NS3 Overview

Network simulation using NS-3

March, 2018

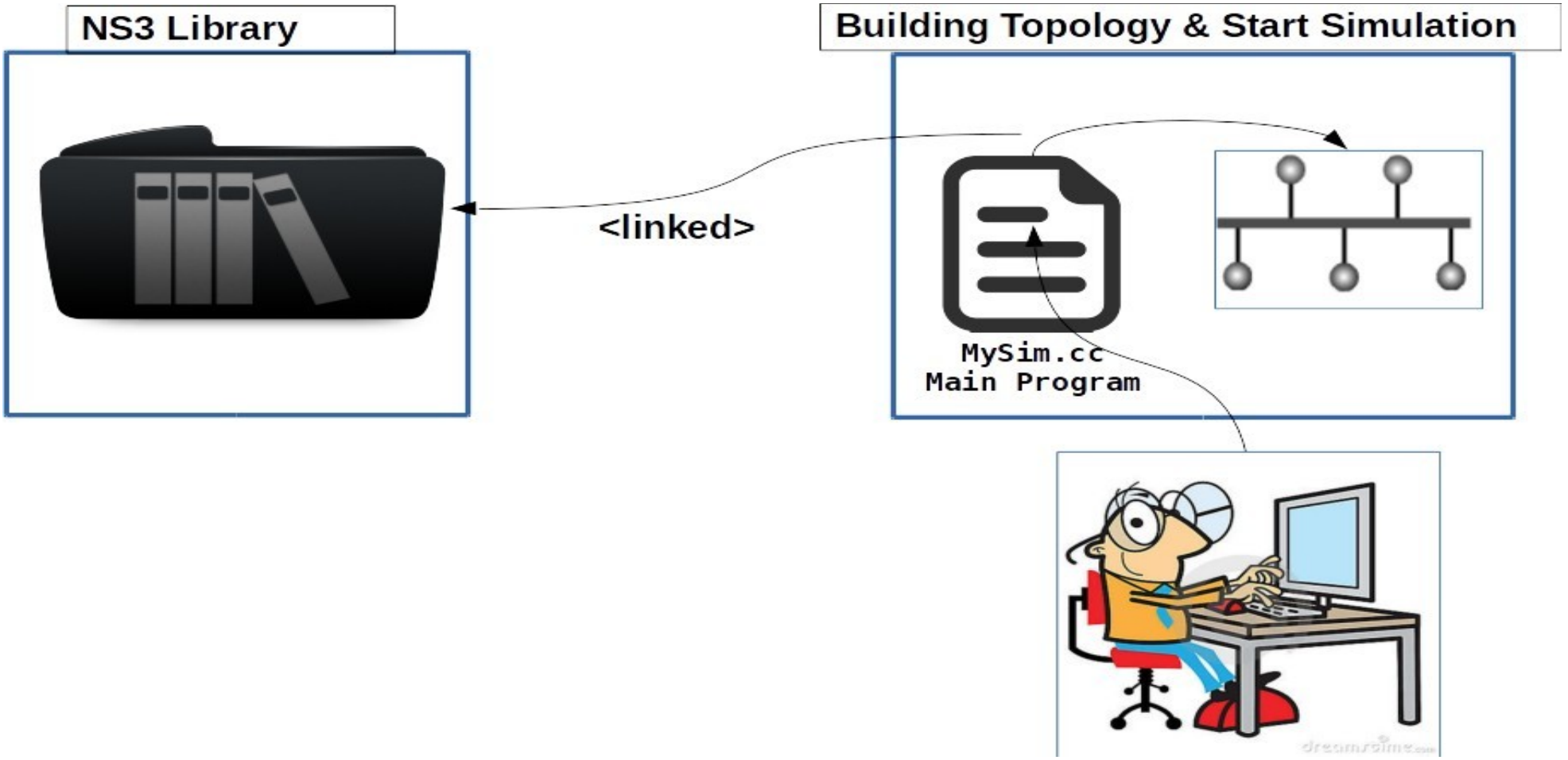
Road Map

- Network Simulation-3
- Installation of NS3
- Tools
- Implementaion (Structural)
  - **PointToPoint**
  - **LAN**
  - **Wifi**
- Implementation (OOP)
- Conclusion

NS3 Overview

- **An open source discrete event simulator**
  - Event model packet transmission , receipt , timers etc.
  - Future events maintained in sorted Event List
  - Processing events results in zero or more new events
- **Written in C++**
  - Extensive use of Templates , Smart Pointers, Callbacks
  - C++ namespace (ns3)
- **Simulation programs are C++ executables**
- **Python is used to bind public APIs provided**
- **NS-3 is built as a library which may be linked to a C++ main program defines the simulation topology and start the simulation.**

