NSCOM02

3rd Term– AY2020 – 2021

Instructor: Dr. Marnel Peradilla

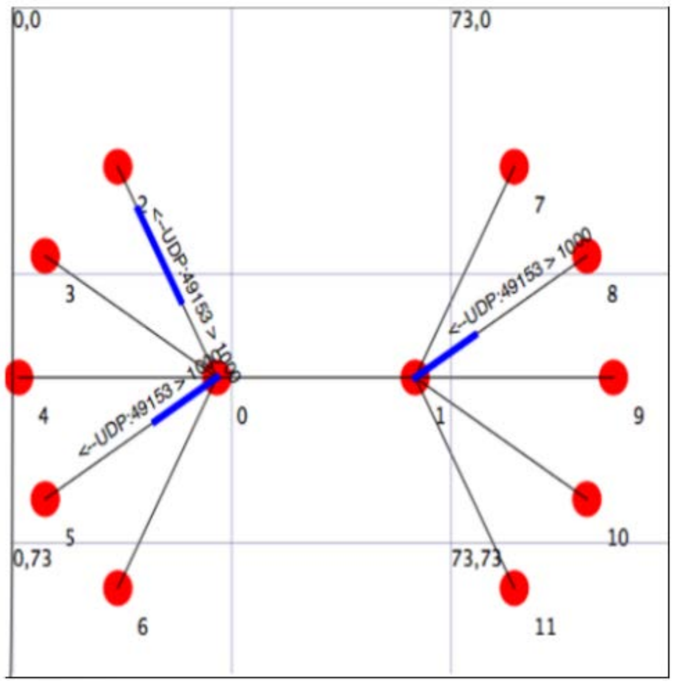
NS3 INTRODUCTION

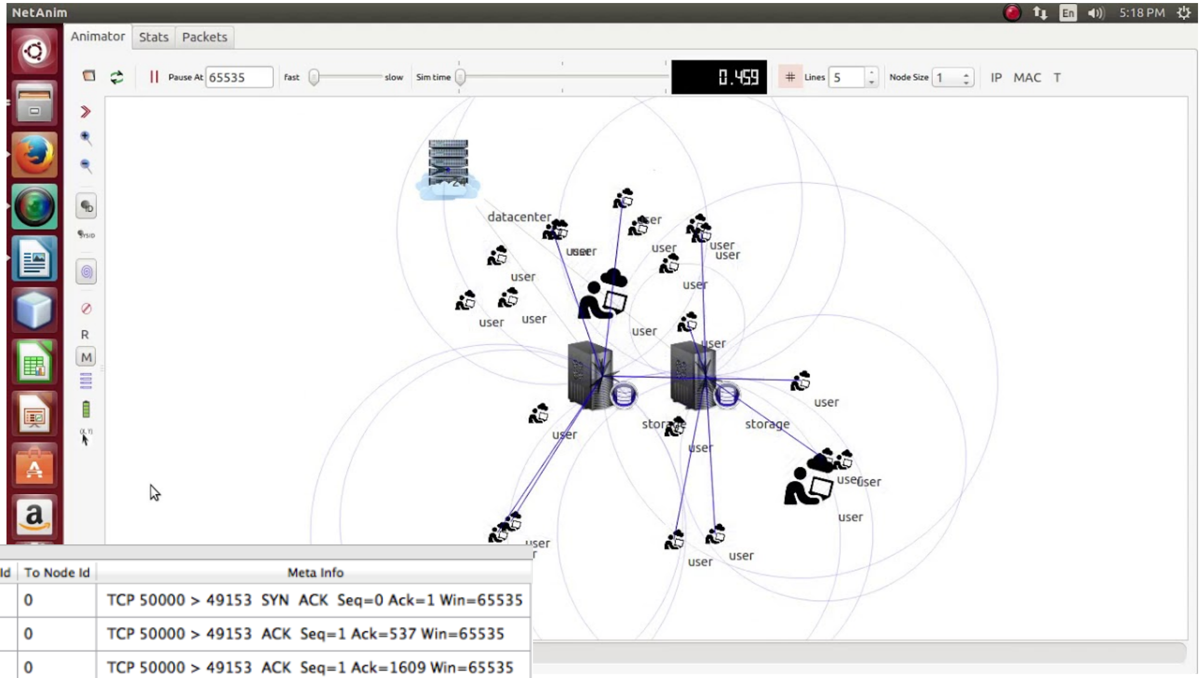
RESOURCES

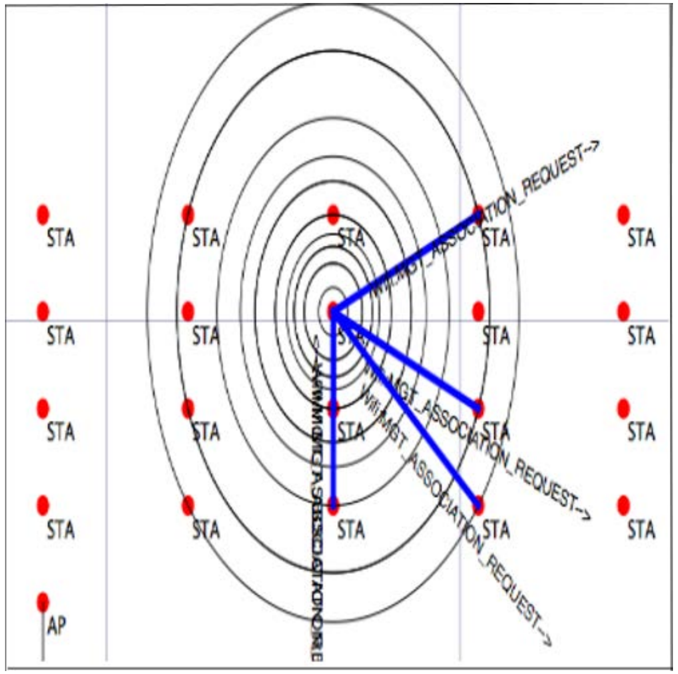
- **Linux**
- **Install via Terminal**
  - https://www.nsnam.org/wiki/Installation
- **https://www.nsnam.org/docs/tutorial/ns-3-tutorial.pdf**

