

Last update: 10/20/2016

NS3 Simulator Installation & Overview

By Adil Alsuhaimeh

(aalsuha@g.clemson.edu)

NS3 Simulation

- Open Source Network Simulation Tool
- Allows coding in C++ & Python
 - Good knowledge of Object-Oriented Programming is recommended.
- Can simulate simple and complex networking scenarios.

NS3 Download & Installation

- **MacOS**

- Make sure your OSX updated.
- Install xCode tools from the App Store
 - Install *command-line tools* option.
 - `xcode-select -install`
 - Install MacPorts (<http://MacPorts.org>) & HomeBrew tools (<http://Brew.sh>)
 - Install dependencies (more on that)

- **Ubuntu**

- Install the following pre-requisites
 - `apt-get install build-essential`
 - `apt-get install mercurial`
- More details here (and later in slides)
<https://www.nsnam.org/wiki/Installation#Ubuntu.2FDebian>

NS3 Download & Installation

- **Windows**

- Method 1

- Install Oracle VirtualBox.
 - Install Ubuntu on your virtual machine
 - Follow instruction for Ubuntu.

- Method 2 (Windows 10 Anniversary)

- Windows Subsystem for Linux install *Bash for Ubuntu in Windows 10*.
 - *Instructions [here](#)*
 - Follow instructions for Ubuntu.
 - You may want to install [VcXsrv Windows X Server](#) or [Xming X Server](#) if you want windowed applications

Pre-requisites for Ubuntu

- To satisfy all pre-requisites for Ubuntu, run the following command as root

```
sudo apt-get install build-essential
```

```
sudo apt-get install gcc g++ python python-dev qt4-dev-  
tools libqt4-dev mercurial bzip2 cmake libc6-dev libc6-dev-  
i386 g++-multilib libgtk2.0-0 libgtk2.0-dev graphviz  
python-pygraphviz python-kiwi python-pygoocanvas  
libgoocanvas-dev ipython cvs git p7zip-full unzip xz-utils
```

Pre-Requisites for Mac OS

- Install Xcode tools, MacPort & HomeBrew tools (as mentioned earlier)
- Check if you have Python installed by running this command in the terminal:
 - `Python --version`
 - You should have Python 2.7 (or some version earlier than Python 3)
- You need to use MacPorts to install the following dependencies.
 - `sudo port install bzip unzip unrar xz cvs autoconf graphviz py-pygraphviz py27-gooocanvas py27-ipython py27-kiwi py-pygtk py27-setuptools py27-pygtk py27-pygtk gtk2 gtk3`

More pre-requisites for Mac OS

- Use HomeBrew tools to install QT4 for Mac.
`>brew install cartr/qt4/qt`
 - This will take a while, and disk space. This installs a version of QT4 that is patched to work on El Capitan & Sierra.
- Test that you have QT4 installed by running
 - `qmake -version`
- QT4 is an optional component and is needed for NetAnim tool.

NS3 Download & Install

- **Method 1**

- Download tarball source from nsnam.org
 - <https://www.nsnam.org/release/ns-allinone-3.26.tar.bz2>
- Compile sources: Two options:
 - `./configure` & then `make`
 - `./build.py` & then `make`
- You may have to “build” the netanim tool later, but you will need qt4 libraries for that.

- **Method 2**

- Use Mercurial to download latest stable code

```
hg clone http://code.nsnam.org/ns-3-allinone
```

NS3 Download & Install

- **Method 3 : Bake tool**

- Download Bake tool using Mercurial

- hg clone <http://code.nsnam.org/bake>

- Go into bake directory

- Run `bake.py` to check for dependencies.

- `./bake.py check`

- Configure what you want from bake to download (we want NS-3.26)

- `./bake configure -e ns-3.26`

- This creates a `bakefile.xml` that will be used for download & installation.

- If you want to pick & choose packages specifically, you can check the file `bakeconf.xml`

Bake tool (contd. From last slide)

- Show what's going to be installed, and see if anything is needed
 - `./bake.py show`
 - On Mac, you will probably see that python-dev & QT4 are missing. That's ok. Do not worry about that.
- Download
 - `./bake.py download`
- Build
 - `./bake.py build`
- You can alternatively use `deploy` to download & build in one go
 - `./bake.py deploy`

Notes on Windows 10 Anniversary

- When you use [Bash On Ubuntu On Windows](#), you only have a console only with no ability to run applications that has GUI windows.
- Install an X Window Server application like XMING or XcXsrv.
- In Bash, run the command `export DISPLAY=:0`
- Alternatively, you can add it as a line in `~/ .bashrc` if the file doesn't exist create it.

Notes on Mac

- Install XQuartz to run visualization tools
- You may need to make a small hack on Mac OS to get pybindgen working.
- When you run `./waf configure` You may find that Python Bindings & PyViz are not enabled.
- You will see something like this:

```
Checking for pybindgen version          : 0.17.0.post41+ngd10fa60.d20161017
pybindgen (found '0.17.0.post41+ngd10fa60.d20161017'), (need
'0.17.0.post41+ngd10fa60')
```

Mac OS solution

- We obviously have the correct version here, but string `.d20161017` is confusing ns3
- We will remove this string from the file under
 - `../pybindgen/pybindgen/version.py`
 - Change the version string by removing `.d20161017` from the version string.
 - Re-run `./waf configure` from the ns3 directory.
- This should solve the problem!

- The file `version.py` looks like this

```
# coding: utf-8
# file generated by setuptools_scm
# don't change, don't track in version control
version = '0.17.0.post41+ngd10fa60.d20161017'
```

- We will change the version string by removing the date.
- This trick seems to work!

NS3 Directory Structure

- This is how the directory would look like when you install all needed components
- If some are missing, you may add them again
- Below, directories are shown in blue.