Use of Library



NS3 Installation

Download NS3

- $ wget http://www.nsnam.org/release/ns-allinone-3.26.tar.bz2

Extaract to a directory

- $ tar xjf ns-allinone-3.26.tar.bz2

Uses build script and enable examples and tests

- ./build.py --enable-examples --enable-tests
- cd ns-3.26
- ./test.py
-

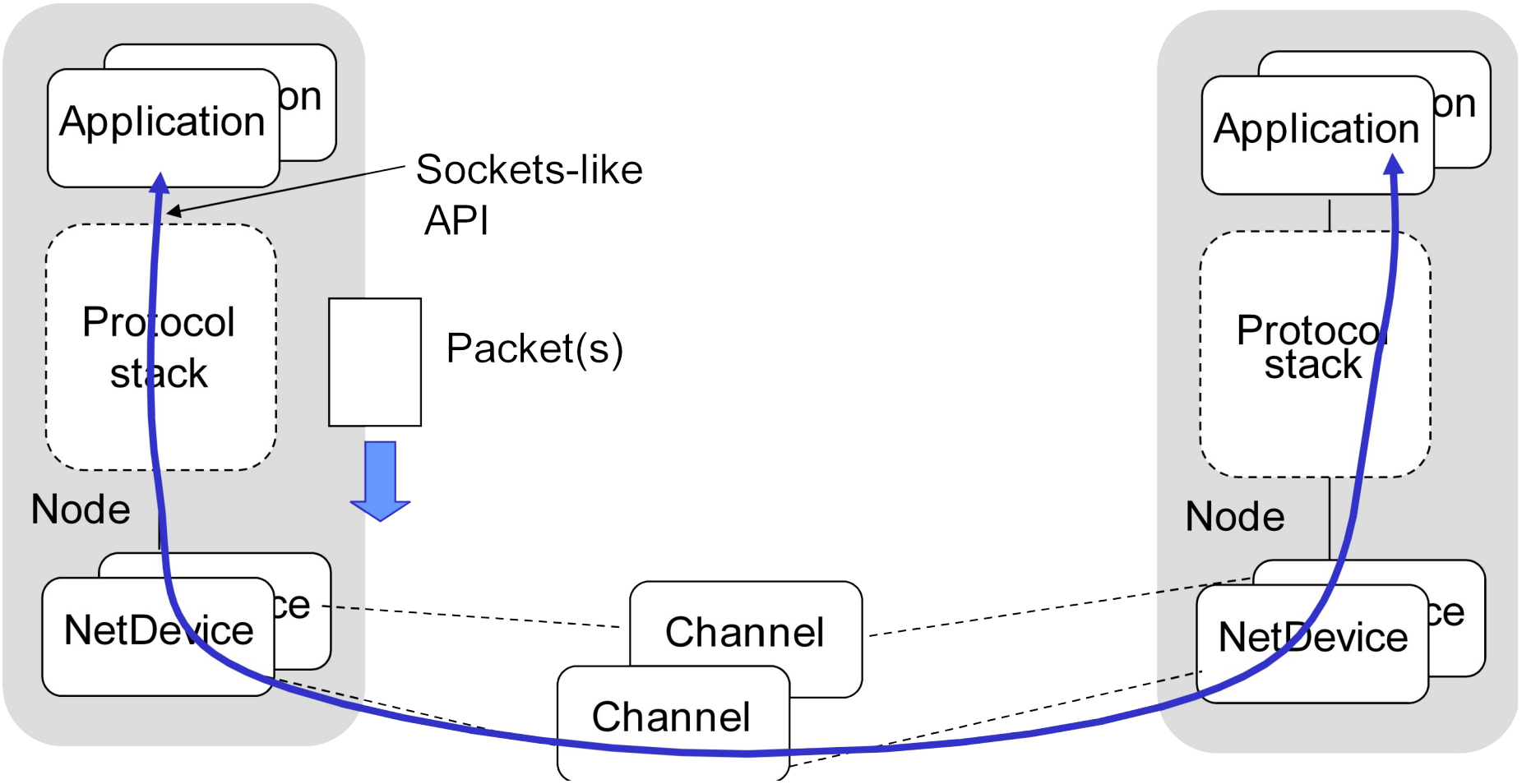
To execute NS3 script

- ./waf –run scratch/first

Detailed directions:

- http://www.nsnam.org/docs/release/3.14/tutorial/singlehtml/index.html#building-ns-3

Basic NS3 Architechture

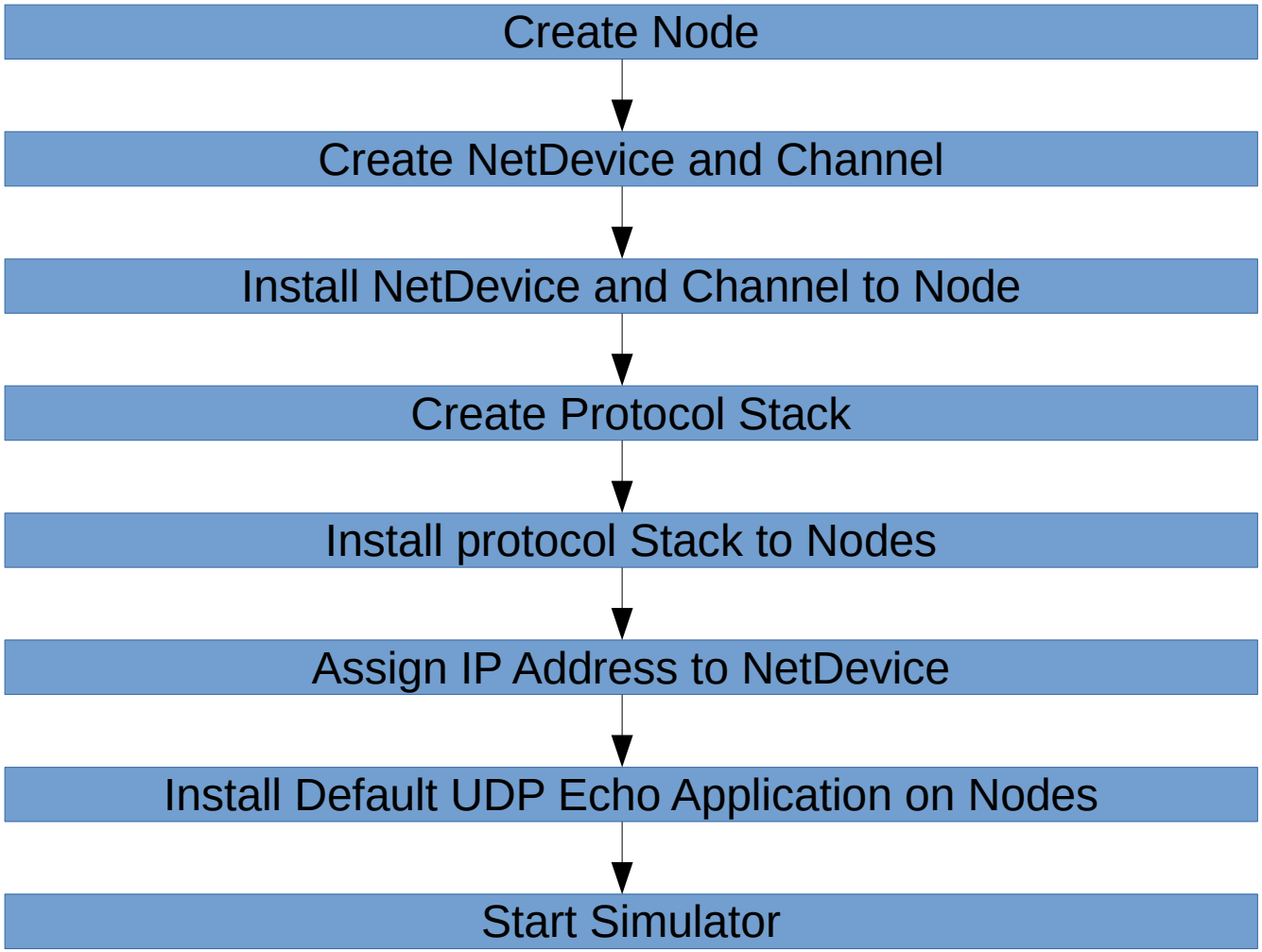


Structure of Generic NS3 Script

Structure of generic NS3 is step by step process:

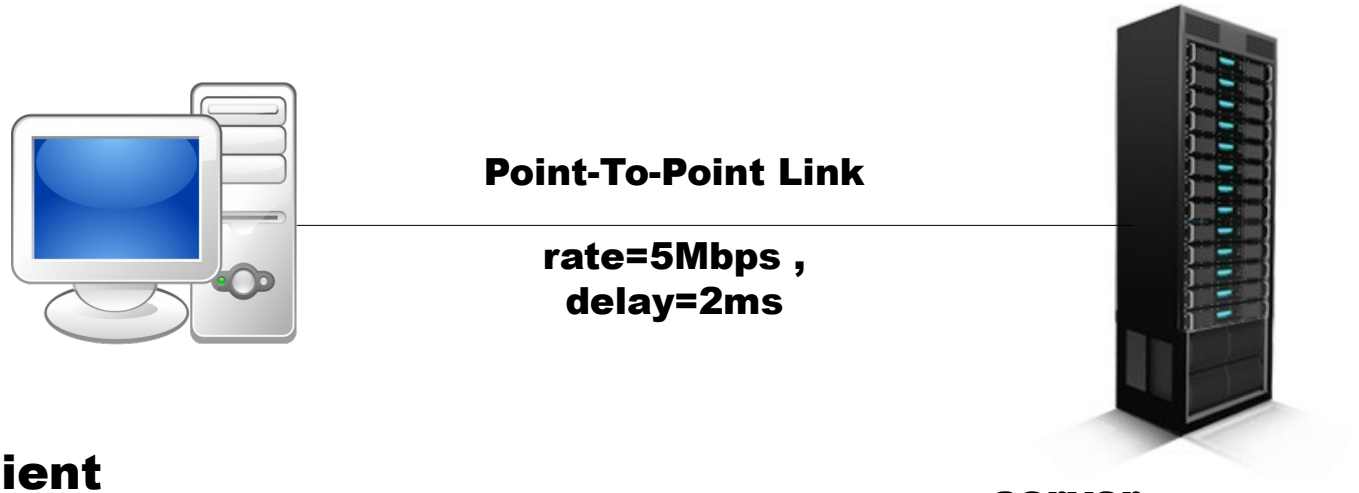
- Boilerplate: important for documentation
- Module includes: include header files
- ns-3 namespace: global declaration
- Logging: optional
- Main function: declare main function
- Topology helpers: objects to combine distinct operations
- Applications: on/off, UdpEchoClient/Server
- Tracing: .tr and/or .pcap files
- Simulator: start/end simulation, cleanup

Flow Chart of Generic NS3 Script



PointToPoint Network

A point to point Network that will be implemented in NS3:



Implementation (1)

```
#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[])
{
  CommandLine cmd;
  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);
```

include modules that will be used

ns-3 project namespace

To get command line input

enable and disable console message logging by reference to the name

Topology Configuration

Implementation (2)

```
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 (5));
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;
}
```

Set up Internet stack

Set up application

Run Simulation

Output

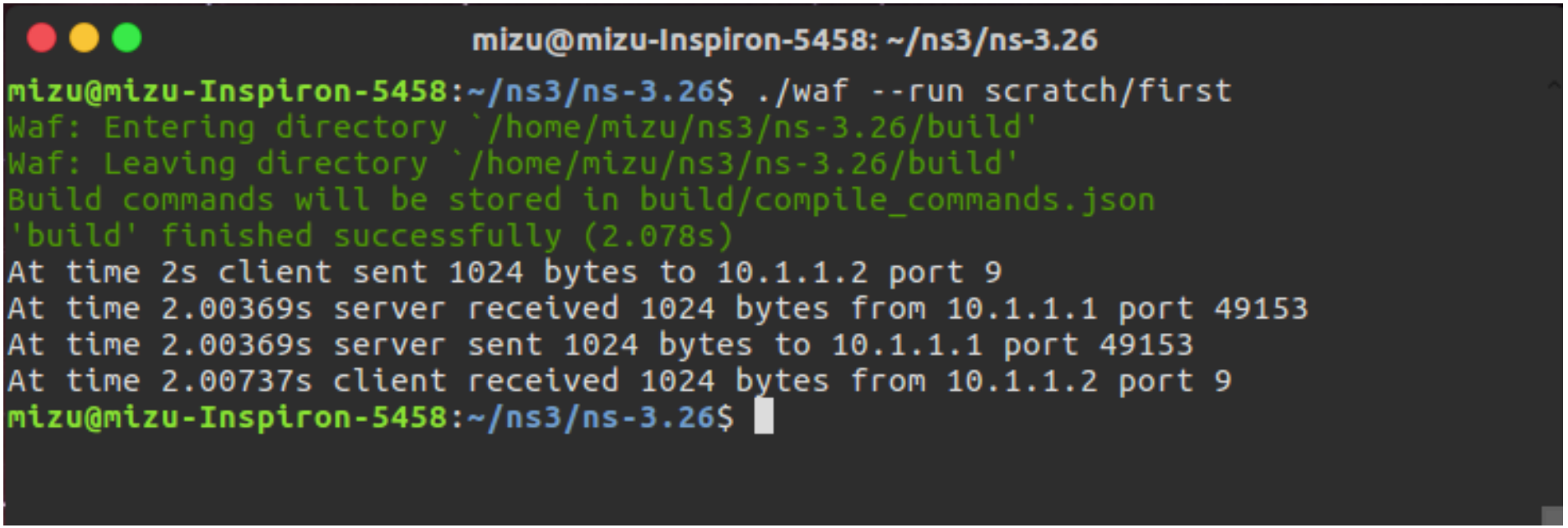
The output of ns3 file can be shown in three way:

1. Console output
2. Pyviz tool
3. NetAnim tool

Console output

Console Output:

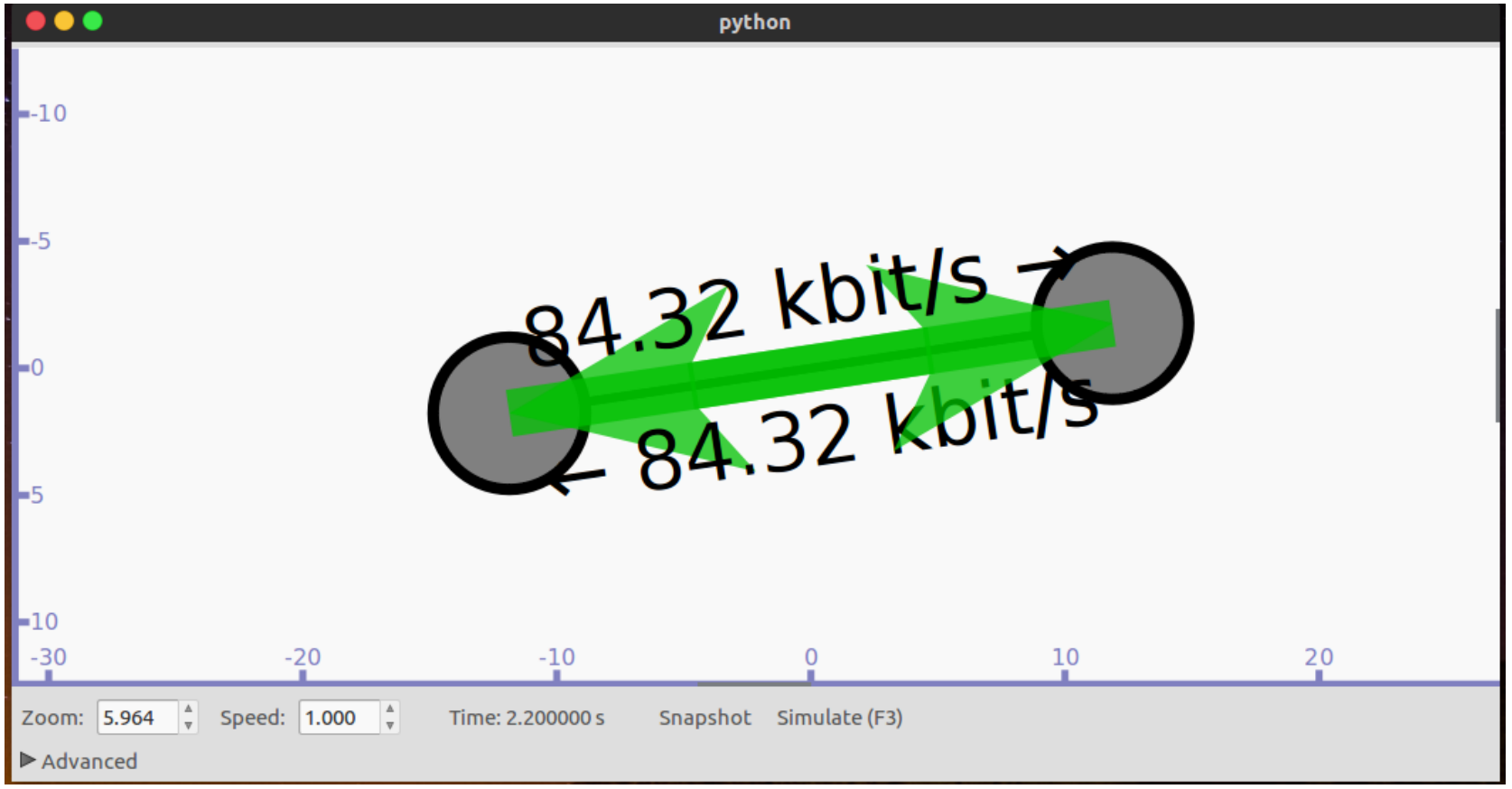
$ ./waf –-run scratch/first



Pyviz Output

PyViz tool:

$ ./waf –-run scratch/first --vis



NetAnim Output (1)

NetAnim tool:

To use NetAnim tool the implemented code needs to be changed.

- Module netanim-module.h need to add to module section

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

- AnimationInterface object needs to define before Simulator::Run()

```
AnimationInterface anim ("scratch/animation.xml");
```

- Set give positions to your nodes.

```
anim.SetConstantPosition (node, double x, double y);
```

