xml      DIRsrpSinrStats.txt  test.py        utils.py
AUTHORS        DLTxPhyStats.txt    testpy-output  utils.pyc
bindings       doc                  testpy.sup     VERSION
build          examples            UInterferenceStats.txt waf
CHANGES.html  LICENSE             ULMacStats.txt  waf.bat
contrib        Makefile            ULPdcpStats.txt waf-tools
different.pcap README              ULRlcStats.txt  wscript
DLMacStats.txt RELEASE_NOTES      ULSinrStats.txt wutils.py
DLPdcpStats.txt scratch          ULTxPhyStats.txt wutils.pyc
DLRlcStats.txt src              utils

ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28$ ./waf --run scratch/mysecond
waf: Entering directory '/home/ns3/Desktop/ns-allinone-3.28/ns-3.28/build'
[2355/2748] Compiling scratch/mysecond.cc
[2689/2748] Linking build/scratch/subdir/subdir
[2725/2748] Linking build/scratch/scratch-simulator
[2728/2748] Linking build/scratch/mysecond
waf: Leaving directory '/home/ns3/Desktop/ns-allinone-3.28/ns-3.28/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1m2.188s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.01761s client received 1024 bytes from 10.1.2.4 port 9
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28$

```





**Source Code –**

```
#include <iostream>
#include <cmath>
#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/v4ping-helper.h"
#include "ns3/yans-wifi-helper.h"

using namespace ns3;

class AodvExample
{
public:
    AodvExample ();

    bool Configure (int argc, char **argv);
    void Run ();
    void Report (std::ostream & os);

private:
    uint32_t size;
    double step;
    double totalTime;
    bool pcap;
    bool printRoutes;

    NodeContainer nodes;
    NetDeviceContainer devices;

    Ipv4InterfaceContainer interfaces;
private:
    void CreateNodes ();
    void CreateDevices ();
    void InstallInternetStack ();
    void InstallApplications ();
};
```

```

int main (int argc, char **argv)
{
    AodvExample test;
    if (!test.Configure (argc, argv))
        NS_FATAL_ERROR ("Configuration failed. Aborted.");

    test.Run ();
    test.Report (std::cout);
    return 0;
}

AodvExample::AodvExample () :
    size (10),
    step (50),
    totalTime (100),
    pcap (true),
    printRoutes (true)
{
}

bool
AodvExample::Configure (int argc, char **argv)
{
    // Enable AODV logs by default. Comment this if too noisy
    // LogComponentEnable("AodvRoutingProtocol", LOG_LEVEL_ALL);

    SeedManager::SetSeed (12345);
    CommandLine cmd;
    cmd.AddValue ("pcap", "Write PCAP traces.", pcap);
    cmd.AddValue ("printRoutes", "Print routing table dumps.", printRoutes);
    cmd.AddValue ("size", "Number of nodes.", size);
    cmd.AddValue ("time", "Simulation time, s.", totalTime);
    cmd.AddValue ("step", "Grid step, m", step);

    cmd.Parse (argc, argv);
    return true;
}

void
AodvExample::Run ()
{
    // Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", UintegerValue (1)); //
    enable rts cts all the time.
    CreateNodes ();
    CreateDevices ();
    InstallInternetStack ();
    InstallApplications ();
}

```

```

std::cout << "Starting simulation for " << totalTime << " s ...\n";

Simulator::Stop (Seconds (totalTime));
Simulator::Run ();
Simulator::Destroy ();
}

void
AodvExample::Report (std::ostream &)
{
}

void
AodvExample::CreateNodes ()
{
    std::cout << "Creating " << (unsigned)size << " nodes " << step << " m apart.\n";
    nodes.Create (size);
    // Name nodes
    for (uint32_t i = 0; i < size; ++i)
    {
        std::ostringstream os;
        os << "node-" << i;
        Names::Add (os.str (), nodes.Get (i));
    }
    // Create static grid
    MobilityHelper mobility;
    mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
        "MinX", DoubleValue (0.0),
        "MinY", DoubleValue (0.0),
        "DeltaX", DoubleValue (step),
        "DeltaY", DoubleValue (0),
        "GridWidth", UIntegerValue (size),
        "LayoutType", StringValue ("RowFirst"));
    mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
    mobility.Install (nodes);
}

void
AodvExample::CreateDevices ()
{
    WifiMacHelper wifiMac;
    wifiMac.SetType ("ns3::AdhocWifiMac");
    YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
    YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
    wifiPhy.SetChannel (wifiChannel.Create ());
    WifiHelper wifi;

```