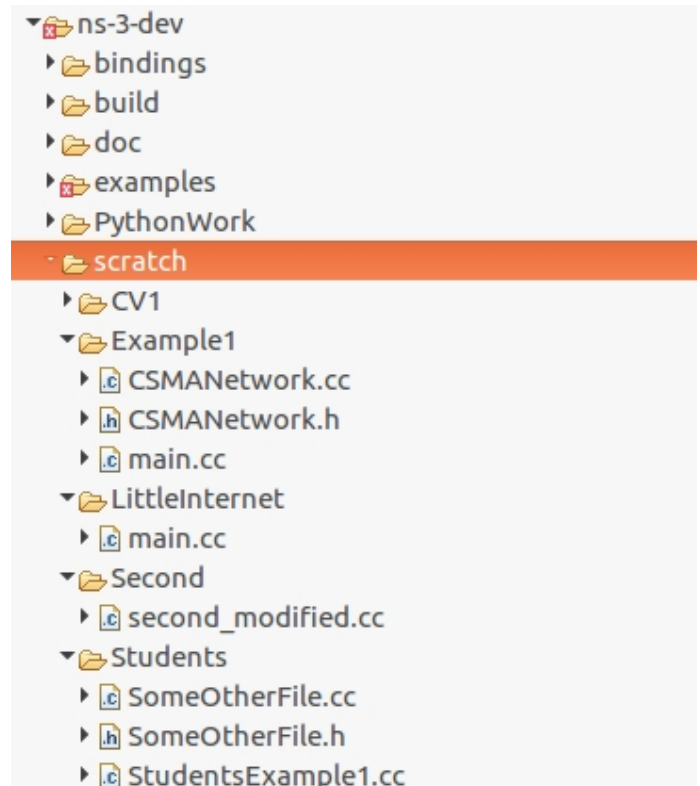
```
Adils-MBP:source adil$ ls
g++                pygoocanvas
gccxml             pygraphviz
netanim-3.107     python-dev
ns-3.26            pyviz-prerequisites
pybindgen         qt4
pygccxml          setuptools
pygccxml-1.0.0.zip
Adils-MBP:source adil$
```

NS3 Directory structure

- For example, you may install qt4 later, and then go into `netanim-3.107` and run `make` (or `qmake`)
- We will run `./waf configure` from the `ns-3.26` directory.
- Projects should be placed under `scratch` directory in their respective folders.

Project file placement

- Place your work under `ns-3xyz/scratch`
- Create a separate folder for each project you are working on.
 - Current Projects:
 - CV1
 - Example1
 - LittleInternet
 - Second
 - Students
 - ...
- Use `.cc` as file extension!



NS3 Projects

- Once you have a directory for your project, you may have multiple files for your development
- This includes `.h` files & `.cc` files.
- You may create C++ for your components or to extend existing ns3 components.
- One of your `.cc` files should contain a `main` function. This is the one used by NS3 to start execution.

Compiling & Running Examples

WAF Tool

- This is your `ns3` make tool.
- Execute `./waf` command on top `ns3` directory
 - Top directory, in my case, is `ns-3.26`
`~/workspace/NS3Work/ns-3.26# ./waf`
- `./waf` goes over all projects under `scratch` and compiles them then links them. Executables will be placed in the `build` directory.
- To run the `scratch-simulator` project we type
`./waf -run scratch-simulator`
- There's another version of Scratch Simulator under `subdir`, we run this like this:
`./waf -run subdir`
- For our projects, we will create directories under `scratch`, and give them whatever name we want.

WAF Debugging

- If you use a C++ debugger like gdb or llvm, you can debug your NS3 projects.

- For GDB

- ```
./waf --run <prog-name> --command-template "gdb -args %s"
```

- For LLVM

- ```
./waf --run <prog-name> --command-template "lldb %s"
```

Using Eclipse CDT

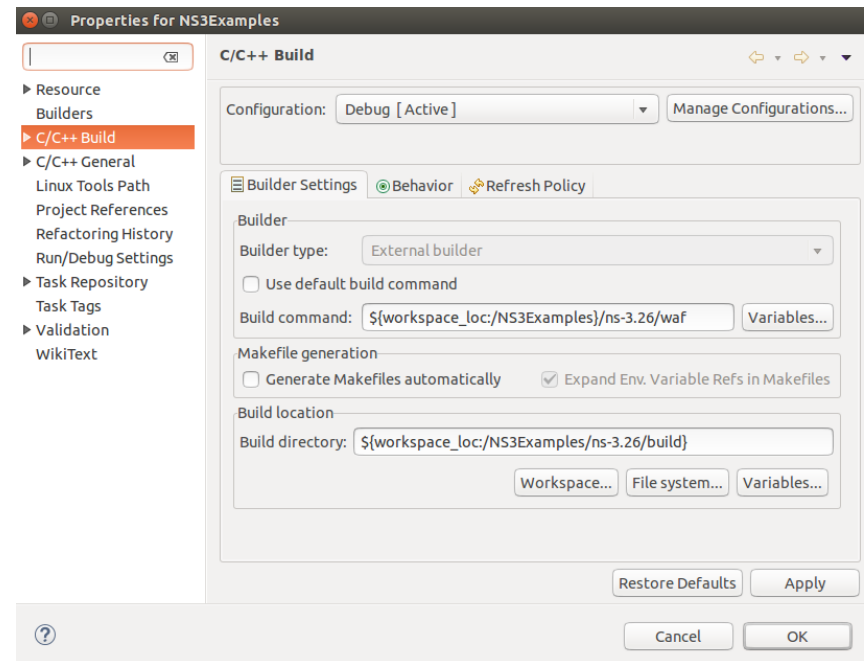
- If you prefer to use Eclipse to edit your C++ code
 - Make sure you have a Java Runtime Environment installed on your system
 - Download eclipse from eclipse.org
 - We want C/C++ Development Tools CDT so that we can work with C++ code.

Using Eclipse CDT

- Create a new C++ Project. Give it any name you want.
- Import “File System” and import “bake/source” directory
 - This is the directory listed in slide 12. We want to import all of these
- Now, go to ns-3.26, and see if you have the “goodies” that Eclipse offers, such as syntax highlighting, auto-completion and so on
 - If that didn’t work. You may want to remove the import, and import one directory higher or lower.

