#### Ubuntu/Debian/Mint

The following list of packages should be accurate for Ubuntu 16.04 release; other releases or other Debian-based systems may slightly vary.

* **minimal requirements for C++ (release):** This is the minimal set of packages needed to run ns-3 from a released tarball.

apt-get install gcc g++ python

**Note:** Ubuntu 14.04 LTS release requires upgrade to gcc-4.9 from default gcc-4.8. More recent Ubuntu versions are OK. The instructions at this link: <http://askubuntu.com/questions/466651/how-do-i-use-the-latest-gcc-on-ubuntu?answertab=votes#tab-top> have been found to work.

* **minimal requirements for Python (release):** This is the minimal set of packages needed to work with Python bindings from a released tarball.

apt-get install gcc g++ python python-dev

* **minimal requirements for Python (development):** For use of ns-3-allinone repository (cloned from Mercurial), additional packages are needed to fetch and successfully install pybindgen:

apt-get install mercurial python-setuptools git

* **Netanim animator:** qt5 development tools are needed for Netanim animator; qt4 will also work but we have migrated to qt5

apt-get install qt5-default

* Support for ns-3-pyviz visualizer

apt-get install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev ipython

* Support for MPI-based distributed emulation

apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev

* Support for bake build tool:

apt-get install autoconf cvs bzr unrar

* Debugging:

apt-get install gdb valgrind

* Support for utils/check-style.py code style check program

apt-get install uncrustify

* Doxygen and related inline documentation:

apt-get install doxygen graphviz imagemagick apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils texlive-lang-portuguese dvipng

* The ns-3 manual and tutorial are written in reStructuredText for Sphinx (doc/tutorial, doc/manual, doc/models), and figures typically in dia (also needs the texlive packages above):

apt-get install python-sphinx dia

**Note:** Sphinx version >= 1.12 required for ns-3.15. To check your version, type "sphinx-build". To fetch this package alone, outside of the Ubuntu package system, try "sudo easy\_install -U Sphinx".

* GNU Scientific Library (GSL) support for more accurate WiFi error models

apt-get install gsl-bin libgsl2 libgsl-dev

* The Network Simulation Cradle (nsc) requires the flex lexical analyzer and bison parser generator:

apt-get install flex bison libfl-dev

* To read pcap packet traces

apt-get install tcpdump

* Database support for statistics framework

apt-get install sqlite sqlite3 libsqlite3-dev

* Xml-based version of the config store (requires libxml2 >= version 2.7)

apt-get install libxml2 libxml2-dev

* Support for generating modified python bindings

apt-get install cmake libc6-dev libc6-dev-i386 libclang-dev pip install cxxfilt

and you will want to install castxml and pygccxml as per the instructions for python bindings (or through the *bake* build tool as described in the tutorial).

* A GTK-based configuration system

apt-get install libgtk2.0-0 libgtk2.0-dev

* To experiment with virtual machines and ns-3

apt-get install vtun lxc

* Support for openflow module (requires some boost libraries)

apt-get install libboost-signals-dev libboost-filesystem-dev

### Manual installation

The ns-3 code is available in Mercurial repositories on the server [http://code.nsnam.org](http://code.nsnam.org/) (look for the latest release e.g., "ns-3.26"). You can download a tarball of the latest release at <http://www.nsnam.org/releases> or you can work with our repositories using Mercurial. We recommend using Mercurial unless there's a good reason not to (See the end of this section for instructions on how to get a tarball release).

The simplest way to get started using Mercurial repositories is to use the **ns-3-allinone** environment. This is a set of scripts that manages the downloading and building of various subystems of ns-3 for you. We recommend that you begin your ns-3 adventures in this environment as it can really simplify your life at this point.

#### Downloading ns-3 Using Mercurial

One practice is to create a directory called **repos** in one's home directory under which one can keep local Mercurial repositories. If you adopt that approach, you can get a copy of ns-3-allinone by typing the following into your Linux shell (assuming you have installed Mercurial):

cd mkdir repos cd repos hg clone <http://code.nsnam.org/ns-3-allinone>

As the hg (Mercurial) command executes, you should see something like the following displayed,

destination directory: ns-3-allinone requesting all changes adding changesets adding manifests adding file changes added 26 changesets with 40 changes to 7 files 7 files updated, 0 files merged, 0 files removed, 0 files unresolved

After the clone command completes, you should have a directory called ns-3-allinone under your ~/repos directory, the contents of which should look something like the following:

build.py\* constants.py dist.py\* download.py\* README util.py

Notice that you really just downloaded some Python scripts. The next step will be to use those scripts to download and build the ns-3 distribution of your choice.