```

wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate6Mbps"), "RtsCtsThreshold", UIntegerValue (0));
devices = wifi.Install (wifiPhy, wifiMac, nodes);

if (pcap)
{
    wifiPhy.EnablePcapAll (std::string ("aodv"));
}
}
void
AodvExample::InstallInternetStack ()
{
    AodvHelper aodv;
    // you can configure AODV attributes here using aodv.Set(name, value)
    InternetStackHelper stack;
    stack.SetRoutingHelper (aodv); // has effect on the next Install ()
    stack.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.0.0.0", "255.0.0.0");
    interfaces = address.Assign (devices);

    if (printRoutes)
    {
        Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ("aodv.routes",
std::ios::out);
        aodv.PrintRoutingTableAllAt (Seconds (8), routingStream);
    }
}
void
AodvExample::InstallApplications ()
{
    V4PingHelper ping (interfaces.GetAddress (size - 1));
    ping.SetAttribute ("Verbose", BooleanValue (true));

    ApplicationContainer p = ping.Install (nodes.Get (0));
    p.Start (Seconds (0));
    p.Stop (Seconds (totalTime) - Seconds (0.001));

    // move node away
    Ptr<Node> node = nodes.Get (size/2);
    Ptr<MobilityModel> mob = node->GetObject<MobilityModel> ();
    Simulator::Schedule (Seconds (totalTime/3), &MobilityModel::SetPosition, mob, Vector (1e5, 1e5,
1e5));
}

```

# Output

```
ns3@ubuntu: ~/Desktop/ns-allinone-3.28/ns-3.28
64 bytes from 10.0.0.10: icmp_seq=54 ttl=60 time=7 ms
64 bytes from 10.0.0.10: icmp_seq=55 ttl=60 time=7 ms
64 bytes from 10.0.0.10: icmp_seq=56 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=57 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=58 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=59 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=60 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=61 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=62 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=63 ttl=60 time=6 ms
64 bytes from 10.0.0.10: icmp_seq=64 ttl=60 time=8 ms
64 bytes from 10.0.0.10: icmp_seq=65 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=66 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=67 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=68 ttl=60 time=6 ms
64 bytes from 10.0.0.10: icmp_seq=69 ttl=60 time=7 ms
64 bytes from 10.0.0.10: icmp_seq=70 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=71 ttl=60 time=7 ms
64 bytes from 10.0.0.10: icmp_seq=72 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=73 ttl=60 time=6 ms
64 bytes from 10.0.0.10: icmp_seq=74 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=75 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=76 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=77 ttl=60 time=6 ms
64 bytes from 10.0.0.10: icmp_seq=78 ttl=60 time=9 ms
64 bytes from 10.0.0.10: icmp_seq=79 ttl=60 time=6 ms
64 bytes from 10.0.0.10: icmp_seq=80 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=81 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=82 ttl=60 time=7 ms
64 bytes from 10.0.0.10: icmp_seq=83 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=84 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=85 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=86 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=87 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=88 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=89 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=90 ttl=60 time=7 ms
64 bytes from 10.0.0.10: icmp_seq=91 ttl=60 time=6 ms
64 bytes from 10.0.0.10: icmp_seq=92 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=93 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=94 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=95 ttl=60 time=4 ms
64 bytes from 10.0.0.10: icmp_seq=96 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=97 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=98 ttl=60 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=99 ttl=60 time=4 ms
--- 10.0.0.10 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99999ms
rtt min/avg/max/ndev = 4/12.25/712/70.09 ms
ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28$
```

To return to your computer, move the mouse pointer outside or press Ctrl+Alt.

**Source Code –**

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/dsdv-helper.h"
#include <iostream>
#include <cmath>

using namespace ns3;

uint16_t port = 9;

NS_LOG_COMPONENT_DEFINE ("DsdvManetExample");
/**
 * \ingroup dsdv
 * \ingroup dsdv-examples
 * \ingroup examples
 *
 * \brief DSDV Manet example
 */
class DsdvManetExample
{
public:
    DsdvManetExample ();
    /**
     * Run function
     * \param nWifis The total number of nodes
     * \param nSinks The total number of receivers
     * \param totalTime The total simulation time
     * \param rate The network speed
     * \param phyMode The physical mode
     * \param nodeSpeed The node speed
     * \param periodicUpdateInterval The routing update interval
     * \param settlingTime The routing update settling time
     * \param dataStart The data transmission start time
     * \param printRoutes print the routes if true
     * \param CSVfileName The CSV file name
     */
```

```

void CaseRun (uint32_t nWifis,
              uint32_t nSinks,
              double totalTime,
              std::string rate,
              std::string phyMode,
              uint32_t nodeSpeed,
              uint32_t periodicUpdateInterval,
              uint32_t settlingTime,
              double dataStart,
              bool printRoutes,
              std::string CSVfileName);