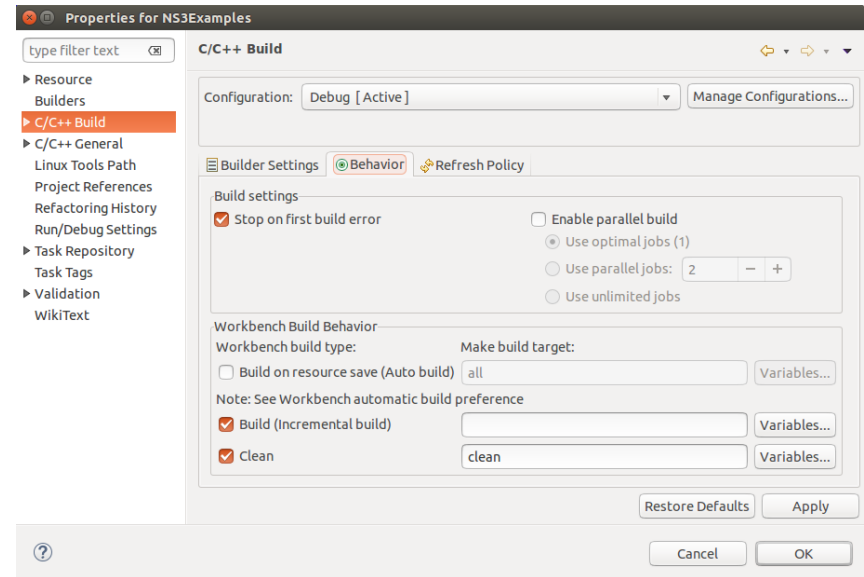
Eclipse Configuration

- Right Click on your project, select “Properties”
- From C/C++ Build
 - Uncheck “Use default build command”. Set it to waf within your workspace
 - Pick a build directory (preferably under ns-3.26/build)



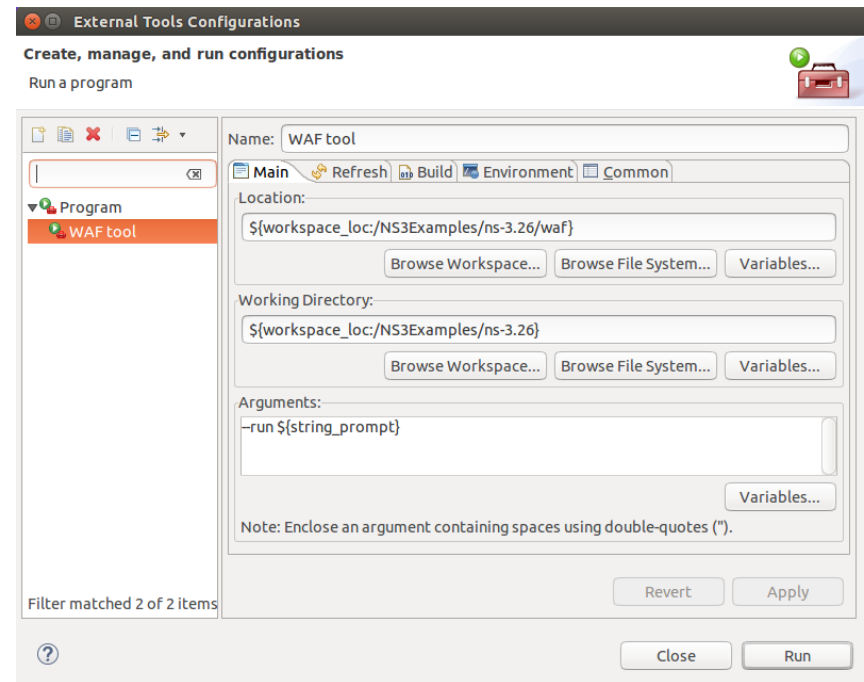
Eclipse Configuration

- In the Behavior Tab, remove all form “Build”
 - This is because we use the command `./waf` with no parameters to build.
 - There is a `./waf clean` command, so we leave that as it is.



Eclipse Configuration

- In Eclipse, we will pick the “Run” menu:
 - Run > External Tools > External Tools Configurations
- We will create a new program, Let’s call it WAF Tool
- Under location we will pick the waf file.
- Under “Working Directory” we will pick the ns3 directory where the waf file is.
 - For simplicity we can click on “Browse Workspace” button and choose the file through the GUI.
- In arguments type
 - `--run ${string_prompt}`
 - Is an eclipse variable. We can browse variables by clicking “Variables” button.
 - We may also add other arguments if we need
- If we’re running PyViz, we can type.
 - `--run ${string_prompt} --vis`
- The `${string_prompt}` will cause Eclipse to prompt user to enter the name of the project to run



Eclipse Configuration

- Now go in the terminal to the directory of your Eclipse project and under the ns3 directory.
- Run `./waf configure`
 - This is needed before we can run anything
- There are other `waf` files in other directories.
- Make sure you work with the one under `ns-3.26`, or `ns-3-dev`, depending on which method or version you used to install ns3.

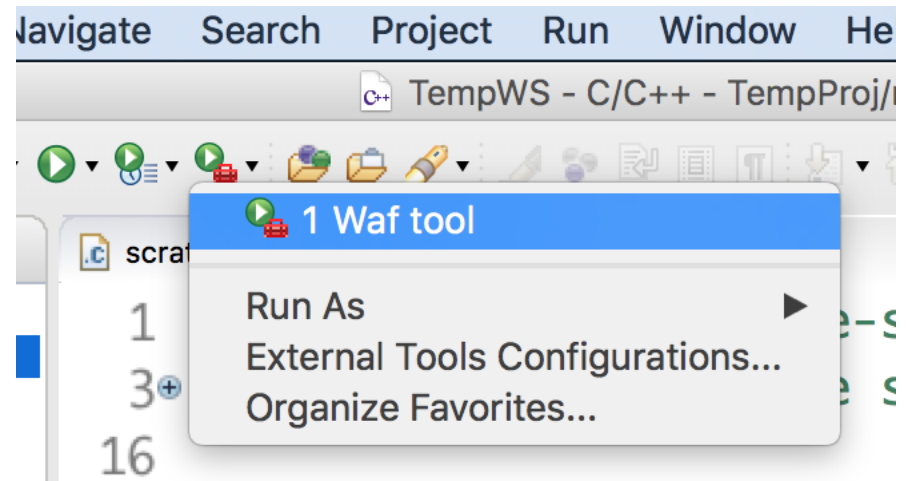
Another method to configure NS3 for Eclipse

- This method might be old or out of date, but worth a look.
 - [HOWTO configure Eclipse with ns-3](#)

