

Worklet Name: Performance Evaluation of DSCP values in various network

Worklet Details

1. Worklet ID: CP214MS
2. College Name: Ramaiah Institute of Technology.

KPIs achieved till now

We have implemented different types of topologies in NS3 and analysed the traffic using Wire shark.
We learnt to configure client and server.

Any Challenges/ Issues faced

To simulate different network scenarios and also to set up different DSCP code points between client and the server.

Next Steps

Performance evaluation with different DSCP values in different network conditions like lossy, congested etc.

Key Achievements/ Outcome till now

We have setup different client and server and have implemented different types of protocols.

The background features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are primarily located on the left and right sides of the slide, framing the central text area.

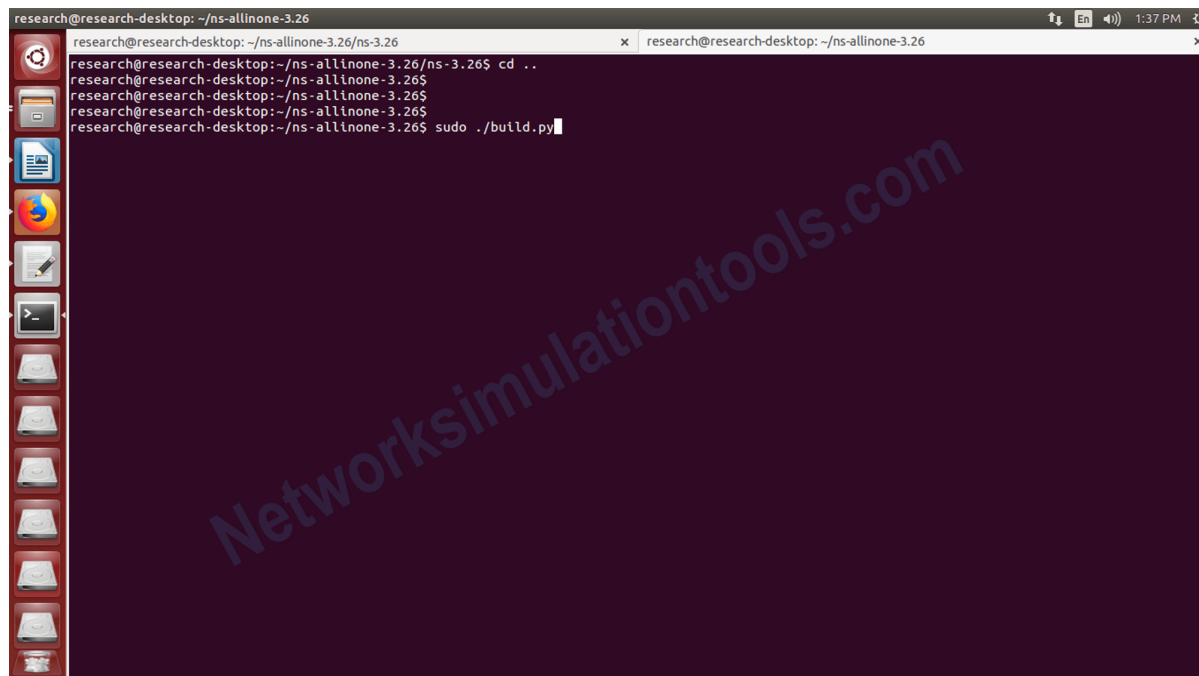
Client and Server Configurations using NS3

Using the NS3 ,we can generate various point to point codes such as first.cc ,second.cc and third.cc etc

The basic steps to be followed to generate the various client server networks are :

Install the NS-3.26

Initially , install the NS-3.26 tool by using the ns-allinone-3.26.tar.bz2 package. For make install use and execute the command ./b.py