```

private:

```

uint32_t m_nWifis; ///< total number of nodes
uint32_t m_nSinks; ///< number of receiver nodes
double m_totalTime; ///< total simulation time (in seconds)
std::string m_rate; ///< network bandwidth
std::string m_phyMode; ///< remote station manager data mode
uint32_t m_nodeSpeed; ///< mobility speed
uint32_t m_periodicUpdateInterval; ///< routing update interval
uint32_t m_settlingTime; ///< routing setting time
double m_dataStart; ///< time to start data transmissions (seconds)
uint32_t bytesTotal; ///< total bytes received by all nodes
uint32_t packetsReceived; ///< total packets received by all nodes
bool m_printRoutes; ///< print routing table
std::string m_CSVfileName; ///< CSV file name

```

```

NodeContainer nodes; ///< the collection of nodes
NetDeviceContainer devices; ///< the collection of devices
Ipv4InterfaceContainer interfaces; ///< the collection of interfaces

```

private:

```

/// Create and initialize all nodes
void CreateNodes ();
/**
 * Create and initialize all devices
 * \param tr_name The trace file name
 */
void CreateDevices (std::string tr_name);
/**
 * Create network
 * \param tr_name The trace file name
 */
void InstallInternetStack (std::string tr_name);
/// Create data sinks and sources
void InstallApplications ();

```



```

/// Setup mobility model
void SetupMobility ();
/**
 * Packet receive function
 * \param socket The communication socket
 */

void ReceivePacket (Ptr <Socket> socket);
/**
 * Setup packet receivers
 * \param addr the receiving IPv4 address
 * \param node the receiving node
 * \returns the communication socket
 */
Ptr <Socket> SetupPacketReceive (Ipv4Address addr, Ptr <Node> node );
/// Check network throughput
void CheckThroughput ();

};

int main (int argc, char **argv)
{
    DsdvManetExample test;
    uint32_t nWifis = 30;
    uint32_t nSinks = 10;
    double totalTime = 100.0;
    std::string rate ("8kbps");
    std::string phyMode ("DsssRate11Mbps");
    uint32_t nodeSpeed = 10; // in m/s
    std::string appl = "all";
    uint32_t periodicUpdateInterval = 15;
    uint32_t settlingTime = 6;
    double dataStart = 50.0;
    bool printRoutingTable = true;
    std::string CSVfileName = "DsdvManetExample.csv";

    CommandLine cmd;
    cmd.AddValue ("nWifis", "Number of wifi nodes[Default:30]", nWifis);
    cmd.AddValue ("nSinks", "Number of wifi sink nodes[Default:10]", nSinks);
    cmd.AddValue ("totalTime", "Total Simulation time[Default:100]", totalTime);
    cmd.AddValue ("phyMode", "Wifi Phy mode[Default:DsssRate11Mbps]", phyMode);
    cmd.AddValue ("rate", "CBR traffic rate[Default:8kbps]", rate);
    cmd.AddValue ("nodeSpeed", "Node speed in RandomWayPoint model[Default:10]", nodeSpeed);
    cmd.AddValue ("periodicUpdateInterval", "Periodic Interval Time[Default=15]", periodicUpdateInterval);
    cmd.AddValue ("settlingTime", "Settling Time before sending out an update for changed metric[Default=6]",
    settlingTime);
    cmd.AddValue ("dataStart", "Time at which nodes start to transmit data[Default=50.0]", dataStart);
    cmd.AddValue ("printRoutingTable", "print routing table for nodes[Default:1]", printRoutingTable);