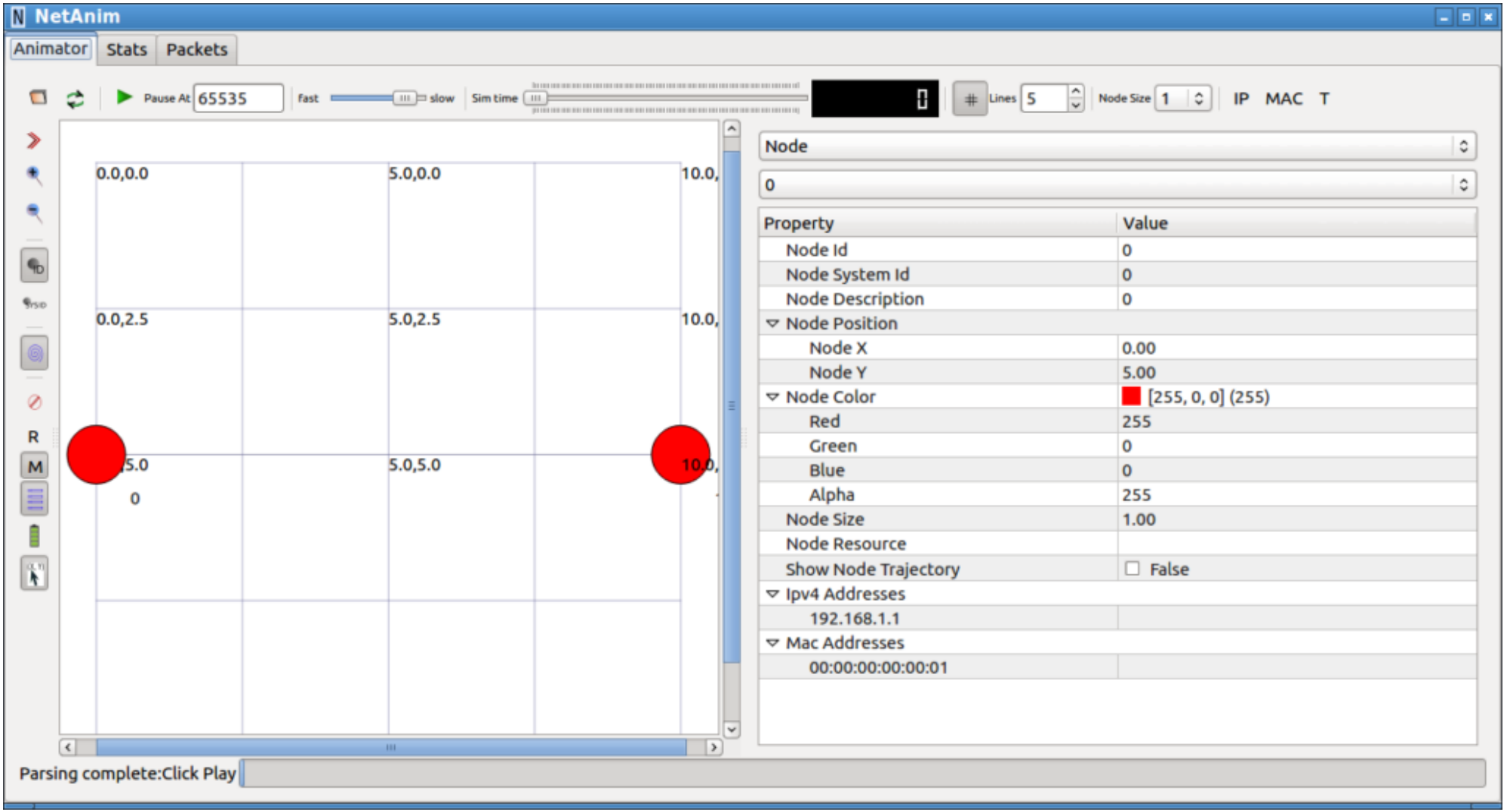
- A file animation.xml will be generated in scratch directory
- Then goto NetAnim directory and run using command

```
$ ./NetAnim
```

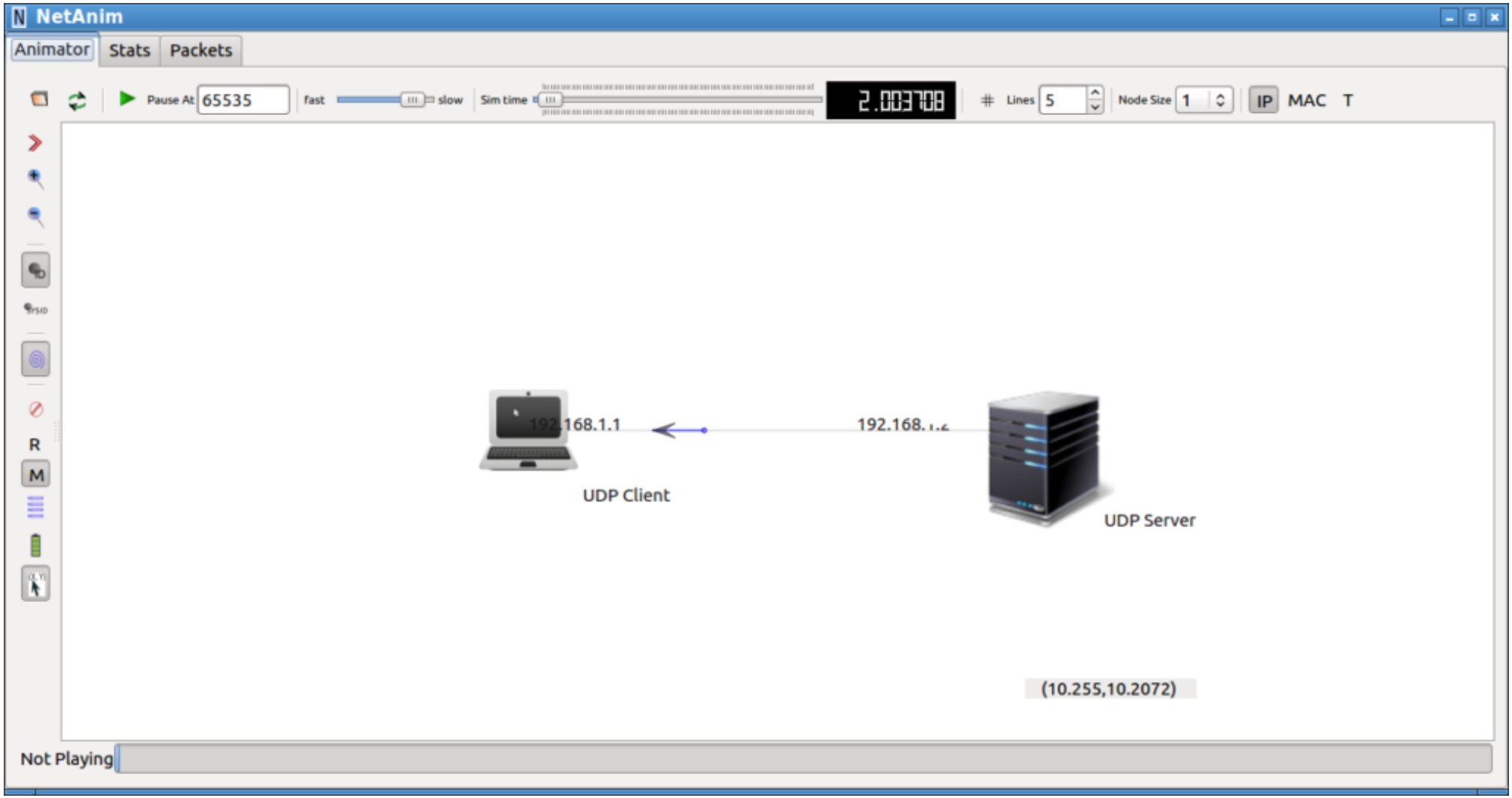
–

- Select animation.xml file to see the output.

