NS-3 Simulator Basics

- NS-3 is a network simulator
- Developed for network research and education
- Developed after ns-2
- ns-3 is written in C++
- Bindings in Python
- ns-3 uses the waf build system
- simulation programs are C++ executables or python scripts

Features

- It it a discrete event simulator
- Modular design / Open source
- Actively developed (Contrast NS-2)
- Developed in C++. Python binding available.
- Live visualizer
- Logging facility for debugging
- Tracing facility fo getting output
- Can be connected to a real network
- Direct Code Execution (DCE)

Discrete Event Simulator

- Simulation time moves discretely from event to event
- Schedule events to occur at specific simulation times
- A simulation scheduler orders the event execution
- Simulator execute events one by one
- Simulation stops at specific time or when events end

Installation

- Primary site www.nsnam.org
- Latest release 3.18
- date-driven schedules
- 3-4 months cycle
- Source download link:- https://www.nsnam.org/release/ns-allinone-3.18.tar.bz2
- Documentation download link:- http://www.nsnam.org/ns-3-18/documentation
- http://www.nsnam.org/docs/release/3.18/tutorial/ns-3-tutorial.pdf

Installation

- Download the source tar file
- Extract the tar file (A tour of src tree)
- Install c++, python
- From command line

  $cd nsallinone-3.18

  $./build.py --enable-examples –enable-tests
- Waf: Leaving directory ‘/path/to/workspace/ns-allinone-3.18/ns-3.18/build’ ’build’ finished successfully (6m25.032s)

Installation

- If using Ubuntu or Debian, there is a package named ns3. Install this package. You may need additional packages. Serach ns3 in Synaptic package manager and install ns3, libns3-3, libns3-dev, ns3-doc, python-ns3
- Other packages may be required for additional features

Run a Script

To test the installation copy one example available in the distribution to scratch directory and build and run the same using the commands below: -

$cd ns-3.18

$cp examples/tutorial/first.cc scratch/first1.cc

$waf –run first1

Output

Waf: Entering directory `/home/abhijit/workspace/ns-allinone-3.18/ns-3.18/build'

Waf: Leaving directory `/home/abhijit/workspace/ns-allinone-3.18/ns-3.18/build'

'build' finished successfully (1.006s)

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

Organization

Key Abstractions: -

- Node
- Application
- Net device
- Channel

  Corresponding to these abstractions there are modules (There are other modules for other functionalities)

Modules

Different entities are implemented in different modules. Each modules has 'Attributes'. Functionalities of modules are provided by different C++ member functions. There is helper API available with each module. These helper methods makes programming easier. Low level API can still be used.

- It is easy extend module to add new features/functionalities
- It is easy to add new module

Structure of NS-3 script

For simulation, we need to write a simulation script which is a C++ program. To this program the ns-3 library is linked to build our simulation executible. API calls are used in the program to do the necessary simulation. The waf build system is used to build the simulation.

Steps in writing scripts

- Include necessary files
- Use appropriate name space
- Set simulation time resolution(Optional)
- Enable logging for different modules(Optional)
- Create nodes
- Create net devices with MAC and PHY
- Attach Net devices to nodes and set interconnections

Steps in writing scripts

- Install protocol stack in nodes
- Set network address for interfaces
- Setup routing
- Install applications in nodes
- Setup tracing
- Set application start and stop time
- Set simulation start time
- Run simulation
- Release resources at end of simulation

Example

A simple point-to-point link connecting two hosts. A client application is instaled in one and a server application in the other.

```
#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;
```

Example

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

  Time::SetResolution (Time::NS);

  LogComponentEnable ("UdpEchoClientApplication",
                                        LOG_LEVEL_INFO);

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

Example

```
NodeContainer nodes;
nodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate",
                                 StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute("Delay",
                                 StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

Example

```
InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces =
                                    address.Assign (devices);
```

Example

```
UdpEchoServerHelper echoServer (9);

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

serverApps.Start (Seconds (1.0));

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

Example

```
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));
```

Example

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

Conversion

Conversion: -

(Serialize De serialize, refcount)