```

```

cmd.AddValue ("CSVfileName", "The name of the CSV output file name[Default:DsdvManetExample.csv]",
CSVfileName);
cmd.Parse (argc, argv);

std::ofstream out (CSVfileName.c_str ());
out << "SimulationSecond," <<
"ReceiveRate," <<
"PacketsReceived," <<
"NumberOfSinks," <<
std::endl;
out.close ();

SeedManager::SetSeed (12345);

Config::SetDefault ("ns3::OnOffApplication::PacketSize", StringValue ("1000"));
Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue (rate));
Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode", StringValue (phyMode));
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("2000"));

test = DsdvManetExample ();
test.CaseRun (nWifis, nSinks, totalTime, rate, phyMode, nodeSpeed, periodicUpdateInterval,
    settlingTime, dataStart, printRoutingTable, CSVfileName);

return 0;
}

DsdvManetExample::DsdvManetExample ()
: bytesTotal (0),
  packetsReceived (0)
{
}

void
DsdvManetExample::ReceivePacket (Ptr <Socket> socket)
{
    NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << " Received one packet!");
    Ptr <Packet> packet;
    while ((packet = socket->Recv ()))
    {
        bytesTotal += packet->GetSize ();
        packetsReceived += 1;
    }
}

Void DsdvManetExample::CheckThroughput ()
{

```

```

double kbs = (bytesTotal * 8.0) / 1000;
bytesTotal = 0;

std::ofstream out (m_CSVfileName.c_str (), std::ios::app);

out << (Simulator::Now ()).GetSeconds () << "," << kbs << "," << packetsReceived << "," << m_nSinks <<
std::endl;

out.close ();
packetsReceived = 0;
Simulator::Schedule (Seconds (1.0), &DsdvManetExample::CheckThroughput, this);
}

Ptr <Socket>
DsdvManetExample::SetupPacketReceive (Ipv4Address addr, Ptr <Node> node)
{

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr <Socket> sink = Socket::CreateSocket (node, tid);

InetSocketAddress local = InetSocketAddress (addr, port);

sink->Bind (local);
sink->SetRecvCallback (MakeCallback ( &DsdvManetExample::ReceivePacket, this));

return sink;
}

void
DsdvManetExample::CaseRun (uint32_t nWifis, uint32_t nSinks, double totalTime, std::string rate,
                           std::string phyMode, uint32_t nodeSpeed, uint32_t periodicUpdateInterval, uint32_t
settlingTime,
                           double dataStart, bool printRoutes, std::string CSVfileName)
{
m_nWifis = nWifis;
m_nSinks = nSinks;
m_totalTime = totalTime;
m_rate = rate;
m_phyMode = phyMode;
m_nodeSpeed = nodeSpeed;
m_periodicUpdateInterval = periodicUpdateInterval;
m_settlingTime = settlingTime;
m_dataStart = dataStart;
m_printRoutes = printRoutes;
m_CSVfileName = CSVfileName;

std::stringstream ss;
ss << m_nWifis;

