NS3 Demo Class 1

Installation and Running first.cc

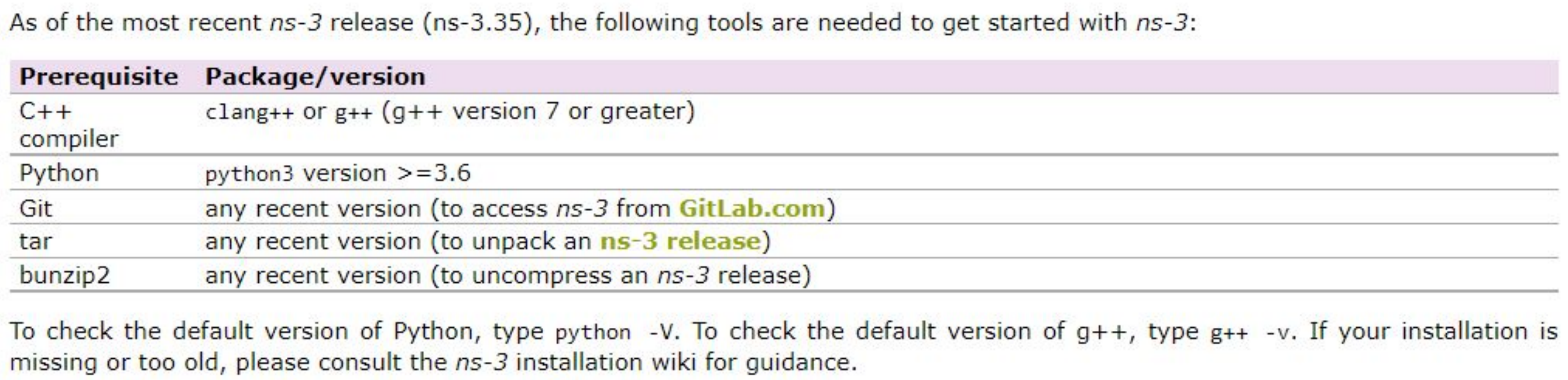
Why NS3?

- Network Simulation Tool
- **Simulate** how a topology/protocol would work in real life
- Obtain different statistics (e.g. delay, throughput, packet drop etc.)
- Advantages over NS2:
  - Can be fully scripted in C++
  - Optional python binding support
  - Documentation and troubleshooting techniques much more readily available

Note: NS2 and NS3 are not backward compatible. Here for more details.

Installation

- Use a Linux based OS with LTS (preferably Ubuntu, known to have good compatibility with NS3)
- For windows/MacOS, you can use WSL/Virtual Machine
- Run `sudo apt-get update` to update all your installed packages first
- You would also require the following tools if you don't have them already:



As of the most recent *ns-3* release (ns-3.35), the following tools are needed to get started with *ns-3*:

Prerequisite	**Package/version**
C++ compiler	`clang++` or `g++` (g++ version 7 or greater)
Python	`python3` version >=3.6
Git	any recent version (to access *ns-3* from **GitLab.com**)
tar	any recent version (to unpack an **ns-3 release**)
bunzip2	any recent version (to uncompress an *ns-3* release)

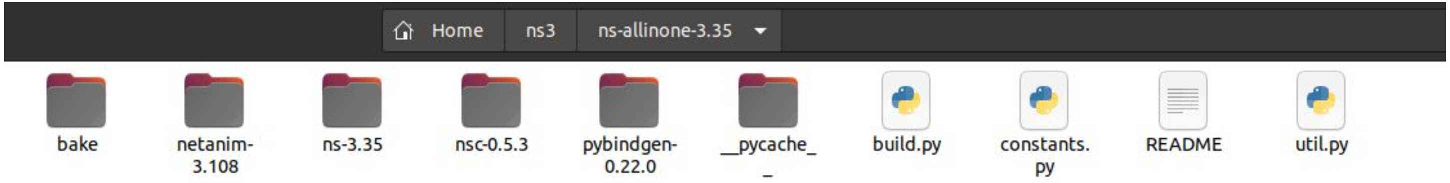
To check the default version of Python, type `python -V`. To check the default version of g++, type `g++ -v`. If your installation is missing or too old, please consult the *ns-3* installation wiki for guidance.