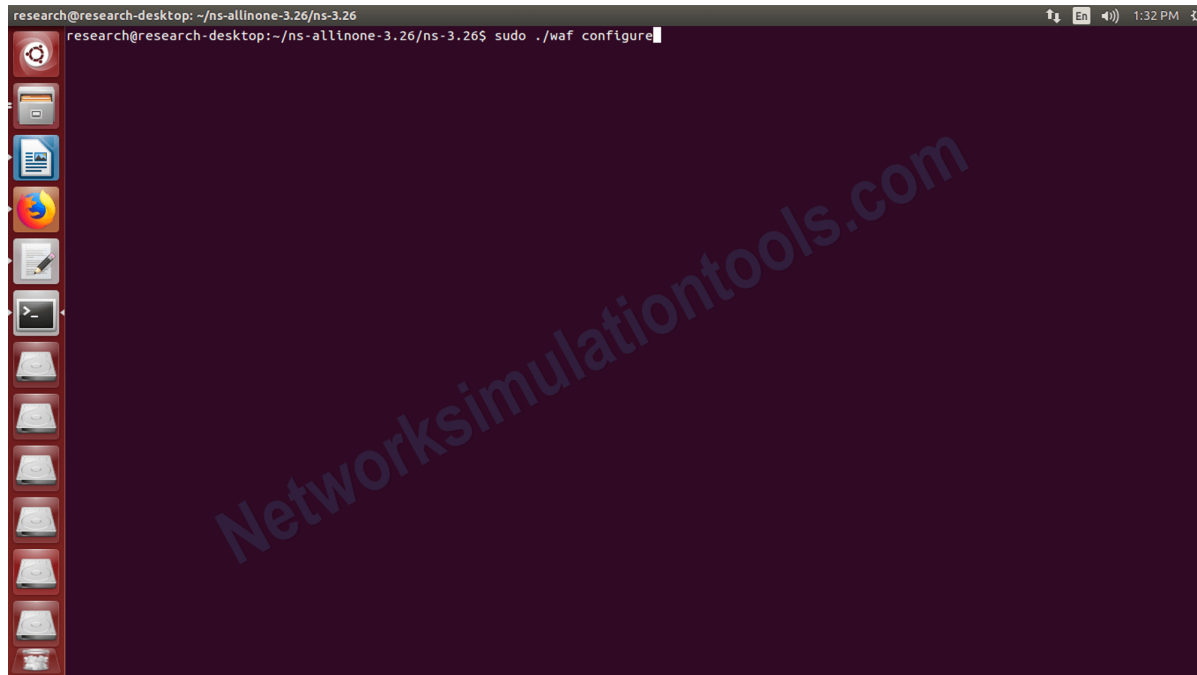
A terminal window titled 'research@research-desktop: ~/ns-allinone-3.26' showing the installation process. The terminal output is as follows:

```
research@research-desktop: ~/ns-allinone-3.26
research@research-desktop: ~/ns-allinone-3.26$ cd ..
research@research-desktop: ~/ns-allinone-3.26$
research@research-desktop: ~/ns-allinone-3.26$
research@research-desktop: ~/ns-allinone-3.26$ sudo ./b.py
```

The terminal window is part of a desktop environment with a sidebar on the left containing icons for various applications. A large, semi-transparent watermark 'Networksimulationtools.com' is visible across the terminal area.

1. Configure the package

Change the ns-allinone installation location in the terminal, using the command `cd ns-allinone-3.26/ns-3.26`. And execute the command `sudo ./waf configure` to binding the python



Build the package

For build the configured package by using the `sudo ./waf build` command in the `ns-allinone-3.26/ns-3.26` . Package and view the build packages.