```

```

std::string t_nodes = ss.str ();

std::stringstream ss3;
ss3 << m_totalTime;
std::string sTotalTime = ss3.str ();

std::string tr_name = "Dsdv_Manet_" + t_nodes + "Nodes_" + sTotalTime + "SimTime";
std::cout << "Trace file generated is " << tr_name << ".tr\n";

CreateNodes ();
CreateDevices (tr_name);
SetupMobility ();
InstallInternetStack (tr_name);
InstallApplications ();

std::cout << "\nStarting simulation for " << m_totalTime << " s ...\n";

CheckThroughput ();

Simulator::Stop (Seconds (m_totalTime));
Simulator::Run ();
Simulator::Destroy ();
}

void
DsdvManetExample::CreateNodes ()
{
    std::cout << "Creating " << (unsigned) m_nWifis << " nodes.\n";

    nodes.Create (m_nWifis);
    NS_ASSERT_MSG (m_nWifis > m_nSinks, "Sinks must be less or equal to the number of nodes in
network");
}

void
DsdvManetExample::SetupMobility ()
{
    MobilityHelper mobility;
    ObjectFactory pos;
    pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
    pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1000.0]"));
    pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1000.0]"));

    std::ostringstream speedConstantRandomVariableStream;
    speedConstantRandomVariableStream << "ns3::ConstantRandomVariable[Constant="
        << m_nodeSpeed
        << "]";

```

```

Ptr <PositionAllocator> taPositionAlloc = pos.Create ()->GetObject <PositionAllocator> ();
mobility.SetMobilityModel ("ns3::RandomWaypointMobilityModel", "Speed", StringValue
(speedConstantRandomVariableStream.str ()),
    "Pause", StringValue ("ns3::ConstantRandomVariable[Constant=2.0]"), "PositionAllocator",
PointerValue (taPositionAlloc));
mobility.SetPositionAllocator (taPositionAlloc);
mobility.Install (nodes);
}

```

```

void
DsdvManetExample::CreateDevices (std::string tr_name)
{
    WifiMacHelper wifiMac;
    wifiMac.SetType ("ns3::AdhocWifiMac");
    YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
    YansWifiChannelHelper wifiChannel;
    wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
    wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
    wifiPhy.SetChannel (wifiChannel.Create ());
    WifiHelper wifi;
    wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
    wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue (m_phyMode),
"ControlMode",
    StringValue (m_phyMode));
    devices = wifi.Install (wifiPhy, wifiMac, nodes);

    AsciiTraceHelper ascii;
    wifiPhy.EnableAsciiAll (ascii.CreateFileStream (tr_name + ".tr"));
    wifiPhy.EnablePcapAll (tr_name);
}

```

```

void
DsdvManetExample::InstallInternetStack (std::string tr_name)
{
    DsdvHelper dsdv;

    dsdv.Set ("PeriodicUpdateInterval", TimeValue (Seconds (m_periodicUpdateInterval)));
    dsdv.Set ("SettlingTime", TimeValue (Seconds (m_settlingTime)));
    InternetStackHelper stack;
    stack.SetRoutingHelper (dsdv); // has effect on the next Install ()
    stack.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    interfaces = address.Assign (devices);
    if (m_printRoutes)
    {