Installation

- Download the latest release from the following website:

  https://www.nsnam.org/releases/ns-3-35/

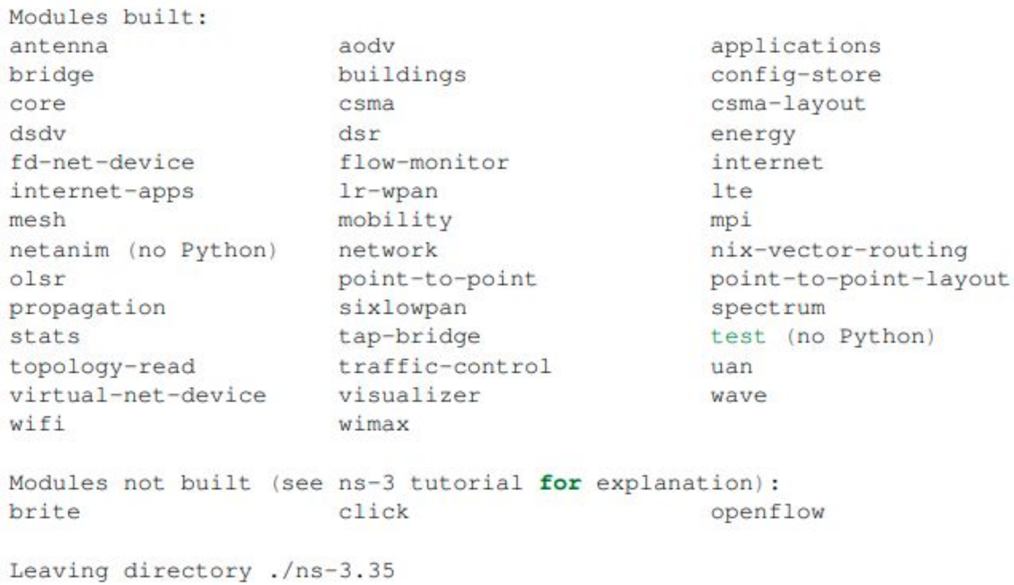
- Extract the tar.bz2 file. You will get a ns-allinone-3.35 folder, which will have these files and directories. [Warning: The directory path to your folder should not contain any spaces, i.e. no parent folder should have a name that contains any spaces.]



- Run `./build.py --enable-examples --enable-tests`

Installation

- The end would look something like this (don’t worry if some modules are not built, they need extra dependencies to be resolved first, which you can do later if you need. I had 5 modules not built during my installation.)

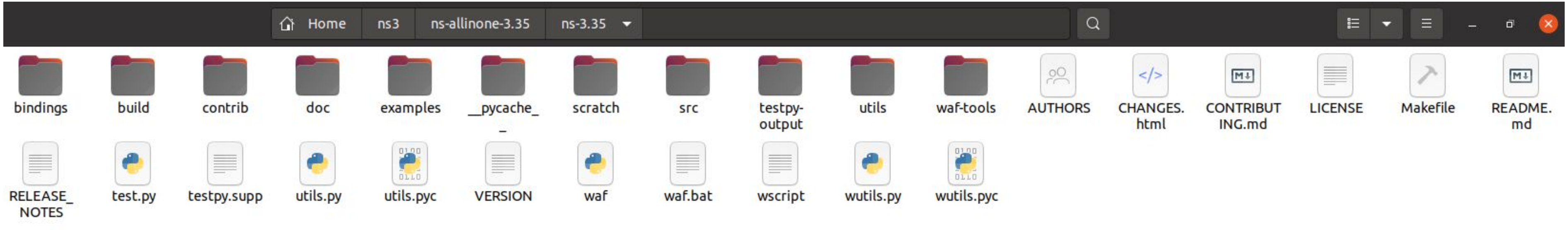


```
Modules built:
antenna                aodv                      applications
bridge                 buildings                 config-store
core                   csma                      csma-layout
dsdv                   dsr                       energy
fd-net-device          flow-monitor     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

Modules not built (see ns-3 tutorial for explanation):
brite                  click                     openflow

Leaving directory ./ns-3.35
```

Installation



- Now go to the ns-3.35 directory.
- Run `./waf clean` and then

```
./waf configure --build-profile=debug --enable-examples
--enable-tests
```

Installation

- Now to test if ns3 is installed properly, run `./waf --run hello-simulator`
- It should output “Hello Simulator”.
- Since this is the first run (or in any run after a waf configuration or clean), a compiling and linking process will happen, which will take a bit of time.