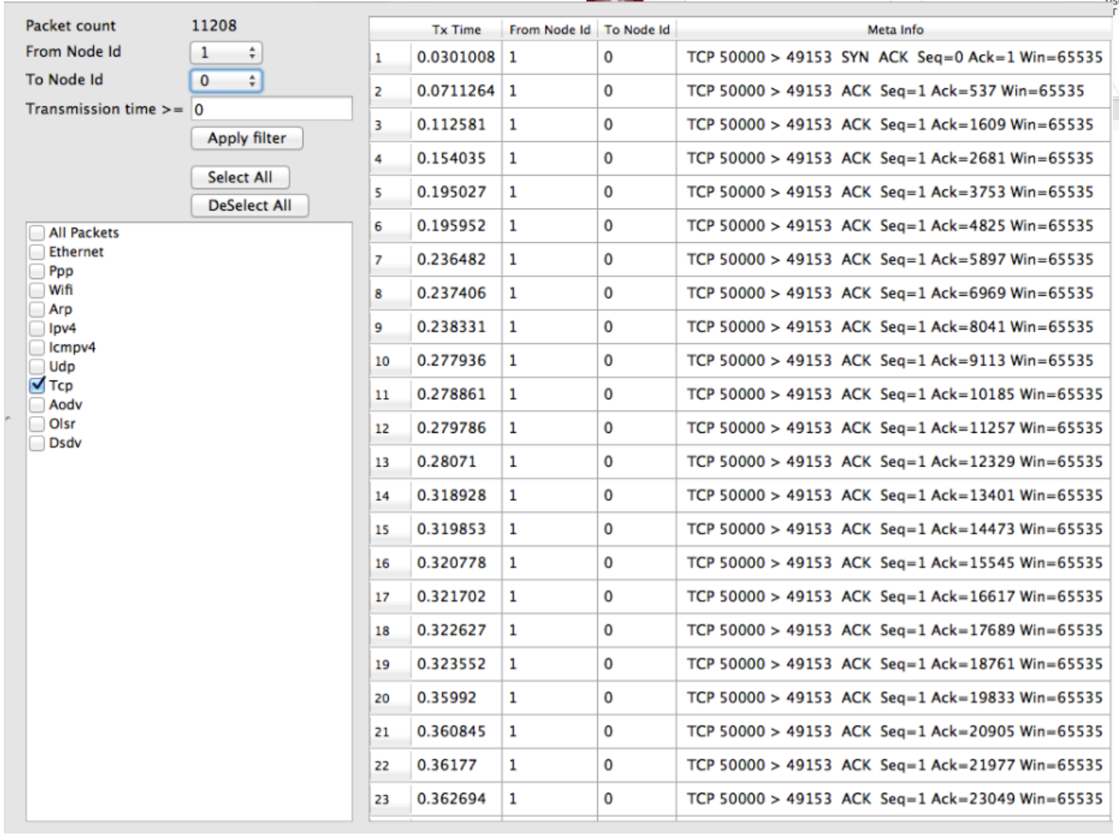
NS3

- **Discrete event network simulator**
- **Open source**
- **Collection of C++ Libraries, not a program**
- **Linux, FreeBSD, Cygwin, WSL**
- **Supports: external animators, data analysis, visualization tools**
  - gnuplot
  - Wireshark
  - NetAnim
  - Trace Metrics
  - Flow monitor
  - Visualizer