```

```

    Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ((tr_name + ".routes"),
std::ios::out);
    dsdv.PrintRoutingTableAllAt (Seconds (m_periodicUpdateInterval), routingStream);
}
}

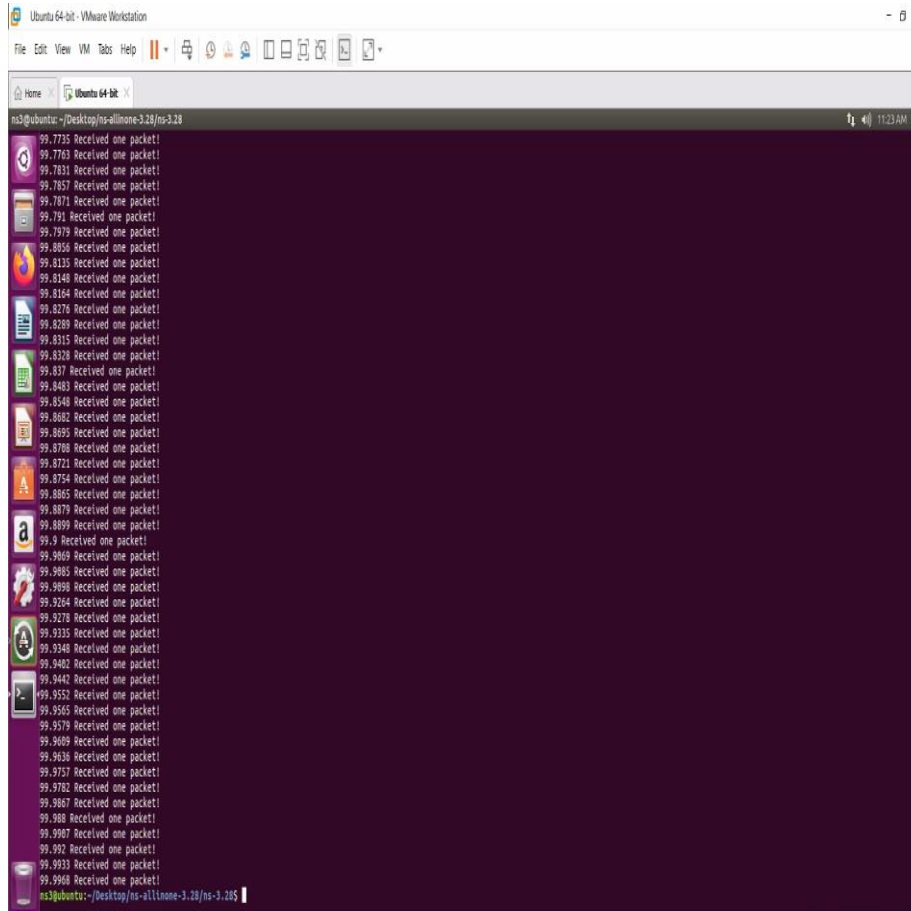
void
DsdvManetExample::InstallApplications ()
{
    for (uint32_t i = 0; i <= m_nSinks - 1; i++)
    {
        Ptr<Node> node = NodeList::GetNode (i);
        Ipv4Address nodeAddress = node->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
        Ptr<Socket> sink = SetupPacketReceive (nodeAddress, node);
    }

    for (uint32_t clientNode = 0; clientNode <= m_nWifis - 1; clientNode++)
    {
        for (uint32_t j = 0; j <= m_nSinks - 1; j++)
        {
            OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address (InetSocketAddress (interfaces.GetAddress (j),
port)));
            onoff1.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1.0]"));
            onoff1.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0.0]"));

            if (j != clientNode)
            {
                ApplicationContainer apps1 = onoff1.Install (nodes.Get (clientNode));
                Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable> ();
                apps1.Start (Seconds (var->GetValue (m_dataStart, m_dataStart + 1)));
                apps1.Stop (Seconds (m_totalTime));
            }
        }
    }
}

```

# Output



The screenshot shows a terminal window titled 'ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28'. The terminal output consists of a continuous list of messages: '99.7735 Received one packet!', '99.7763 Received one packet!', '99.7831 Received one packet!', '99.7857 Received one packet!', '99.7879 Received one packet!', '99.791 Received one packet!', '99.7979 Received one packet!', '99.8056 Received one packet!', '99.8135 Received one packet!', '99.8148 Received one packet!', '99.8164 Received one packet!', '99.8276 Received one packet!', '99.8289 Received one packet!', '99.8315 Received one packet!', '99.8328 Received one packet!', '99.837 Received one packet!', '99.8483 Received one packet!', '99.8548 Received one packet!', '99.8682 Received one packet!', '99.8695 Received one packet!', '99.8788 Received one packet!', '99.8721 Received one packet!', '99.8754 Received one packet!', '99.8865 Received one packet!', '99.8879 Received one packet!', '99.8899 Received one packet!', '99.9 Received one packet!', '99.9069 Received one packet!', '99.9085 Received one packet!', '99.9098 Received one packet!', '99.9104 Received one packet!', '99.9278 Received one packet!', '99.9319 Received one packet!', '99.9348 Received one packet!', '99.9482 Received one packet!', '99.9442 Received one packet!', '99.9552 Received one packet!', '99.9565 Received one packet!', '99.9578 Received one packet!', '99.9609 Received one packet!', '99.9638 Received one packet!', '99.9757 Received one packet!', '99.9782 Received one packet!', '99.9867 Received one packet!', '99.988 Received one packet!', '99.9907 Received one packet!', '99.992 Received one packet!', '99.9933 Received one packet!', '99.9968 Received one packet!'. The prompt 'ns3@ubuntu:~/Desktop/ns-allinone-3.28/ns-3.28\$' is visible at the bottom.