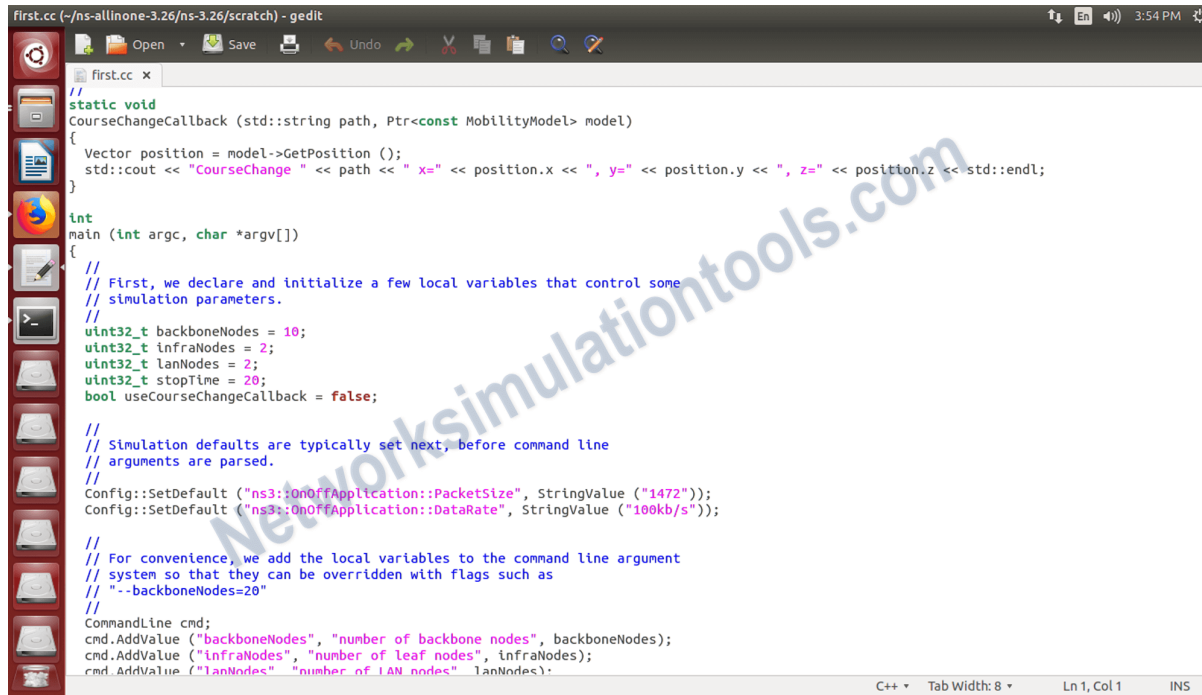
```
research@research-desktop: ~/ns-allinone-3.26/ns-3.26
Gcrypt library      : not enabled (libgcrypt not found: you can use libgcrypt-config to find its location.)
GtkConfigStore      : enabled
MPI Support         : not enabled (option --enable-mpi not selected)
Network Simulation Cradle : not enabled (NSC not found (see option --with-nsc))
PlanetLab FdNetDevice : not enabled (PlanetLab operating system not detected (see option --force-planetlab))
PyViz visualizer    : enabled
Python API Scanning Support : enabled
Python Bindings     : enabled
Real Time Simulator : enabled
SQLite stats data output : enabled
Tap Bridge          : enabled
Tap FdNetDevice     : enabled
Tests               : not enabled (defaults to disabled)
Threading Primitives : enabled
Use sudo to set suid bit : not enabled (option --enable-sudo not selected)
XmlIo               : enabled
'configure' finished successfully (2.579s)
research@research-desktop:~/ns-allinone-3.26/ns-3.26$ sudo ./waf build
waf: Entering directory '/home/research/ns-allinone-3.26/ns-3.26/build'
waf: Leaving directory '/home/research/ns-allinone-3.26/ns-3.26/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (9.474s)

Modules built:
antenna          aodv          applications
bridge           buildings    cognitive (no Python)
config-store     core         csma
csma-layout      dsdv         dsr
energy           evalvid (no Python) fd-net-device
flow-monitor     gusers      internet
internet-apps    lr-wpan     lte
mesh             mobility     mpi
netanim (no Python) network     nix-vector-routing
olsr             openflow    point-to-point
point-to-point-layout propagation spectrum
stats            tap-bridge  test (no Python)
topology-read    traffic-control uan
virtual-net-device visualizer   wave
wifi             wimax      wsn (no Python)

research@research-desktop:~/ns-allinone-3.26/ns-3.26$
```