	Tx Time	From Node Id	To Node Id	Meta Info
1	0.0301008	1	0	TCP 50000 > 49153 SYN ACK Seq=0 Ack=1 Win=65535
2	0.0711264	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=537 Win=65535
3	0.112581	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=1609 Win=65535
4	0.154035	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=2681 Win=65535
5	0.195027	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=3753 Win=65535
6	0.195952	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=4825 Win=65535
7	0.236482	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=5897 Win=65535
8	0.237406	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=6969 Win=65535
9	0.238331	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=8041 Win=65535
10	0.277936	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=9113 Win=65535
11	0.278861	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=10185 Win=65535
12	0.279786	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=11257 Win=65535
13	0.28071	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=12329 Win=65535
14	0.318928	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=13401 Win=65535
15	0.319853	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=14473 Win=65535
16	0.320778	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=15545 Win=65535
17	0.321702	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=16617 Win=65535
18	0.322627	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=17689 Win=65535
19	0.323552	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=18761 Win=65535
20	0.35992	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=19833 Win=65535
21	0.360845	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=20905 Win=65535
22	0.36177	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=21977 Win=65535
23	0.362694	1	0	TCP 50000 > 49153 ACK Seq=1 Ack=23049 Win=65535

KEY TERMS AND ABSTRACTIONS

- ❑ **Node**
- ❑ **Application**
- ❑ **Channel**
- ❑ **Net Device**
- ❑ **Topology helpers**

NODE

- Basic computing device of ns3
- Name of the C++ class is ***Node***
- Represents a ***computer*** which can be added with peripherals, applications, protocol stacks and drivers to do useful work

APPLICATION

- run on ns-3 ***Node*** to drive simulations in the simulated world
- Represented in C++ by the class ***Application***
- Examples:
  - UdpEchoClientApplication
  - UdpEchoServerApplication

CHANNEL

- **Connects a Node to an object representing a communication channel**
- **The basic communication subnetwork abstraction is called the channel and is represented in C++ by the class *Channel***
- **Examples:**
  - CsmaChannel
  - PointToPointChannel
  - WifiChannel

NETDEVICE

- **Network Interface Cards implement networking function and install in the computer to connect to a network**
- **The net device abstraction covers both the software driver and the simulated hardware**
- **The abstraction: *NetDevice***
- **Examples;**
  - CsmaNetDevoce
  - PointToPointNetDevice
  - WifiNetDevice

TOPOLOGY HELPER

- **In a large simulated network you will need to arrange many connections between Nodes, NetDevices and Channels**
- **Topology Helpers: combines many distict ns-3 core operations into an easy to use model such as to**
  - Create a NetDevice
  - Add a MAC address
  - Install that net device on a Node
  - Configuration of the Node's protocol stack
  - Connect the NetDevice to a Channel
- **A helper has its own set of operations**
- **A container keeps track of multiple instances of objects**

INSTALLATION PREPARATION

```
# Netanim animator
apt install –y mercurial qt5-default
# Support for ns-3-pyviz visualizer
apt install -y gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz python3-gi python3-gi-cairo python3-pygraphviz
gir1.2-gtk-3.0 ipython ipython3
# Support for MPI
apt install -y openmpi-bin openmpi-common openmpi-doc libopenmpi-dev
# Support for bake build tool
apt install -y autoconf cvs bzr unrar
# Support for debugging
apt install -y gdb valgrind
# Doxygen and related inline documentation:
apt install -y doxygen graphviz imagemagick
apt install -y texlive texlive-extra-utils texlive-latex-extra texlive-font-utils texlive-lang-portuguese dvipng latexmk
# GNU Scientific Library (GSL) support for more accurate 802.11b WiFi error models (not needed for OFDM)
apt install -y gsl-bin libgsl-dev libgsl23 libgslcblas0
# To read pcap packet traces
apt install -y tcpdump
# Database support for statistics framework
apt install -y sqlite sqlite3 libsqlite3-dev
# Xml-based version of the config store (requires libxml2 >= version 2.7)
apt install -y libxml2 libxml2-dev
# Support for generating modified python bindings
apt install -y cmake libc6-dev libc6-dev-i386 libclang-6.0-dev llvm-6.0-dev automake
python3 -m pip install --user cxxfilt
python -m pip install --user cxxfilt
# A GTK-based configuration system
apt install -y libgtk2.0-0 libgtk2.0-dev
# To experiment with virtual machines and ns-3
apt install -y vtun lxc
# Support for openflow module (requires some boost libraries)
apt install -y libboost-signals-dev libboost-filesystem-dev
```