Your NS3 installation is complete!

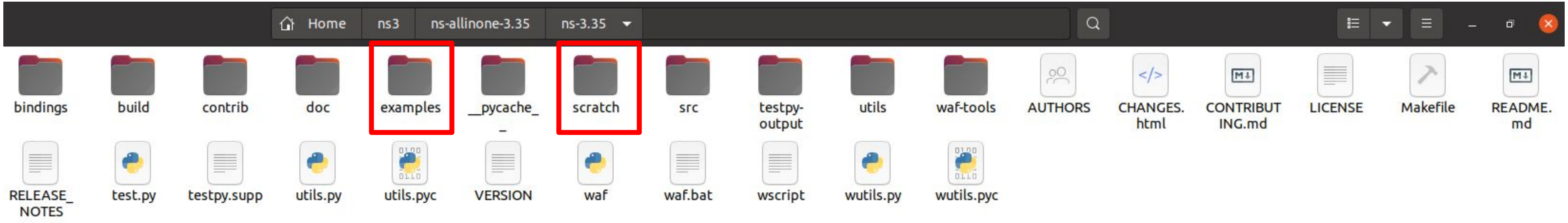
For more details about installation:

https://www.nsnam.org/docs/release/3.35/tutorial/html/getting-started.html

https://www.nsnam.org/wiki/Installation

*At the time of writing I am facing problems with python binding. Will update If I find how to resolve it. But as most of the documentation and all of the source files of ns3 are written in C++, I personally prefer using it. But it is up to you if you want to use C++ or python to write your scripts.

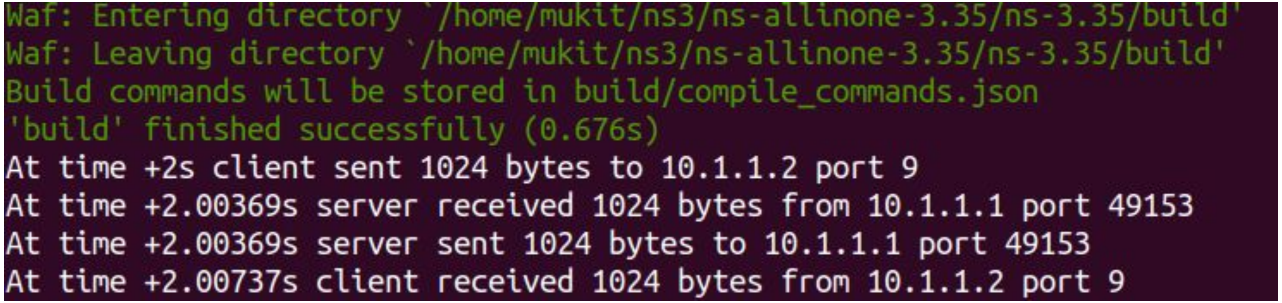
Run your first simulation file



- Copy the ‘examples/tutorial/first.cc’ file to the ‘scratch’ folder
- Run `./waf --run scratch/first`

Run your first simulation file

- Output would look something like:



```
Waf: Entering directory `/home/mukit/ns3/ns-allinone-3.35/ns-3.35/build'
Waf: Leaving directory `/home/mukit/ns3/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.676s)
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
```

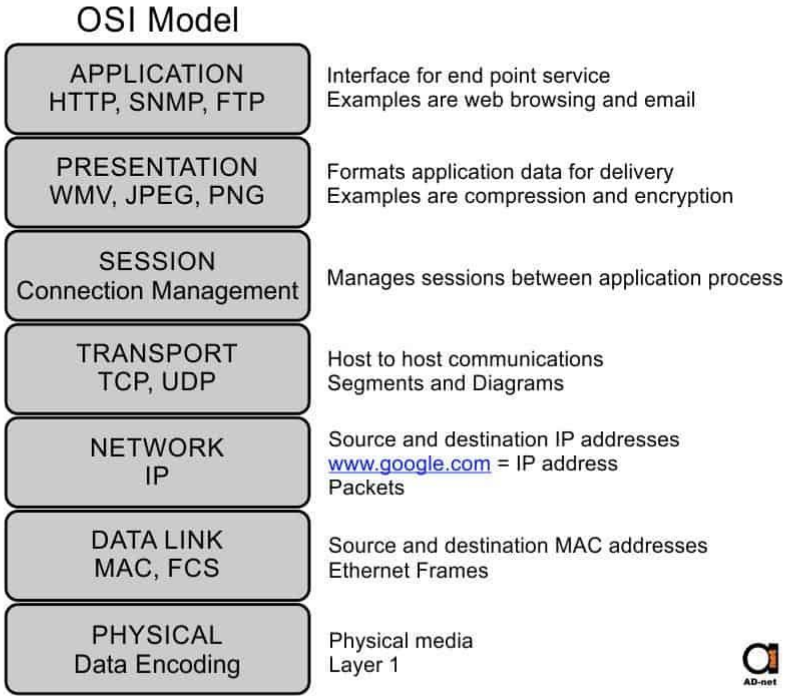
- The first.cc file simulates *two devices*, one client and the other one a server, connected via a *point-to-point channel*. The client sends an echo packet and the server sends an echo reply. The packets are sent as a UDP packets.