StringValue IntegerValue UintegerValue

BooleanValue PointerValue TimeValue

EnumValue AddressValue Ipv4AddressValue

Ipv4MaskValue Mac16AddressValue Mac48AddressValue

List of Modules

Aodv applications bridge click

Config-store core csma

csma-layout dsdv Emu energy

flow-monitor internet

Lte mesh mobility mpi

netanim network

List of Modules

Nix-vector-routing ns3tcp ns3wifi
olsr openflow point-to-point
point-to-point-layout propagation
spectrum stats tap-bridge test
tools topology-read uan
virtual-net-device visualizer
wifi wimax

•

Net Devices

- AlohaNoackNetDevice
- PointToPoint
- Bridge
- Csma
- Wifi
- WiMax
- BaseStationNetDevice
- SubscriberStation
- OpenFlowSwitch
- LoopBack
- Simple

Net Devices

- Virtual (for tunneling)
- Emu (Emulator net device, deprecated)
- Fd (File descriptor)
- Tap
- Uan (Underwater Acoustic)
- Lte (Long Term Evolution)
- Fake
- NonCommunicating
- Error

Applications

bulk-send (sends data as fast as possible)

on-off (On Off pattern)

packet-sink

udp-server (Receives UDP packets)

udp-client (UDP packet with seq no and time stamp)

udp-echo-server

udp-echo-client

v4ping (sends one ICMP ECHO request, reports the RTT)

ping6

radv(Router advertisement daemon)

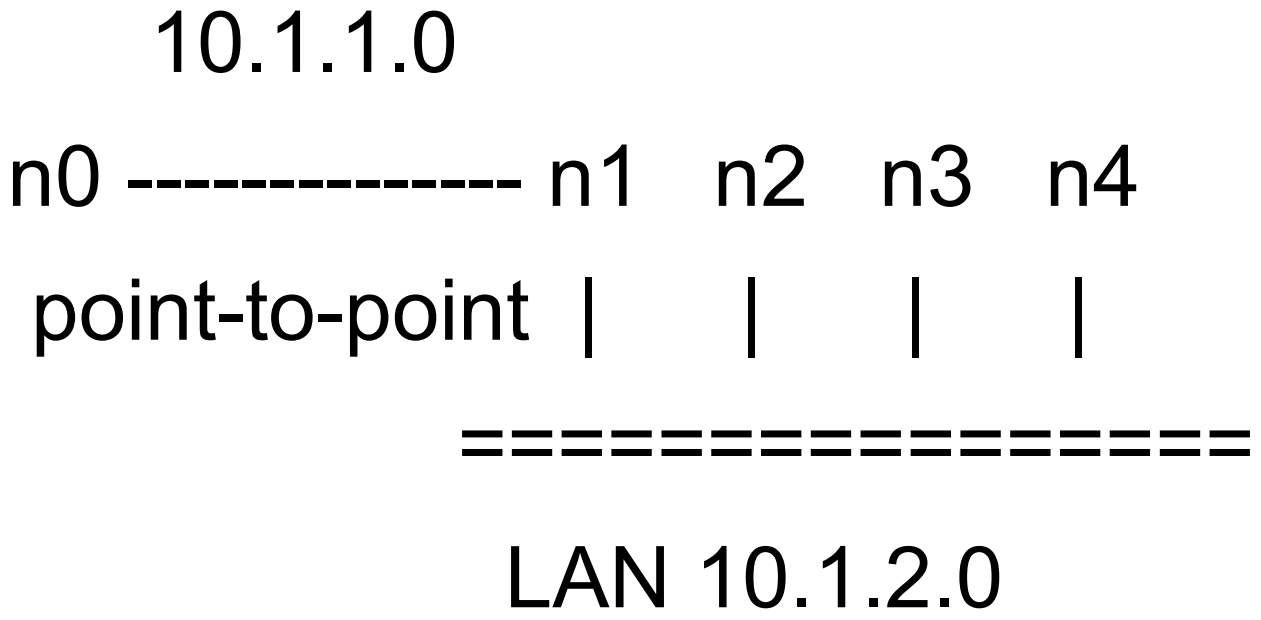
Routing protocols

- Ipv4StaticRouting (unicast and multicast)
- IPv4 Optimized Link State Routing (OLSR)
- IPv4 Ad Hoc On Demand Distance Vector (AODV)
- IPv4 Destination Sequenced Distance Vector (DSDV)
- Dynamic Source Routing (DSR)