GETTING STARTED

- **ns-3 is built as a system of software libraries**
- **3-ways to download and build ns-3**
  - Download and build an official release from the main web site
  - Fetch and build dev copies of a basic ns-3 installation
    - git clone https://gitlab.com/nsnam/ns-3-allinone.git
    - python download.py [-n ns-3.30]
  - Use an additional build tool to download more extensions for ns-3
    - bake, waf

INSTALLATION

```
-> sudo apt update
-> sudo apt –y upgrade
-> apt install –y net-tools
-> mkdir workspace
-> cd workspace

-> wget https://www.nsnam.org/releases/ns-allinone-3.30.1.tar.bz2
-> tar tar xjf ns-allinone-3.30.1.tar.bz2
-> cd ns-allinone-3.30.1
-> ./build.py –enable-examples –enable-tests
```

https://www.nsnam.org/docs/tutorial/ns-3-tutorial.pdf

DIRECTORY HIERARCHY

…/ns-allinone-3.30.1/ns-3.30.1/ https://gitlab.com/nsnam/ns-3-dev

- build/
  - ns3/ include directory
  - lib/ library directory
  - src/ object files built from src/
  - examples/ object files built from examples/
  - utils/ object files built from utils/
  - scratch/ object files built from scratchs/
- doc/
  - manual/
  - tutorial/
- src/
  - core/, csma/, internet/, network/, lte/, …
  - applications/helper
- examples/
  - routing/, tcp/, udp/, ipv6/, wireless/, udp-client-server/, tutorial/, …
- utils/
- scratch/ for your scripts

./examples/tutorial/first.cc → ./build/examples/tutorial/first.cc.2.o
The ns-3 script is just a C++ program.

A First Script – first.cc

…/ns-allinone-3.30.1/ns-3.30.1/examples/tutorial/first.cc

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

#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);
```

Main Part

```
  Simulator::Stop (Seconds (11.0)); /* Good Practice: explicitly stop */
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

- The simulation will stop automatically when no further events are in the event queue, or when a special Stop event is found
- When there is a self sustaining event, *Simulator::Stop* is absolutely necessary to stop the simulation

first.cc – Main Part

```
  NodeContainer nodes;
  nodes.Create (2);
/*-- Prepare a simple point-to-point network */
  PointToPointHelper pointToPoint;   /* Create NetDevices and Channel. */
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
/* Build the network with 2 Nodes and get its devices */
  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);
/* Configure the 2 NetDevices as an IPv4 network */
  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");

  Ipv4InterfaceContainer interfaces = address.Assign (devices);
/* Run the Echo server application on Node 1 (UDP Port 9) */
  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));
/* Configure and Run the Echo client application on Node 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));
```

Building a Script and Run

```
$ cd ../..  …/ns-allinone-3.30.1/ns-3.30.1/
$ cp examples/tutorial/first.cc scratch/myfirst.cc
```

Now build your first example script using waf:

```
$ ./waf
```

Automatically build the scripts

You should see messages reporting that your `myfirst` example was built successfully.

```
Waf: Entering directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
[614/708] cxx: scratch/myfirst.cc -> build/debug/scratch/myfirst_3.o
[706/708] cxx_link: build/debug/scratch/myfirst_3.o -> build/debug/scratch/myfirst
Waf: Leaving directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (2.357s)
```

You can now run the example (note that if you build your program in the scratch directory you must run it out of the scratch directory):

```
$ ./waf --run scratch/myfirst
```

Execute the script

You should see some output:

```
Waf: Entering directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (0.418s)
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2
```

TWEAKING

- **Using the Logging Module**
  - Logging Overview
  - Enabling Logging
  - Adding Logging to your Code
- **Using Command Line Arguments**
  - Overriding Default Attributes
  - Hooking Your Own Values
- **Using the Tracing System**
  - ASCII Tracing
  - PCAP Tracing