Eclipse – Running Projects

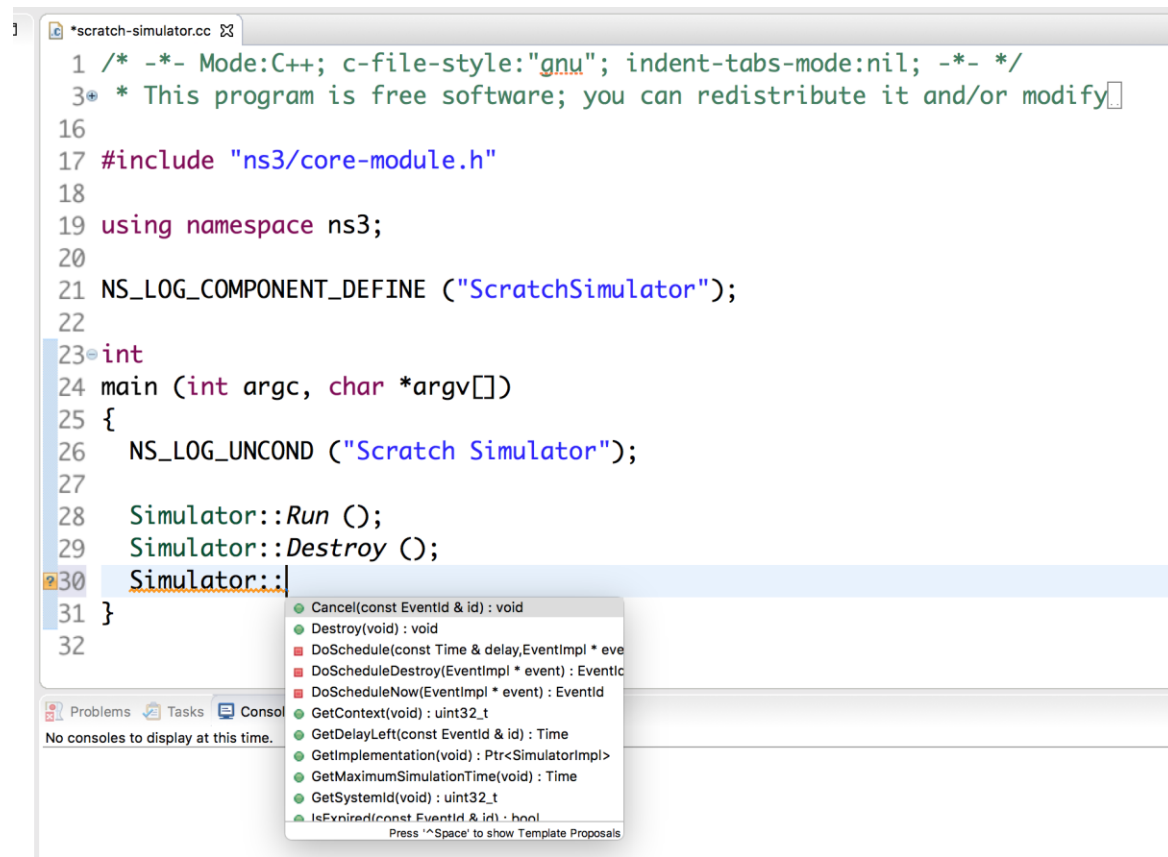
- Click on External Tools icon and then click on your tool to run!

- T



Eclipse

- Syntax Highlighting
- Auto-Completion



The screenshot shows the Eclipse IDE interface with a C++ source file named `*scratch-simulator.cc`. The code is displayed with syntax highlighting: comments are green, preprocessor directives are blue, namespace declarations are purple, and function names are black. The code includes a header file `ns3/core-module.h` and uses the `ns3` namespace. The `main` function calls `Scratch Simulator::Run` and `Scratch Simulator::Destroy`. At line 30, the text `Scratch Simulator::` is entered, and an auto-completion popup is visible, listing various methods of the `Scratch Simulator` class, such as `Cancel`, `Destroy`, `DoSchedule`, `DoScheduleNow`, `GetContext`, `GetDelayLeft`, `GetImplementation`, `GetMaximumSimulationTime`, `GetSystemId`, and `IsExpired`. The IDE also shows a 'Problems' tab and a 'Tasks' tab at the bottom.

```
1 /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
3* * This program is free software; you can redistribute it and/or modify
16
17 #include "ns3/core-module.h"
18
19 using namespace ns3;
20
21 NS_LOG_COMPONENT_DEFINE ("Scratch Simulator");
22
23 int
24 main (int argc, char *argv[])
25 {
26     NS_LOG_UNCOND ("Scratch Simulator");
27
28     Simulator::Run ();
29     Simulator::Destroy ();
30     Simulator::
31 }
32
```

- Cancel(const EventId & id) : void
- Destroy(void) : void
- DoSchedule(const Time & delay, EventImpl * eve
- DoScheduleDestroy(EventImpl * event) : Eventic
- DoScheduleNow(EventImpl * event) : EventId
- GetContext(void) : uint32_t
- GetDelayLeft(const EventId & id) : Time
- GetImplementation(void) : Ptr<SimulatorImpl>
- GetMaximumSimulationTime(void) : Time
- GetSystemId(void) : uint32_t
- IsExpired(const EventId & id) : bool