Routing protocols

- pv4ListRouting (used to store a prioritized list of routing protocols)
- Ipv4GlobalRouting (used to store routes computed by the global route manager)
- Ipv4NixVectorRouting (a more efficient version of global routing)
- Ipv6ListRouting (used to store a prioritized list of routing protocols)
- Ipv6StaticRouting

Example 2



```
int main (int argc, char *argv[]) {
  bool verbose = true;
  uint32_t nCsma = 3;

  CommandLine cmd;
  cmd.AddValue ("nCsma", "Number of ", nCsma);
  cmd.AddValue ("verbose", "Log if true", verbose);

  cmd.Parse (argc,argv);

  nCsma = nCsma == 0 ? 1 : nCsma;
// ./waf --run "second --nCsma=3"
```

```
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));
```

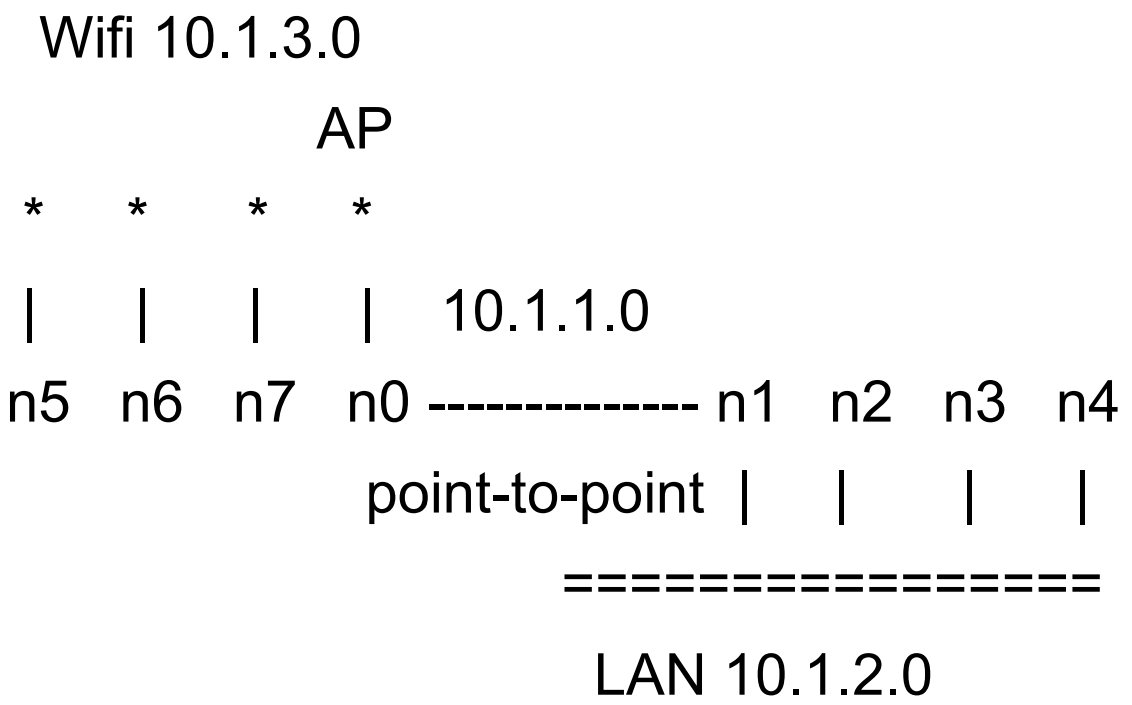
```
UdpEchoClientHelper echoClient
            (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute
              ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute
               ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute
               ("PacketSize", UintegerValue (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;
}
```

Example 3


Wifi 10.1.3.0
AP
* * * *
| | | | 10.1.1.0
n5 n6 n7 n0 ---------- n1 n2 n3 n4
point-to-point | | | |
==========
LAN 10.1.2.0

```
  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 = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelper wifi = WifiHelper::Default ();
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

NqosWifiMacHelper mac = NqosWifiMacHelper::Default ();
```

```
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));

pointToPoint.EnablePcapAll ("third");
phy.EnablePcap ("third", apDevices.Get (0));
csma.EnablePcap ("third", csmaDevices.Get (0), true);

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

Thanks

(If you are still listning)

It's Demo Time