LOGGING MODULE

- ns-3 provides
  - **Tracing** - a general purpose mechanism to get data out of your models which should be preferred for simulation output
  - **Logging** - preferred for debugging information, warnings, error messages, or any time you want to easily get a quick message out of your scripts or models
- Seven levels of log messages of increasing verbosity defined
  - LOG_ERROR — Log error messages (associated macro: NS_LOG_ERROR)
  - LOG_WARN — Log warning messages (associated macro: NS_LOG_WARN)
  - LOG_DEBUG — Log relatively rare, ad-hoc debugging messages (associated macro: NS_LOG_DEBUG)
  - LOG_INFO — Log informational messages about program progress (associated macro: NS_LOG_INFO)
  - LOG_FUNCTION — Log a message describing each function called (two associated macros: NS_LOG_FUNCTION, used for member functions, and NS_LOG_FUNCTION_NOARGS, used for static functions);
  - LOG_LOGIC – Log messages describing logical flow within a function (associated macro: NS_LOG_LOGIC)
  - LOG_ALL — Log everything mentioned above (no associated macro)

ENABLING LOGGING

- Additional macros
  - LOG_LEVEL_TYPE — enables logging of all the levels above it in addition to it's level
    - For example
      - Enabling LOG_INFO will only enable messages provided by NS_LOG_INFO macro,
      - Enabling LOG_LEVEL_INFO will also enable messages provided by NS_LOG_DEBUG, NS_LOG_WARN and NS_LOG_ERROR macros
  - NS_LOG_UNCOND — Log associated message unconditionally (no associated log level)
- Logging can be set up using a shell environment variable (NS_LOG) or by logging system function call, i.e.

```
$ export NS_LOG=UdpEchoClientApplication=level_info
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
```

LOG AND SHOW MORE

- Show every time a function in the application is called during script execution
  - Generally, use of (at least) **NS_LOG_FUNCTION(this)** in member functions is preferred
  - Use NS_LOG_FUNCTION_NOARGS() only in static functions
- Show every message prefixed with the component name

  $ export 'NS_LOG=UdpEchoClientApplication=level_all|**prefix_func**'
- Show every message prefixed with both the component name and the simulation time

  $ export 'NS_LOG=UdpEchoClientApplication=level_all|prefix_func|**prefix_time**:
  UdpEchoServerApplication=level_all|prefix_func|**prefix_time**'
- Turn on all components

  $ export 'NS_LOG=***=**level_all|prefix_func|prefix_time'

Adding Logging to your Code

```
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample")

int
main (int argc, char *argv[])
{
  :
  NS_LOG_INFO ("Creating Topology");
  :
}
```

- Turn off

  $ export NS_LOG=

- Enable LOG_INFO

  $ export NS_LOG=FirstScriptExample=info

  $ export NS_LOG=FirstScriptExample=level_info

USING COMMAND LINE ARGUMENTS

Overriding Default Attributes

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

- The snippet opens the door to the ns-3 global variable and Attribute systems

$ ./waf --run "scratch/myfirst --PrintHelp" # Pass the argument --PrintHelp to the script

```
Waf: Entering directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (0.413s)
TcpL4Protocol:TcpStateMachine()
CommandLine:HandleArgument(): Handle arg name=PrintHelp value=
--PrintHelp: Print this help message.
--PrintGroups: Print the list of groups.
--PrintTypeIds: Print all TypeIds.
--PrintGroup=[group]: Print all TypeIds of group.
--PrintAttributes=[typeid]: Print all attributes of typeid.
--PrintGlobals: Print the list of globals.
```

$ ./waf --run "scratch/myfirst --PrintAttributes=ns3::PointToPointNetDevice"

```
--ns3::PointToPointNetDevice::DataRate=[32768bps]:
  The default data rate for point to point links
```

USING COMMAND LINE ARGUMENTS

Set Attribute Values through the Command Line

- In the program myfirst.cc

```
/*-- Prepare a simple point-to-point network */
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  ...
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
```

- It can be done by the command line argument

```
$ ./waf --run "scratch/myfirst
  --ns3::PointToPointNetDevice::DataRate=5Mbps
  --ns3::PointToPointChannel::Delay=2ms
  --ns3::UdpEchoClient::MaxPackets=2"
```