Press '^Space' to show Template Proposals

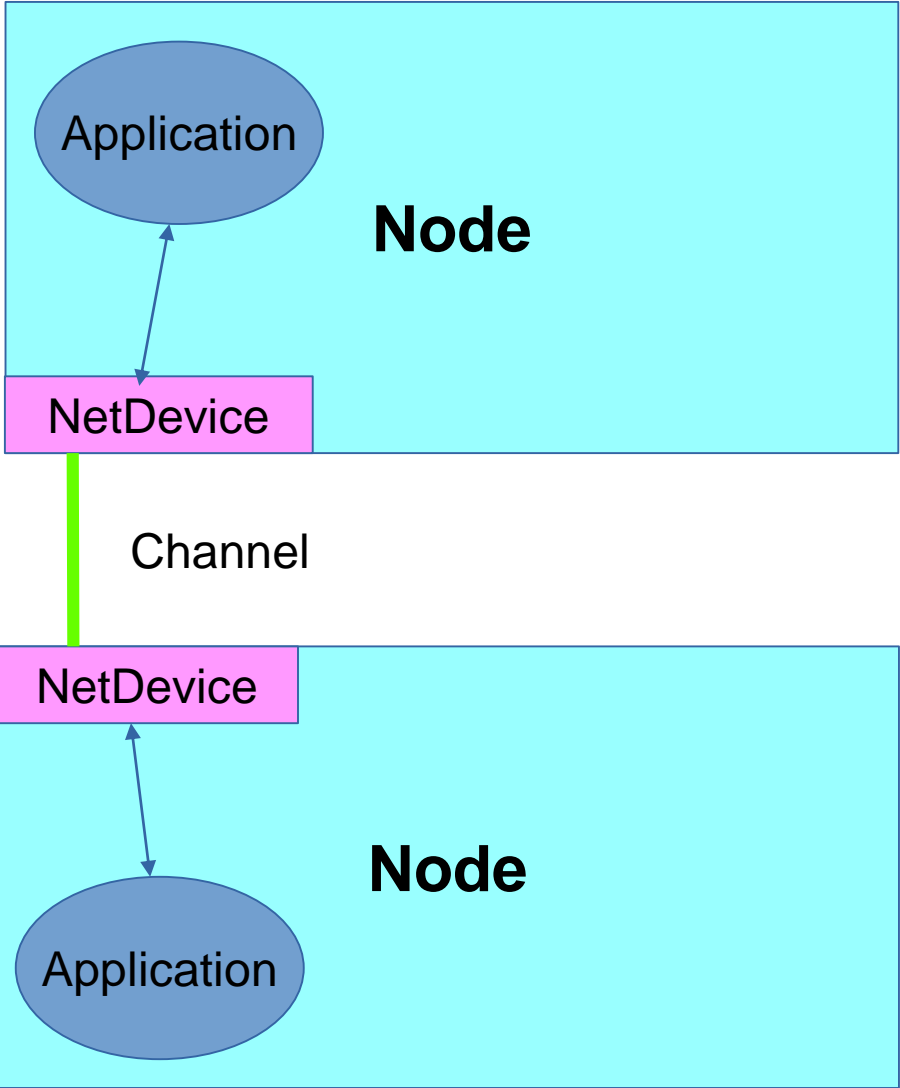
NS3 Simulation

Logical View

Components in NS3

- `Node` – denotes hosts/end systems.
- `Application` – Applications run on nodes.
 - It consumes hosts resources
- `Channel` – media over which data flows.
 - Large Ethernet Switch
 - Three-dimensional space with obstruction.
(wireless)
- `NetDevice` – A network interface attached to a node.
- Topology helpers – simplifies networks configuration process.

Components Working Together



More on Topology Helpers

- `NodeContainer` – create a set of nodes referred to be one reference.
- `PointToPointHelper` – Set channel and device attributes.
- `NetDeviceContainer` – holds network interfaces in one object.
- `InternetStackHelper` – To install an Internet stack (TCP/UDP/IP, etc) for a `NodeContainer`
- `Ipv4AddressHelper` – managing IP4 assignment

More on components & helpers!

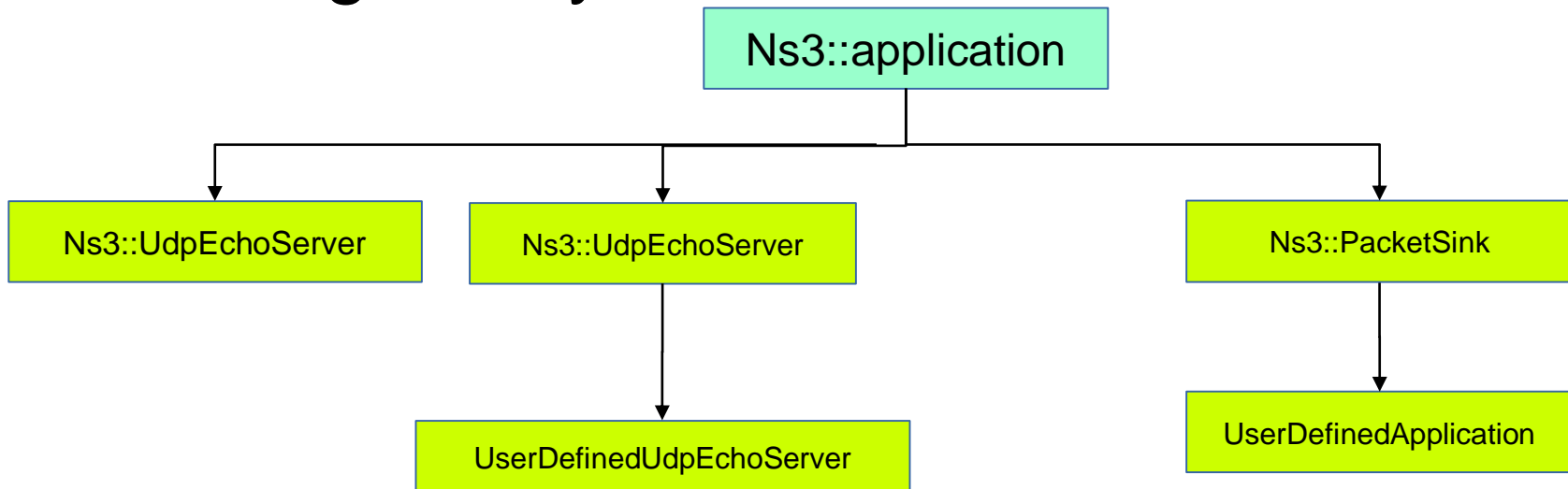
- NS3 has many components....you can search for them in the [NS3 documentation website](#)
- Under NS3 directory, go to `src`, all components are defined there.
- You can find a folder in each called `examples`, so that you have a good starting point to learn how to use a certain component.
- Topology helpers are also listed in each directory under a directory called `helpers`.

NS3 Applications

- You can “extend” a C++ class to add functionalities.
- For example, you can have `MyUserDefinedApp` extends the `ns3::Application` class.
- Some subclasses of `ns3::Application` in `ns3` that define some Apps are:
 - `UdpEchoServerApplication`.
 - `UdpEchoClientApplication`.
 - ...

Applications

- You can create your own & add functionalities to existing applications
- C++ Inheritance.
- Current Python binding does not support using library extension.