Create a main file

Next create new program file in the scratch folder. The program file stored with the file extension .cc. Include the header files for the needed modules.



```
first.cc (-/ns-allinone-3.26/ns-3.26/scratch) - gedit
// static void
CourseChangeCallback (std::string path, Ptr<const MobilityModel> model)
{
    Vector position = model->GetPosition ();
    std::cout << "CourseChange" << path << " x=" << position.x << ", y=" << position.y << ", z=" << position.z << std::endl;
}

int
main (int argc, char *argv[])
{
    //
    // First, we declare and initialize a few local variables that control some
    // simulation parameters.
    //
    uint32_t backboneNodes = 10;
    uint32_t infraNodes = 2;
    uint32_t lanNodes = 2;
    uint32_t stopTime = 20;
    bool useCourseChangeCallback = false;

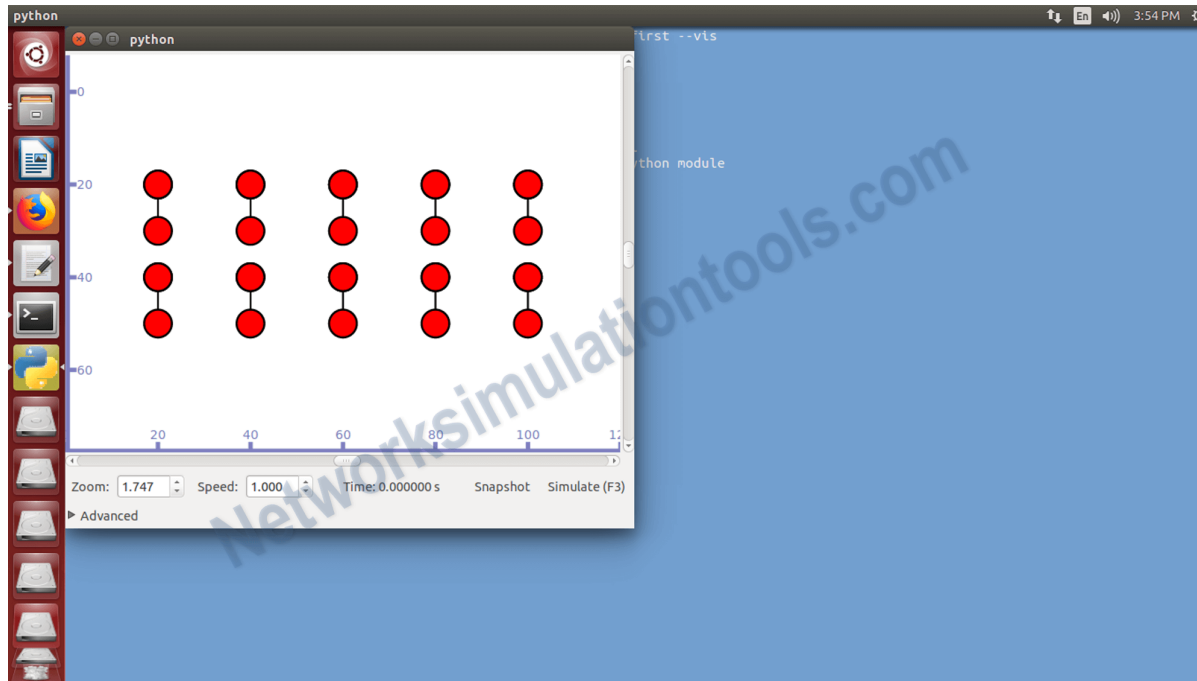
    //
    // Simulation defaults are typically set next, before command line
    // arguments are parsed.
    //
    Config::SetDefault ("ns3::OnOffApplication::PacketSize", StringValue ("1472"));
    Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("100kb/s"));

    //
    // For convenience, we add the local variables to the command line argument
    // system so that they can be overridden with flags such as
    // "--backboneNodes=20"
    //
    CommandLine cmd;
    cmd.AddValue ("backboneNodes", "number of backbone nodes", backboneNodes);
    cmd.AddValue ("infraNodes", "number of leaf nodes", infraNodes);
    cmd.AddValue ("lanNodes", "number of LAN nodes", lanNodes);

C++ Tab Width: 8 Ln 1, Col 1 INS
```

1. Run the main command

Next, implement the foremost main file by using the command `sudo ./waf --run first --vis` to run the simulation result.



Execution of First.cc and its output

In the first.cc point to point echo client server connection is established n0-----n1 where n0 client and n1 server and data rate and delay can be configured .

FIRST.CC

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */  
/*  
 * This program is free software; you can redistribute it and/or modify  
 * it under the terms of the GNU General Public License version 2 as  
 * published by the Free Software Foundation;  
 *  
 * This program is distributed in the hope that it will be useful,  
 * but WITHOUT ANY WARRANTY; without even the implied warranty of  
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
 * GNU General Public License for more details.  
 *  
 * You should have received a copy of the GNU General Public License  
 * along with this program; if not, write to the Free Software  
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  
 */
```

```
#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"
```

```
//      10.1.1.0  
// n0 ----- n1  
// point-to-point  
//
```



```

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
    CommandLine cmd (__FILE__);
    cmd.Parse (argc, 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;

```