- Show available TypeId names and attributes of a specific group

```
$ ./waf --run "scratch/myfirst --PrintGroup=PointToPoint"
TypeIds in group PointToPoint:
  ns3::PointToPointChannel
  ns3::PointToPointNetDevice
  ns3::PointToPointRemoteChannel
  ns3::PppHeader
$ ./waf --run "scratch/myfirst --PrintAttributes=ns3::PointToPointChannel"
```

https://www.nsnam.org/doxygen/_attribute_list.html

USING COMMAND LINE ARGUMENTS

Show Available TypeId Names and Attributes

```
nuk@ubuntu: ~/ns3/ns-allinone-3.30.1/ns-3.30.1
File  Edit  View  Search  Terminal  Help
nuk@ubuntu:~/ns3/ns-allinone-3.30.1/ns-3.30.1$ ./waf --run "scratch/myfirst --PrintHelp"
Waf: Entering directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Waf: Leaving directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.995s)
myfirst [General Arguments]

General Arguments:
    --PrintGlobals:              Print the list of globals.
    --PrintGroups:               Print the list of groups.
    --PrintGroup=[group]:        Print all TypeIds of group.
    --PrintTypeIds:              Print all TypeIds.
    --PrintAttributes=[typeid]:  Print all attributes of typeid.
    --PrintHelp:                 Print this help message.

nuk@ubuntu:~/ns3/ns-allinone-3.30.1/ns-3.30.1$ ./waf --run "scratch/myfirst --PrintGroup=PointToPoint"
Waf: Entering directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Waf: Leaving directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.001s)
TypeIds in group PointToPoint:
    ns3::PointToPointChannel
    ns3::PointToPointNetDevice
    ns3::PointToPointRemoteChannel
    ns3::PppHeader
nuk@ubuntu:~/ns3/ns-allinone-3.30.1/ns-3.30.1$ ./waf --run "scratch/myfirst --PrintAttributes=ns3::PointToPointChannel"
Waf: Entering directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Waf: Leaving directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.999s)
Attributes for TypeId ns3::PointToPointChannel
    --ns3::PointToPointChannel::Delay=[+0.0ns]
        Propagation delay through the channel
nuk@ubuntu:~/ns3/ns-allinone-3.30.1/ns-3.30.1$
```

USING COMMAND LINE ARGUMENTS

Hooking Your Own Values

- Add your own hooks to the command line system by AddValue, i.e. nPackets

```
int
main (int argc, char *argv[])
{
  uint32_t nPackets = 1;    /* Declare the variable type with the default value */

  CommandLine cmd;
  cmd.AddValue("nPackets", "Number of packets to echo", nPackets);
  cmd.Parse (argc, argv);
  :
  echoClient.SetAttribute ("MaxPackets", UintegerValue (nPackets));

/*echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); replacing the constant */
```

```
$ ./waf --run "scratch/myfirst --PrintHelp"

$ ./waf --run "scratch/myfirst --nPackets=2"
```

USING COMMAND LINE ARGUMENTS

Demonstration of Command Line Hooking

```
nuk@ubuntu: ~/ns3/ns-allinone-3.30.1/ns-3.30.1
File  Edit  View  Search  Terminal  Help
nuk@ubuntu:~/ns3/ns-allinone-3.30.1/ns-3.30.1$ ./waf --run "scratch/myfirst --PrintHelp"
Waf: Entering directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
[2751/2822] Compiling scratch/myfirst.cc
[2781/2822] Linking build/scratch/myfirst
Waf: Leaving directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (3.695s)
myfirst [Program Options] [General Arguments]

Program Options:
    --nPackets:  Number of packets to echo [1]

General Arguments:
    --PrintGlobals:              Print the list of globals.
    --PrintGroups:               Print the list of groups.
    --PrintGroup=[group]:        Print all TypeIds of group.
    --PrintTypeIds:              Print all TypeIds.
nuk@ubuntu:~/ns3/ns-allinone-3.30.1/ns-3.30.1$ ./waf --run "scratch/myfirst --nPackets=2"
Waf: Entering directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Waf: Leaving directory `/home/nuk/ns3/ns-allinone-3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.001s)
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
At time 3s client sent 1024 bytes to 10.1.1.2 port 9
At time 3.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 3.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 3.00737s client received 1024 bytes from 10.1.1.2 port 9
nuk@ubuntu:~/ns3/ns-allinone-3.30.1/ns-3.30.1$
```