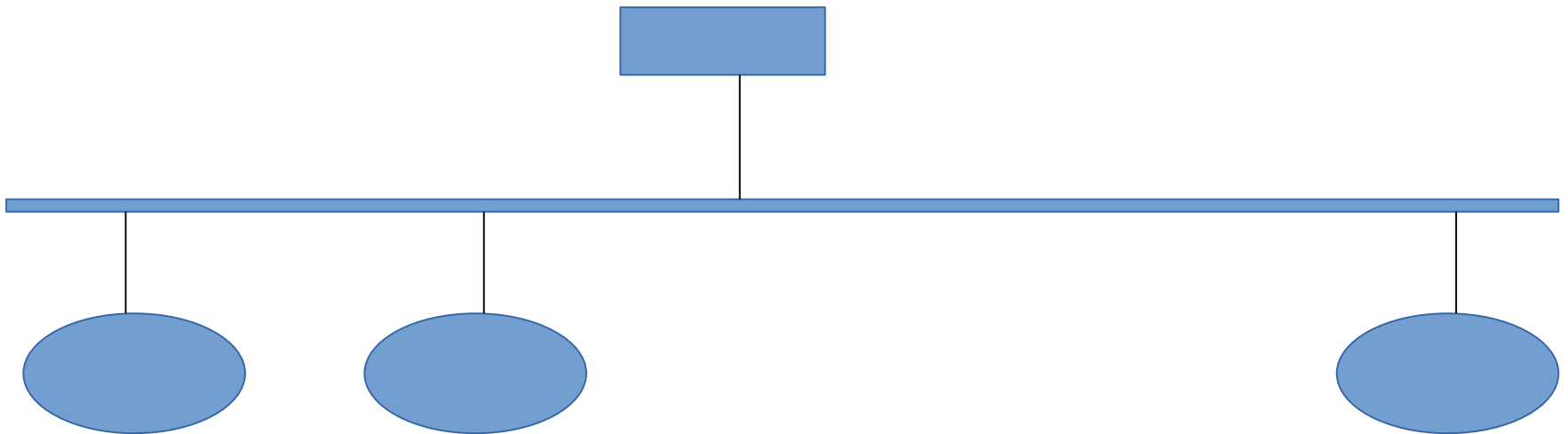


Types of Connection Channels

- Point-to-point

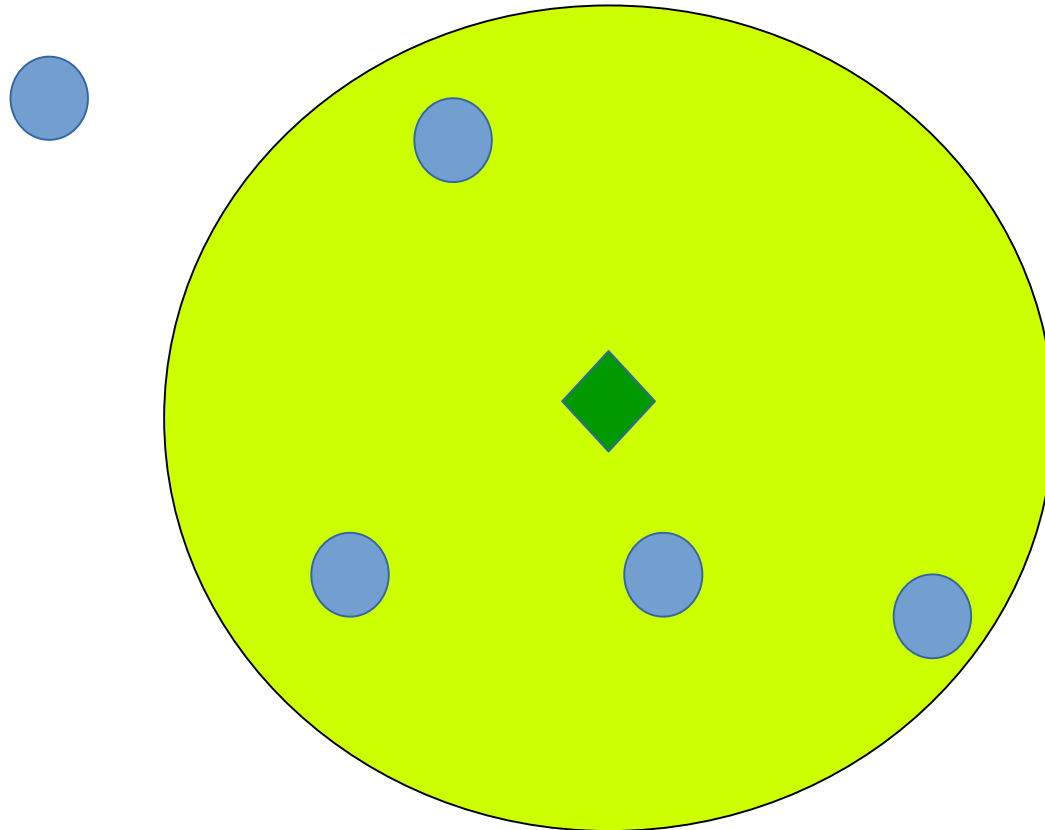


- CSMA (Carrier-Sense Multiple Access)