NetAnim Output (2)

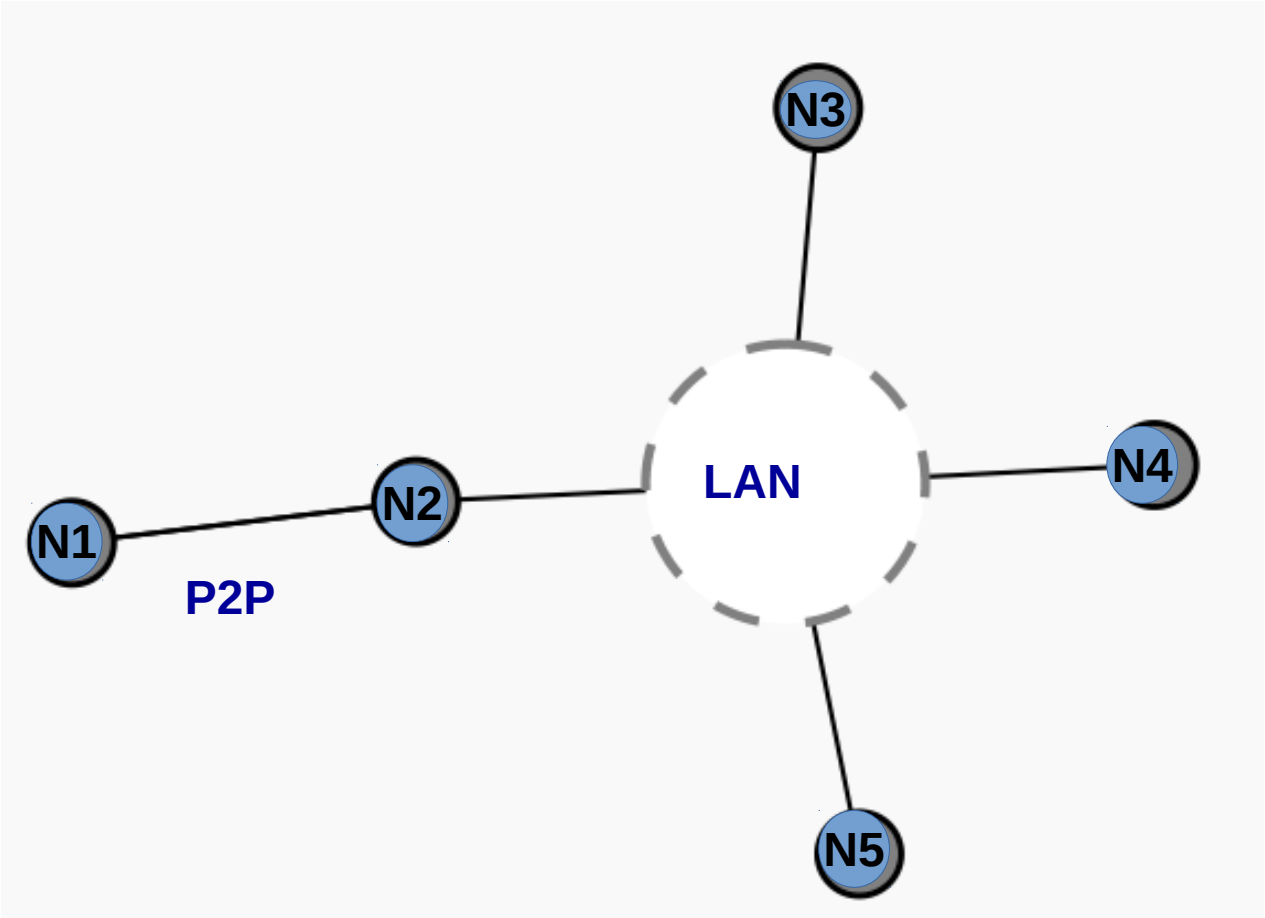


NetAnim Output (3)



LAN Network

A LAN Network will be implemented in NS3:



Implementation

- Module csma-module.h need to add to module section

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

- Declare Csma Nodes & theor properties

```
NodeContainer csmaNodes;
csmaNodes.Create (nCsma);

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
```

- Device Initialization

```
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);
```

- Assaign IP Address

```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
```