USING TRACING SYSTEM

ns-3 Tracing System

- The basic goals of the ns-3 tracing system
  - For ***basic*** tasks, the tracing system should allow the user to generate standard tracing for popular tracing sources, and to customize which objects generate the tracing;
  - ***Intermediate*** users must be able to extend the tracing system to modify the output format generated, or to insert new tracing sources, without modifying the core of the simulator;
  - ***Advanced*** users can modify the simulator core to add new tracing sources and sinks
- The ns-3 tracing system is built on the concepts of
  - Independent tracing sources and tracing sinks, and
    - Trace sources are entities that can
      - Signal events that happen in a simulation and
      - Provide access to interesting underlying data
    - Trace sinks are consumers of the events and data provided by the trace sources
  - A uniform mechanism for connecting sources to sinks
    - A user could define a new tracing sink in her script and attach it to an existing tracing source defined in the simulation core by editing only the user script

USING TRACING SYSTEM

ASCII Tracing

- **AsciiTraceHelper**: ns-3 provides helper functionality that wraps the low-level tracing system to help you with the details involved in configuring some easily understood packet traces

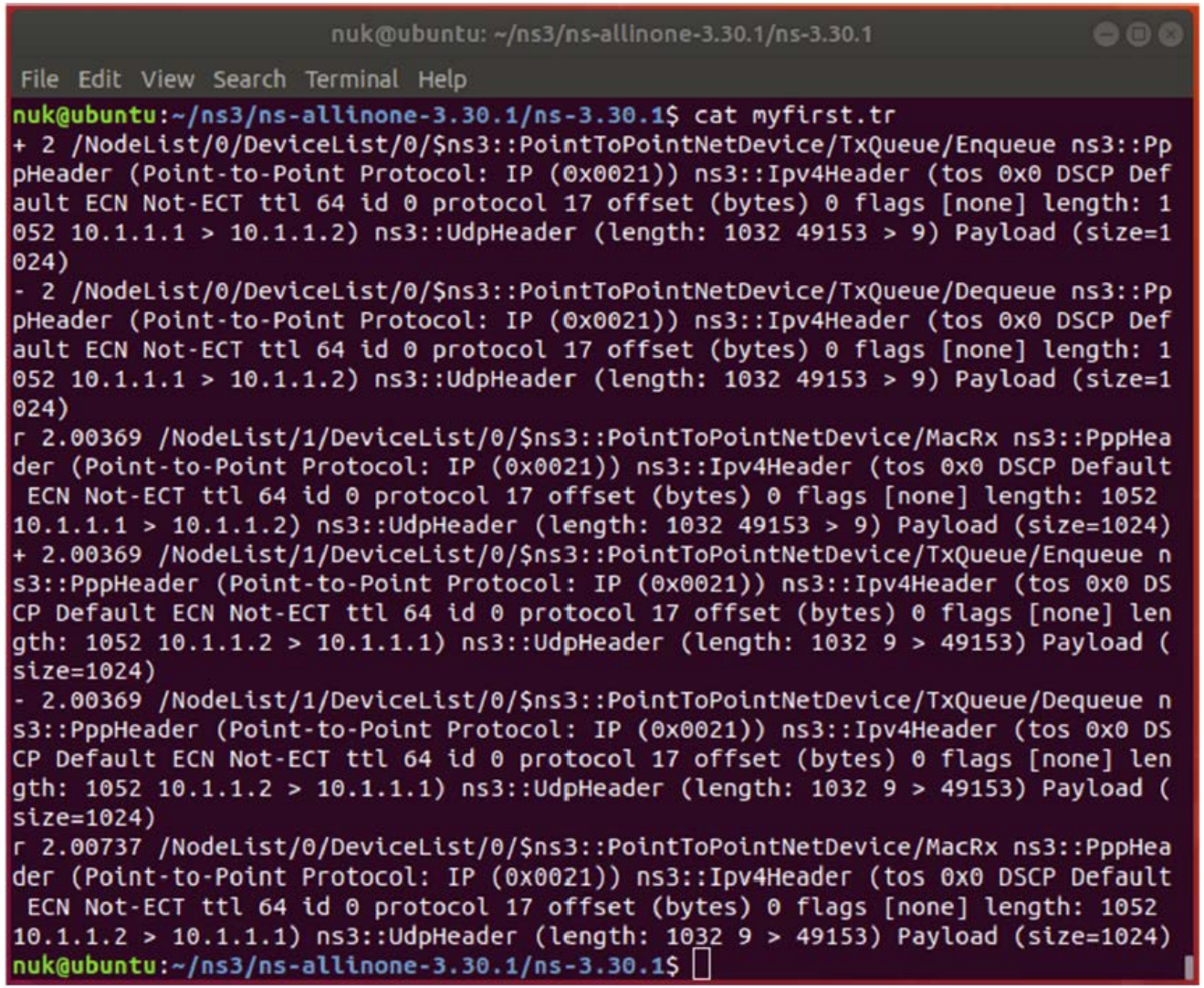
```
int
main (int argc, char *argv[])
{
  :
  AsciiTraceHelper ascii;
  pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("myfirst.tr"));
  Simulator::Run ();
  Simulator::Destroy ();
  :
}
```

```
$ ./waf --run scratch/myfirst

  (output trace file: ~/ns3/ns-allinone-3.30.1/ns-3.30.1/myfirst.tr)
```