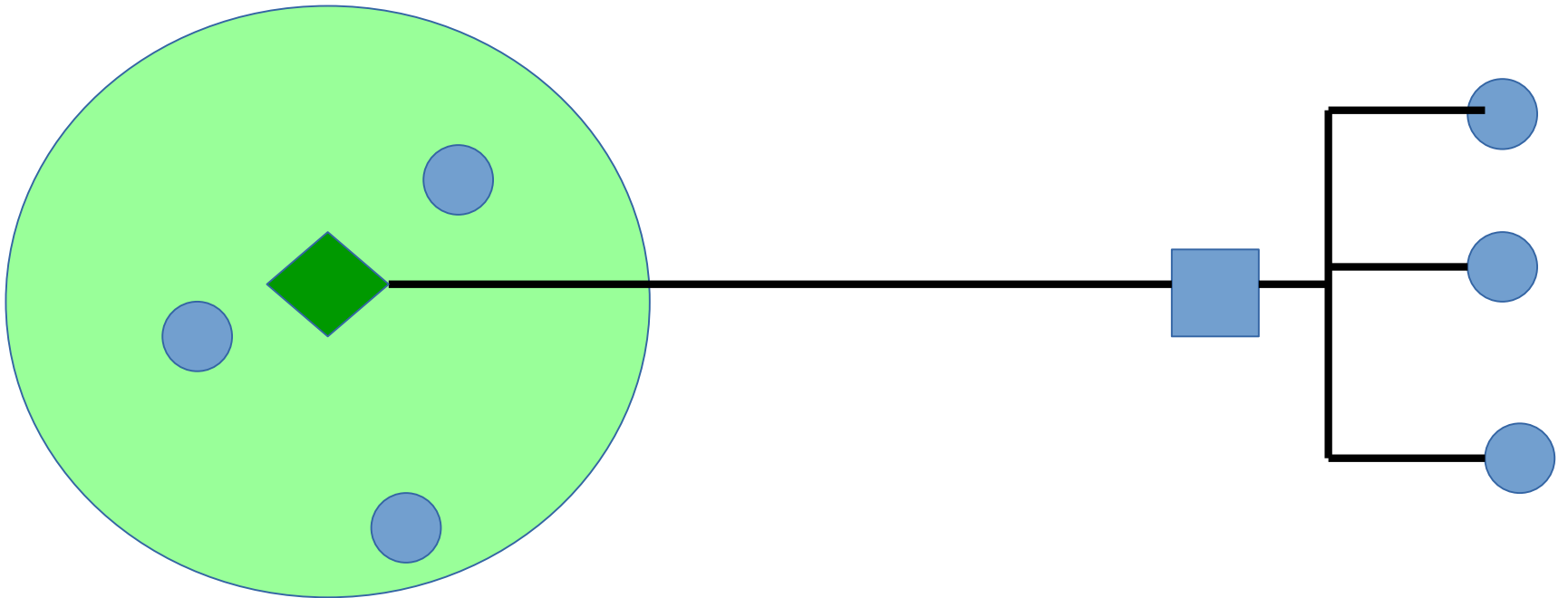


More Channels

- WIFI Channel : Wireless AP & some clients



Complex Examples

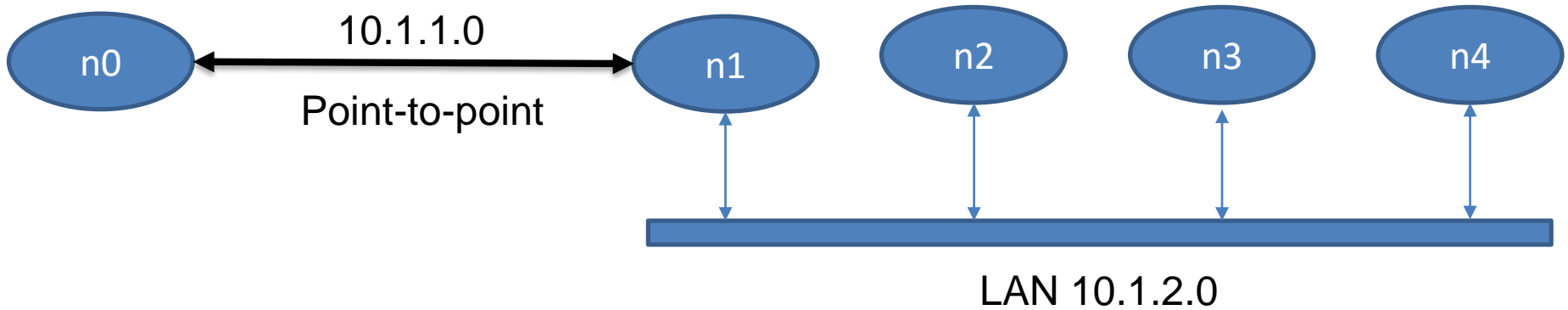


Code & Implementation

So many examples come with NS3

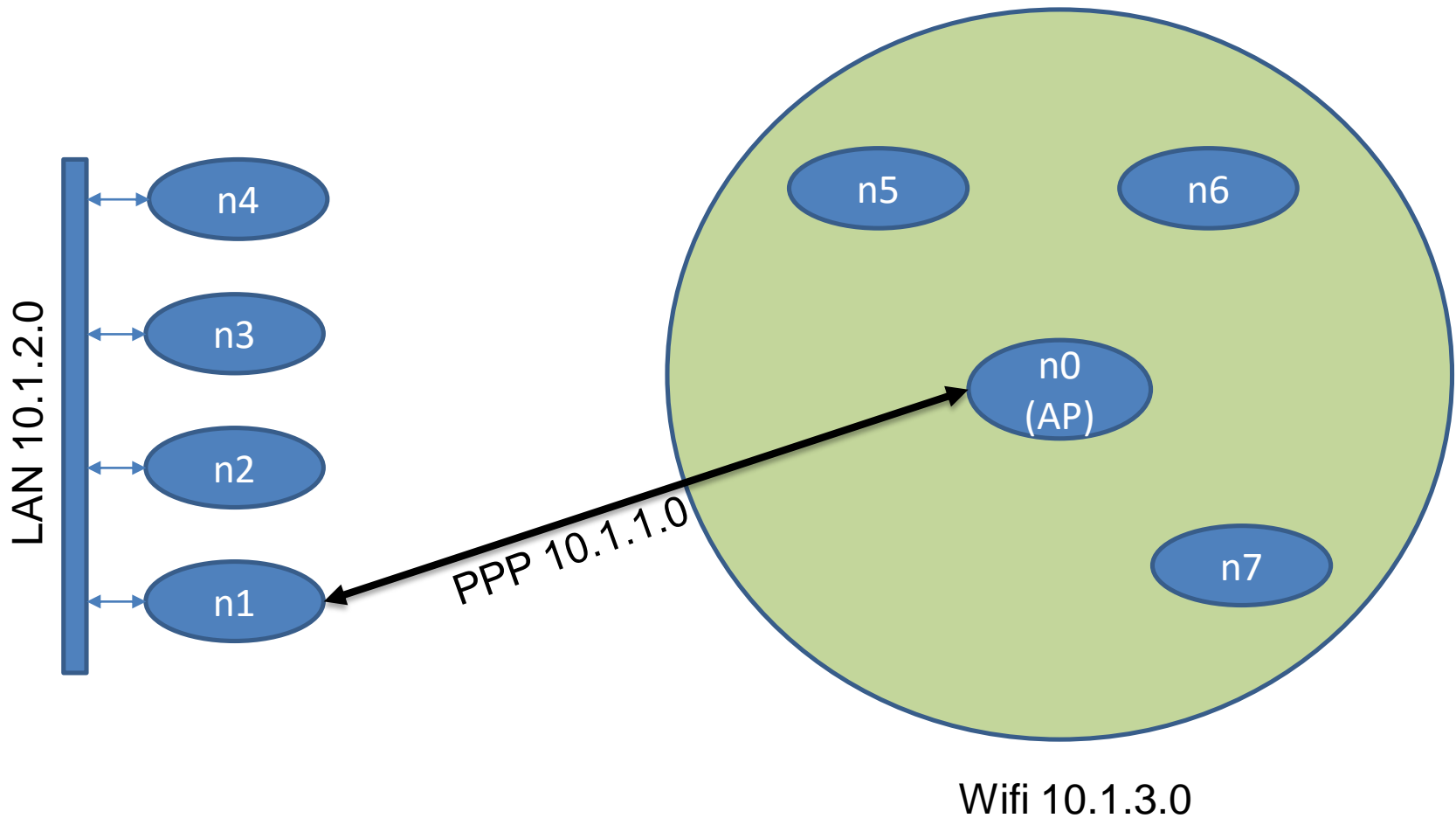
CSMA Example

`/ns-3.26/examples/tutorial/second.cc`



CSMA/Wifi Example

`/ns-3.26/examples/tutorial/third.cc`



Visualization

NetAnim Tool

- NS3 can generate XML files readable by NetAnim
 - Network topology and data transmissions.
- Create resulting visualization file as
 - `AnimationInterface anim ("VisualizationOutputFile.xml");`
- Your simulation will not appear correctly. You will need to set mobility models for your nodes.
- NetAnim visualizes simulation after the fact (off-line)

NetAnim

```
NodeContainer nodes;
```

```
...
```

```
MobilityHelper mobility;
```

```
mobility.SetPositionAllocator  
    ("ns3::GridPositionAllocator",  
        "MinX", DoubleValue (10.0),  
        "MinY", DoubleValue (10.0),  
        "DeltaX", DoubleValue (5.0),  
        "DeltaY", DoubleValue (5.0),  
        "GridWidth", UIntegerValue (1),  
        "LayoutType", StringValue  
        ("ColumnFirst")  
    );
```

```
mobility.Install(nodes);
```

PyViz (Python Visualization)

- Live Simulation Tool.
- Works with programs in scratch directory with the `--vis` option.
 - `./waf --run myprogram --vis`
- Need to enable PyViz (PyBingGen & PyGraphViz)

Logging

Motivation

- We can use `printf` and `cout` to print information or error messages.
 - However, sometimes you only want information from some components and not others
 - In addition, within the same component, you may want information about certain issues, errors, warning, etc.
- NS3 logging mechanism enables defining components, and log levels

Defining a component for LOGGING

```
...  
#include "ns3/applications-module.h"  
  
using namespace ns3;  
  
NS_LOG_COMPONENT_DEFINE ("FirstSinkExample"); // Defines a name our component.  
  
int main (int argc, char *argv[])  
{  
    Time::SetResolution (Time::NS);  
    LogComponentEnable ("UdpClient", LOG_LEVEL_INFO); //Pick a level of logging info  
    LogComponentEnable ("PacketSink", LOG_ALL); //pick error level of logging info  
    ...  
}
```

For `UdpClient`, show messages that are generated by `NS_LOG_INFO` function call and above.

For `PacketSink` is concerned show all `NS_LOG` messages.

NS3 Log Level

```