If you go to the following link: <http://code.nsnam.org/> you will see a number of repositories. Many are the private repositories of the ns-3 development team. The repositories of interest to you will be prefixed with **ns-3**. Official releases of ns-3 will be numbered as ns-3.release.hotfix. For example, a second hotfix to a still hypothetical release nine of ns-3 would be numbered as ns-3.9.2 on this page.

The current development snapshot (unreleased) of ns-3 may be found at <http://code.nsnam.org/ns-3-dev/>. The developers attempt to keep these repository in consistent, working states but they are in a development area with unreleased code present, so you may want to consider staying with an official release if you do not need newly-introduced features.

Since the release numbers are going to be changing, we will stick with the more constant ns-3-dev here, but you can replace the string *ns-3-dev* with your choice of release (e.g., ns-3.26) in the text below. You can find the latest version of the code either by inspection of the repository list or by going to the *Getting Started* web page and looking for the latest release identifier.

To download the most common options type the following into your shell (remember you can substitute the name of your chosen release number instead of ns-3-dev)

./download.py -n ns-3-dev

After download process completes, you should have several new directories under ~/repos/ns-3-allinone:

build.py\* constants.pyc download.py\* nsc/ README util.pyc constants.py dist.py\* ns-3-dev/ pybindgen/ util.py

Go ahead and change into ns-3-dev under your ~/repos/ns-3-allinone directory. You should see something like the following there:

AUTHORS examples/ RELEASE\_NOTES utils/ wscript bindings/ LICENSE samples/ VERSION wutils.py CHANGES.html ns3/ scratch/ waf\* doc/ README src/ waf.bat\*

You are now ready to build the ns-3 distribution.

#### Downloading ns-3 Using a Tarball

The process for downloading ns-3 via tarball is simpler than the Mercurial process since all of the pieces are pre-packaged for you. You just have to pick a release, download it and decompress it.

As mentioned above, one practice is to create a directory called **repos** in one's home directory under which one can keep local Mercurial repositories. One could also keep a tarballs directory. If you adopt the tarballs directory approach, you can get a copy of a release by typing the following into your Linux shell (substitute the appropriate version numbers, of course):

cd mkdir tarballs cd tarballs wget <http://www.nsnam.org/release/ns-allinone-3.13.tar.bz2> tar xjf ns-allinone-3.13.tar.bz2

If you change into the directory **ns-allinone-3.13** you should see a number of files:

build.py\* ns-3.13/ pybindgen-0.15.0.795/ util.py constants.py nsc-0.5.2/ README

You are now ready to build the ns-3 distribution.

## Building ns-3 with build.py

The first time you build the ns-3 project you should build using the allinone environment. This will get the project configured for you in the most commonly useful way.

Change into the directory you created in the download section above. If you downloaded using Mercurial you should have a directory called ns-3-allinone under your ~/repos directory. If you downloaded using a tarball you should have a directory called something like ns-allinone-3.13 under your ~/tarballs directory. Type the following:

./build.py

You will see lots of typical compiler output messages displayed as the build script builds the various pieces you downloaded. Eventually you should see the following magic words:

Build finished successfully (00:02:37) Leaving directory `./ns-3-dev'

Once the project has built you typically will not use ns-3-allinone scripts. You will now interact directly with Waf and we **do it in the ns-3-dev directory and not in the ns-3-allinone directory**.

### Configuration with Waf

To see valid configure options, type ./waf --help. The most important option is -d <debug level>. Valid debug levels (which are listed in waf --help) are: "debug" or "optimized". It is also possible to change the flags used for compilation with (e.g.):

CXXFLAGS="-O3" ./waf configure

or, alternately, the gcc compiler

CXX=g++-3.4 ./waf configure

**Note:** Unlike some other build tools, to change the build target, the option must be supplied during the configure stage rather than the build stage (i.e., "./waf -d optimized" will not work; instead, do

./waf -d optimized configure; ./waf

The resulting binaries are placed in build/<debuglevel>/srcpath. For example, in a debug build you can find the executable for the first.cc example as build/examples/first. You can debug the executable directly by:

./waf --shell cd build/debug/examples gdb ns-<version>-first-debug

Of course, you can run gdb in emacs, or use your favorite debugger such as ddd or insight just as easily. In an optimized build you can find the executable for the first.cc example as build/examples/ns-<version>-first-optimized.