```

//          10.1.1.0
// n0 ----- n1
// point-to-point
//

```

```
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;
}
```

```
vatsalya@Vatsalya:~/Desktop/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/first.cc
Consolidate compiler generated dependencies of target scratch_first
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
```

Execution of second.cc and its output

Second.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
```

```
#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 nCsmas = 3;
```

```
// Default Network Topology
```

```
//
```

```
//
```

```
//          10.1.1.0
```

```
// n0 ----- n1    n2    n3    n4
```

```
// point-to-point |    |    |    |
```

```
//
```

```
//
```

```
=====
LAN 10.1.2.0
```

```
CommandLine cmd (__FILE__);

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

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);
}

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 (nCsm));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));
```

```
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsmas, 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);
```

```
nCsmas = nCsmas == 0 ? 1 : nCsmas;
```

```
NodeContainer p2pNodes;  
p2pNodes.Create (2);  
NodeContainer csmaNodes;  
csmaNodes.Add (p2pNodes.Get (1));  
csmaNodes.Create (nCsmas);  
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");  
Simulator::Run ();  
    Simulator::Destroy ();  
    return 0;  
}
```

```
vatsalya@Vatsalya:~/Desktop/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/second.cc  
Consolidate compiler generated dependencies of target scratch_second  
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
```

Intermediate Nodes N0-----N1-----N2

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
    CommandLine cmd;
    cmd.Parse (argc, argv);

    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication",
LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication",
LOG_LEVEL_INFO);

    NodeContainer nodes;
    nodes.Create (3);

    PointToPointHelper pointToPoint1;
    pointToPoint1.SetDeviceAttribute ("DataRate", StringValue
("5Mbps"));
    pointToPoint1.SetChannelAttribute ("Delay", StringValue
("2ms"));
```

```
NetDeviceContainer devices1;  
devices1 = pointToPoint1.Install (nodes.Get(0),nodes.Get(1));  
  
NetDeviceContainer devices2;  
devices2 = pointToPoint2.Install (nodes.Get(1),nodes.Get(2));  
  
InternetStackHelper stack;  
stack.Install (nodes);  
  
Ipv4AddressHelper address1;  
address1.SetBase ("10.1.1.0", "255.255.255.0");  
  
Ipv4AddressHelper address2;  
address2.SetBase ("10.1.2.0", "255.255.255.0");  
  
Ipv4InterfaceContainer interfaces1 = address1.Assign  
(devices1);  
Ipv4InterfaceContainer interfaces2 = address1.Assign  
(devices2);  
  
SystemSettingsHelper echoServer (9);  
  
ApplicationContainer serverApps = echoServer.Install  
(nodes.Get (2));  
serverApps.Start (Seconds (1.0));  
serverApps.Stop (Seconds (10.0));
```

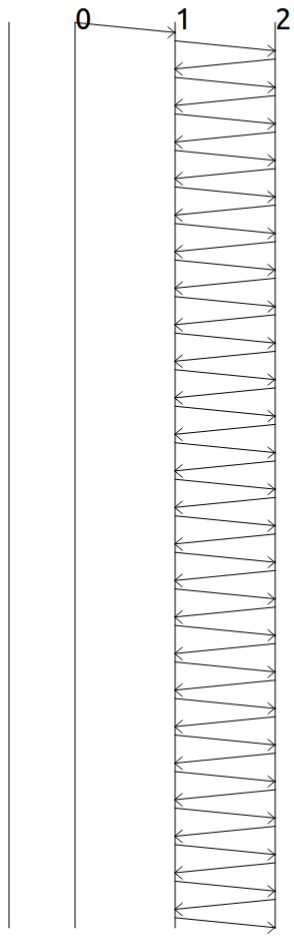


```
UdpEchoClientHelper echoClient (interfaces2.GetAddress (1),
9);
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds
(1.0)));
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));

ApplicationContainer clientApps = echoClient.Install
(nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

AnimationInterface anim ("anim1.xml");
anim.SetConstantPosition(nodes.Get(0),1.0,2.0);
anim.SetConstantPosition(nodes.Get(1),2.0,3.0);
anim.SetConstantPosition(nodes.Get(2),3.0,4.0);

Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```



port Tab		
	From Id	To Id
1	0	1
2	1	2
3	2	1
4	1	2
5	2	1
6	1	2
7	2	1
8	1	2
9	2	1
10	1	2
11	2	1
12	1	2
...		
13	2	1
14	1	2
15	2	1
16	1	2
17	2	1
18	1	2
19	2	1
20	1	2
21	2	1
22	1	2
23	2	1
24	1	2
25	2	1
26	1	2
27	2	1