Integration with Visual Studio Code (Optional)

- Better if you use any IDE of your choice to write your codes in ns3. You would need to configure them as such.
- Recommended you write your code in editor, and execute your code waf in the terminal (not use your IDE for this).
- To integrate VS code, run `code .` while in the ns-3.35 directory. It would prompt the VS code editor to open in this directory.
- You would only need the C/C++ extension to run this.
- Go to help > Show all commands. Type C++ edit configurations (UI). In the include path section, add the path to your ‘build’ folder in ns-3.35.
- You should be ready to code now. Open the first.cc file in scratch folder and see if you get syntax suggestions as expected.

Understanding Simulator Code (first.cc)

- Helpful to keep the OSI Model in mind.
- In NS3, you will:
  - Define Nodes (assume this as but buying a barebone computer)
  - Add network devices to them, configure the medium channel
  - Build the topology
  - Provide IP addresses to nodes
  - Provide protocol stack
  - Setup port numbers and run application to test
  - Setup times for simulation events
  - Configure to create suitable trace/pcap files [for now assume some kind of log file as the simulation runs]. These files would be analyzed to gather the desired statistics.
  - Run the simulation.



OSI Model

APPLICATION HTTP, SNMP, FTP	Interface for end point service Examples are web browsing and email
PRESENTATION WMV, JPEG, PNG	Formats application data for delivery Examples are compression and encryption
SESSION Connection Management	Manages sessions between application process
TRANSPORT TCP, UDP	Host to host communications Segments and Diagrams
NETWORK IP	Source and destination IP addresses www.google.com = IP address Packets
DATA LINK MAC, FCS	Source and destination MAC addresses Ethernet Frames
PHYSICAL Data Encoding	Physical media Layer 1

Understanding Simulator Code (first.cc)

- The first.cc file does not produce any trace/.pcap files. It just outputs the log from the client and server applications (what you see as output).
- The documentation of all the modules and classes in NS3 can be found at:

  https://www.nsnam.org/docs/release/3.35/doxygen/index.html

  Also called the doxygen API documentation. Visit the class index and module from the tree in the left. You can get a full understanding of any class/method from the documentation. Also hovering around any function in the VSCode would give you a brief documentation of any function used.

Understanding Simulator Code (first.cc)

- To get trace files add this before the Simulation::Run() line in your scratch/first.cc file:

  ```
  AsciiTraceHelper ascii;

  pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("myfirst.tr"));
  ```

- To get .pcap files add this before the Simulation::Run() :

  ```
  pointToPoint.EnablePcapAll ("myfirst");
  ```

- You will get a ‘myfirst.tr’ and ‘myfirst-0-0.pcap’ and ‘myfirst-1-0.pcap’ file in your working directory (where the ./waf is).
- To parse the pcap file, you can use tcpdump/wireshark.
- Installing wireshark : sudo apt-get install wireshark, a configuration prompt will occur in the middle, give yes. After installing, run ‘wireshark’.
- You can open .pcap files using wireshark.

Understanding Simulator Code (first.cc)

- [Highly Recommended] Please go through chapter 5 of the tutorial book for the conceptual overview.

  https://www.nsnam.org/docs/release/3.35/tutorial/html/conceptual-overview.html

- In NS3, the classes can be derived to make your own versions. You can make your own topology, protocols etc. and test out how in works by simulating. The trace/.pcap files are analyzed to gain statistics. You will learn more as you become more familiar.
- More will be shown and discussed in the next demo class!