In order to forcibly disable python bindings, you can provide the following option:

./waf --disable-python configure

In order to tell the build system to use the sudo program to set the suid bit if required, you can provide the following option:

./waf --enable-sudo configure

To start over a configuration from scratch, type:

./waf distclean

Or if you get stuck and all else fails:

rm -rf build

followed by changing back into ns-3-allinone and doing:

./build.py

will basically reset your build state.

To see all waf options:

./waf --help

## Validating

ns-3 has unit tests that can be run to verify the installation:

./test.py

which should produce output like:

PASS: TestSuite histogramPASS: TestSuite ns3-wifi-interferencePASS: TestSuite ns3-tcp-cwndPASS: TestSuite ns3-tcp-interoperabilityPASS: TestSuite sample...

# Enabling/disabling *ns-3* Tests and Examples

The *ns-3* distribution includes many examples and tests that are used to validate the *ns-3* system. Users, however, may not always want these examples and tests to be run for their installation of *ns-3*.

This chapter discusses how to build *ns-3* with or without its examples and tests.

## How to enable/disable examples and tests in *ns-3*

There are 3 ways to enable/disable examples and tests in *ns-3*:

1. Using build.py when *ns-3* is built for the first time
2. Using waf once *ns-3* has been built
3. Using the *ns-3* configuration file once *ns-3* has been built

### Enable/disable examples and tests using build.py

You can use build.py to enable/disable examples and tests when *ns-3* is built for the first time.

By default, examples and tests are not built in *ns-3*.

From the ns-3-allinone directory, you can build *ns-3* without any examples or tests simply by doing:

./build.py

Running test.py in the top level *ns-3* directory now will cause no examples or tests to be run:

0 of 0 tests passed (0 passed, 0 skipped, 0 failed, 0 crashed, 0 valgrind errors)

If you would like build *ns-3* with examples and tests, then do the following from the ns-3-allinone directory:

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

Running test.py in the top level *ns-3* directory will cause all of the examples and tests to be run:

170 of 170 tests passed (170 passed, 0 skipped, 0 failed, 0 crashed, 0 valgrind errors)

Instalação do modulo ofswitch13 (openflow 1.3)

OpenFlow 1.3 module for ns3 was developed to ns3 version 3.25. The virtual machine image (available [here](http://www.lrc.ic.unicamp.br/ofswitch13/ns3-ofswitch13.zip)) is for Ubuntu 14.04.1, but I'll install it on Ubuntu 16.04. Follow these steps:

*sudo apt-get install libpcre2-dev libxerces-c-dev libpcap0.8-dev*

*wget* [*https://bitbucket.org/ljerezchaves/ofswitch13-module/downloads/nbeesrc.zip*](https://bitbucket.org/ljerezchaves/ofswitch13-module/downloads/nbeesrc.zip)

*unzip nbeesrc.zip*

*cd netbee/src/*

*cmake .*

*make*

*sudo cp ../bin/libn\*.so /usr/local/lib*

*sudo ldconfig*

*sudo cp -R ../include/\* /usr/include/*

*git clone* [*https://github.com/ljerezchaves/ofsoftswitch13*](https://github.com/ljerezchaves/ofsoftswitch13)

*cd ofsoftswitch13*

*git checkout v2.0.x*

*./boot.sh*

*./configure --enable-ns3-lib*

*make*

*hg clone* [*http://code.nsnam.org/ns-3.24*](http://code.nsnam.org/ns-3.24)

*cd ns-3.24*

*hg update ns-3.24.1*

*hg clone* [*https://bitbucket.org/ljerezchaves/ofswitch13-module*](https://bitbucket.org/ljerezchaves/ofswitch13-module) *src/ofswitch13*

*cd src/ofswitch13*

*hg update 2.0.2*

*cd ../../*

*patch -p1 < src/ofswitch13/utils/ofswitch13-csma-3\_24\_1.patch*

*patch -p1 < src/ofswitch13/utils/ofswitch13-doc-3\_24\_1.patch*

*./waf configure --enable-examples --enable-tests --enable-mpi --with-ofswitch13=PATH\_TO\_ofsoftswitch13 --with-brite=PATH\_TO\_BRITE*

*./waf*