enum LogLevel {
    LOG_NONE          = 0x00000000, //!< No logging.

    LOG_ERROR         = 0x00000001, //!< Serious error messages only.
    LOG_LEVEL_ERROR   = 0x00000001, //!< LOG_ERROR and above.

    LOG_WARN          = 0x00000002, //!< Warning messages.
    LOG_LEVEL_WARN    = 0x00000003, //!< LOG_WARN and above.

    LOG_DEBUG         = 0x00000004, //!< Rare ad-hoc debug messages.
    LOG_LEVEL_DEBUG   = 0x00000007, //!< LOG_DEBUG and above.

    LOG_INFO          = 0x00000008, //!< Informational messages (e.g., banners).
    LOG_LEVEL_INFO    = 0x0000000f, //!< LOG_INFO and above.

    LOG_FUNCTION      = 0x00000010, //!< Function tracing.
    LOG_LEVEL_FUNCTION = 0x0000001f, //!< LOG_FUNCTION and above.

    LOG_LOGIC         = 0x00000020, //!< Control flow tracing within functions.
    LOG_LEVEL_LOGIC   = 0x0000003f, //!< LOG_LOGIC and above.

    LOG_ALL           = 0x0fffffff, //!< Print everything.
    LOG_LEVEL_ALL     = LOG_ALL,    //!< Print everything.

    LOG_PREFIX_FUNC   = 0x80000000, //!< Prefix all trace prints with function.
    LOG_PREFIX_TIME    = 0x40000000, //!< Prefix all trace prints with simulation time.
    LOG_PREFIX_NODE    = 0x20000000, //!< Prefix all trace prints with simulation node.
    LOG_PREFIX_LEVEL   = 0x10000000, //!< Prefix all trace prints with log level (severity).
    LOG_PREFIX_ALL     = 0xf0000000, //!< All prefixes.
};
```

Example

In NS3 file: `csma-channel.cc`

```
bool CsmaChannel::TransmitStart (Ptr<Packet> p, uint32_t srcId)
{
    NS_LOG_FUNCTION (this << p << srcId);
    NS_LOG_INFO ("UID is " << p->GetUid () << "");

    if (m_state != IDLE)
    {
        NS_LOG_WARN ("CsmaChannel::TransmitStart(): State is not IDLE");
        return false;
    }

    if (!IsActive (srcId))
    {
        NS_LOG_ERROR ("CsmaChannel::TransmitStart(): Selected source is not currently attached to network");
        return false;
    }

    NS_LOG_LOGIC ("switch to TRANSMITTING");
    m_currentPkt = p;
    m_currentSrc = srcId;
    m_state = TRANSMITTING;
    return true;
}
```

And That's it!

- **Contact**

- Adil Alsuhaïm (aalsuha@g.clemson.edu)

Feel free to direct your questions about installation of ns3 to me.

Additional References

- I posted YouTube Playlist of videos. It can be found [here](#)
- [NS3 User Manual](#)
- [NS3 Tutorial](#)
- [Slides from University of Kansas](#)
- [Yeungnam University slides](#)
- Tutorial from [UC Berkeley](#)