USING TRACING SYSTEM

ASCII Trace File: myfirst.tr



Trace Event Example: Line #1 (Enqueue)

```
1  +
2  2
3  /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue
4  ns3::PppHeader (
5    Point-to-Point Protocol: IP (0x0021))
6    ns3::Ipv4Header (
7      tos 0x0 ttl 64 id 0 protocol 17 offset 0 flags [none]
8      length: 1052 10.1.1.1 > 10.1.1.2)
9      ns3::UdpHeader (
10       length: 1032 49153 > 9)
11       Payload (size=1024)
```

- Each line corresponds to a trace event
- Begin with a long character
  - Section 1: Operation
    +: An enqueue operation occurred on the device queue
    -: A dequeue operation occurred on the device queue
    d: A packet was dropped, typically because the queue was full
    r: A packet was received by the net device
  - Section 2: Timestamp 2 (seconds)
  - Section 3: Node ID, Device ID, Device Type, Queue Name, Queue Operation
    - /NodeList/0/: Node 0
    - /DeviceList/0/: Device 0 of the node (Node 0)
    - $ns3::PointToPointNetDevice: kind of the device
    - TxQueue/Enqueue: enqueue to TxQueue (the operation to the queue)
  - Section 4-11: Protocol details

Trace Event Example: Line #3 (Reception)

```
1  r
2  2.25732
3  /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/MacRx
4    ns3::Ipv4Header (
5      tos 0x0 ttl 64 id 0 protocol 17 offset 0 flags [none]
6      length: 1052 10.1.1.1 > 10.1.1.2)
7      ns3::UdpHeader (
8        length: 1032 49153 > 9)
9        Payload (size=1024)
```

- The packet being received by net device 0 on node 1 with the echo server
  - Section 1: ‘r’ indicates that a packet was received by the net device
  - Section 2: Timestamp 2.25732 (seconds)
  - Section 3: Node ID, Device ID, Device Type, Reception
    - /NodeList/1/: Node 1
    - /DeviceList/0/: Device 0 of the node (Node 1)
    - $ns3::PointToPointNetDevice: kind of the device
    - MacRx: Receive from MAC layer
  - Sections 4-9: Protocol details

PCAP Tracing

- ***.pcap*** file format for tcpdump, Wireshark, etc

```
int
main (int argc, char *argv[])
{
  :
  AsciiTraceHelper ascii;
  pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("myfirst.tr"));
  pointToPoint.EnablePcapAll ("myfirst");
  Simulator::Run ();
  Simulator::Destroy ();
  :
}
```

```
$ ./waf --run scratch/myfirst

 (output trace files: ~/ns3/ns-allinone-3.30.1/ns-3.30.1/
                      myfirst-0-0.pcap       Node 0, Device 0
                      myfirst-1-0.pcap       Node 1, Device 0
```

Reading Output with Tcpdump

```
$ tcpdump -nn -tt -r myfirst-0-0.pcap
reading from file myfirst-0-0.pcap, link-type PPP (PPP)
2.000000 IP 10.1.1.1.49153 > 10.1.1.2.9: UDP, length 1024
2.514648 IP 10.1.1.2.9 > 10.1.1.1.49153: UDP, length 1024

tcpdump -nn -tt -r myfirst-1-0.pcap
reading from file myfirst-1-0.pcap, link-type PPP (PPP)
2.257324 IP 10.1.1.1.49153 > 10.1.1.2.9: UDP, length 1024
2.257324 IP 10.1.1.2.9 > 10.1.1.1.49153: UDP, length 1024
```

Note: tcpdump must be installed first (check installation of tcpdump in Ubuntu)

MESSAGE FROM DPO

"The information and data contained in the online learning modules, such as the content, audio/visual materials or artwork are considered the intellectual property of the author and shall be treated in accordance with the IP Policies of DLSU. They are considered confidential information and intended only for the person/s or entities to which they are addressed. They are not allowed to be disclosed, distributed, lifted, or in any way reproduced without the written consent of the author/owner of the intellectual property."