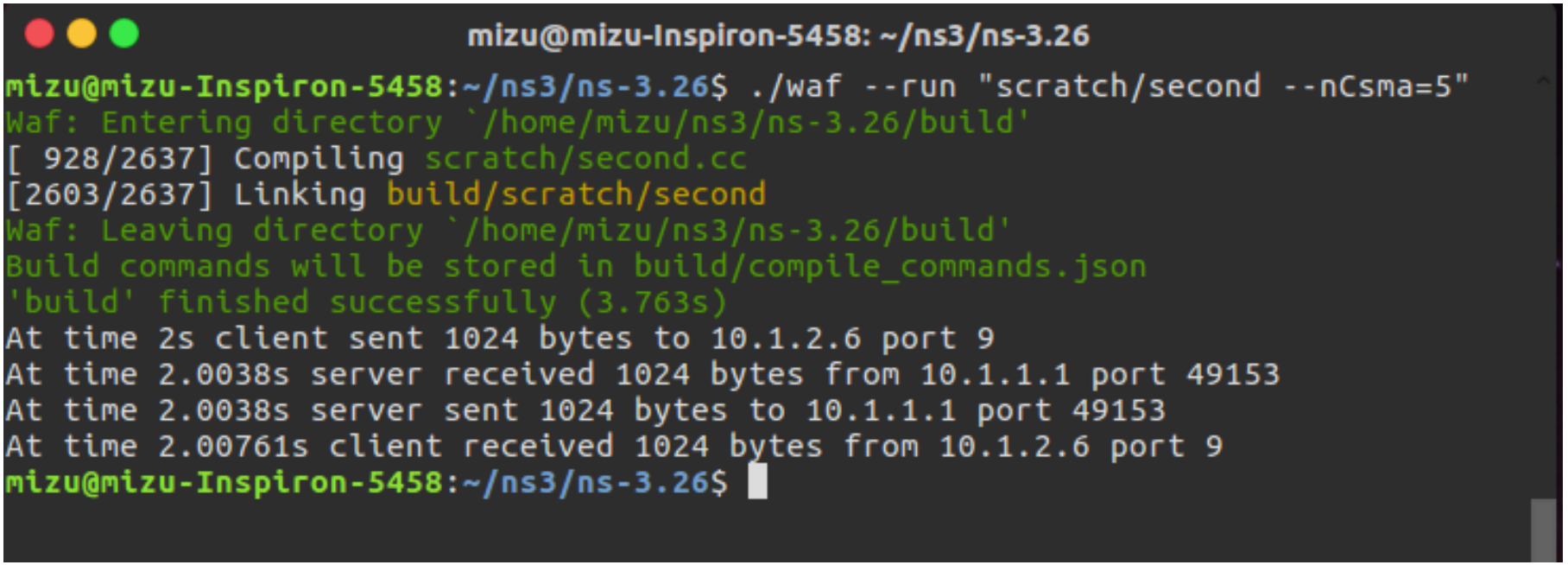
Console output

Console Output:

$ ./waf –-run scratch/second

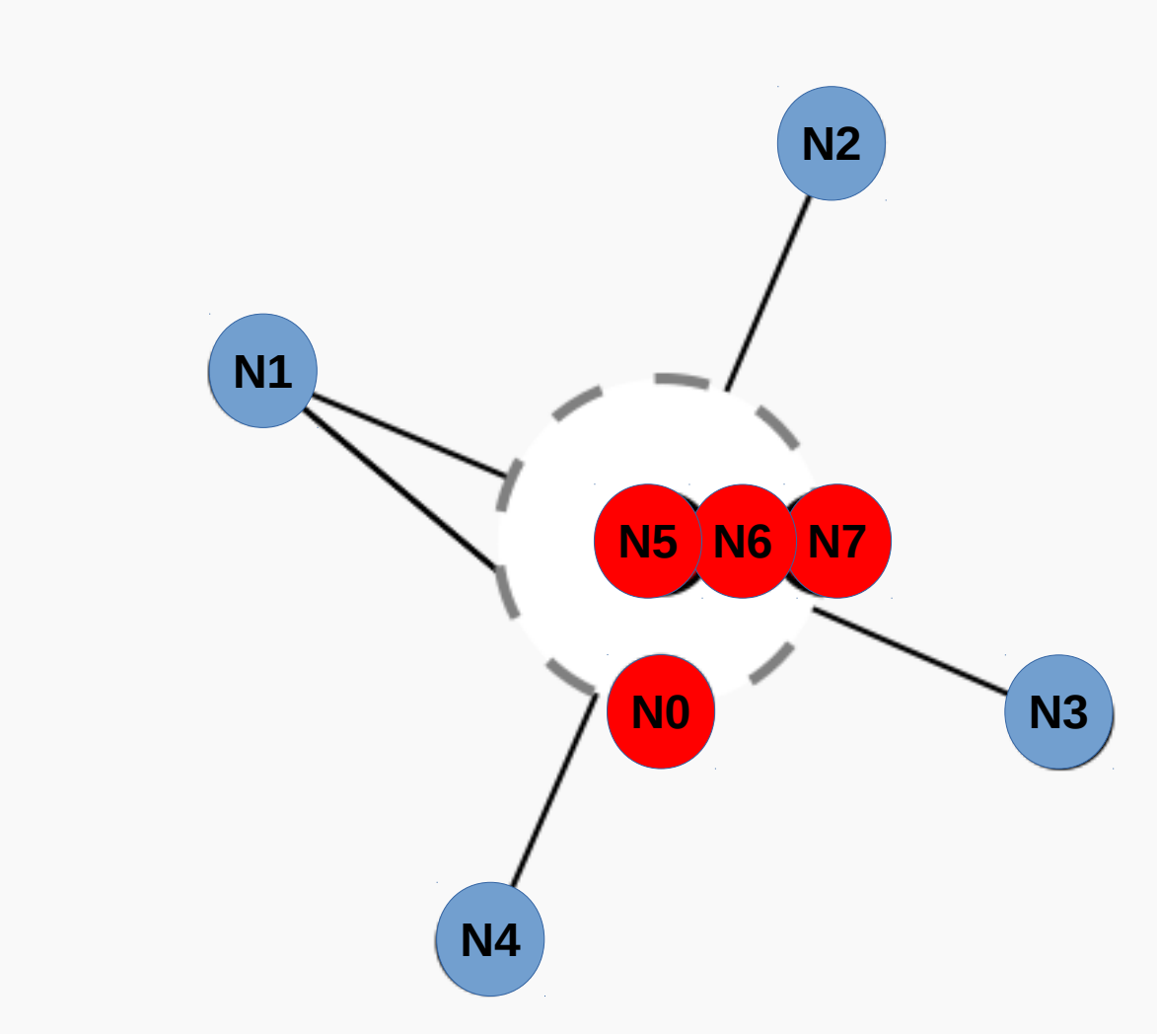
To set csma nodes from command line

$ ./waf –-run “scratch/second --nCsma=5”



Wifi Network

A Wifi Network will be implemented in NS3:



Implementation

- Essential module to be added

  #include "ns3/wifi-module.h"

  - #include "ns3/mobility-module.h"

- Set up wireless network

  //Node Declaration

  NodeContainer wifiStaNodes;

  wifiStaNodes.Create (nWifi);

  NodeContainer wifiApNode = p2pNodes.Get (0);

  -

  //Set Channel

  YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();

  YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();

  phy.SetChannel (channel.Create ());

  //Set Remote Station Manager

  WifiHelper wifi;

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

Implementation

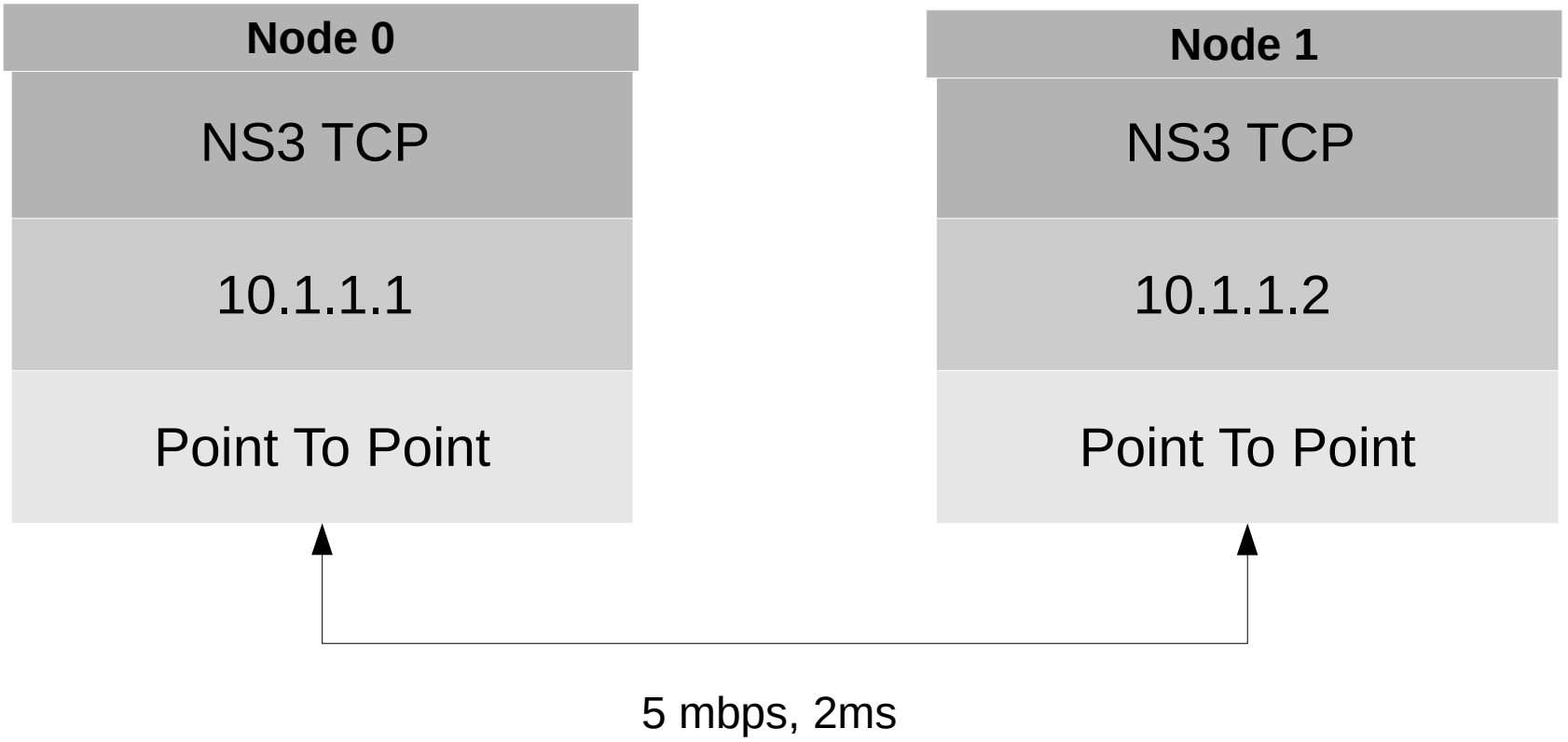
```
//Assaign Mac
WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
             "Ssid", SsidValue (ssid),
            "ActiveProbing", BooleanValue (false));

//Set Wifi Station Device
NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);
mac.SetType ("ns3::ApWifiMac",
             "Ssid", SsidValue (ssid));

//Wireless Access Point
NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);
```

Implementation

```
//Mobility
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);

//Internet Stack
InternetStackHelper stack;
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

//Address Assaign
Ipv4AddressHelper address;
address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);
```

OOP Approach

NS3 is an emerging simulator to replace the previous one named NS



Demo

NS3 Callback

- Class template CallBack<> implements the functor design pattern.
- Callbacks are more like function pointer but, type safed.

Static double CbOne(double a, double b) { }

Callback<double, double, double> one;

- Bind a function with a matching signature to a callback.

```
one = MakeCallback (&CbOne);
double returnOne = one (10.0, 2
```

Enabling Gnuplot

- `examples/wireless/wifi-clear-channel-cmu.cc`

```
CommandLine cmd;
cmd.Parse (argc, argv);

Gnuplot gnuplot = Gnuplot ("clear-channel.eps");

for (uint32_t i = 0; i < modes.size (); i++)
  {
    std::cout << modes[i] << std::endl;
    Gnuplot2dDataset dataset (modes[i]);
```

produce a plot file that will generate an EPS figure

one dataset per mode

```
    uint32_t pktsRecvd = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
    dataset.Add (rss, pktsRecvd);
  }

gnuplot.AddDataset (dataset);
```

Add data to dataset

Add dataset to plot

Resources

Web site:

http://www.nsnam.org

Mailing lists:

https://groups.google.com/forum/#!forum/ns-3-users
http://mailman.isi.edu/mailman/listinfo/ns-developers

Wiki:

http://www.nsnam.org/wiki/

Tutorial:

http://www.nsnam.org/docs/tutorial/tutorial.html

IRC: #ns-3 at freenode.net

Summary

- **NS3 is an emerging simulator to replace the previous one named NS2**
- **NS3 for those who are interested in:**
  - Open source and collaboration
  - More faithful representations of real computers and the Internet
  - Integration with testbeds
  - A powerful low-level API
  - Python scripting
  - Contest programmer

Thank You !