Get Started

We support three ways to install NetworKit:

- NetworKit Virtual Machine: Download and try NetworKit preinstalled on a virtual machine. This is strongly recommended for users using Microsoft Windows.
- Pip install: Download the NetworKit Python package with pip. This is the easier way to get NetworKit but you can only use NetworKit via Python this way.
- Build NetworKit from Source: Clone or download the source code of NetworKit and build the C++ and Python modules from source.

With NetworKit as a Python extension module, you get access to native high-performance code and can at the same time work interactively in the Python ecosystem. Although the standard Python interpreter works fine, we recommend IPython as a great environment for scientific workflows. View the IPython Quickstart Guide for installation instructions and how to use NetworKit with IPython.

Once you have installed NetworKit, please make sure to check out our NetworKit UserGuide for an overview of the features provided in NetworKit.

Install the NetworKit Virtual Machine

If you want a quick and easy way to try NetworKit for your purposes or you use a Microsoft Windows operating system, we strongly recommend the installation of our NetworKit virtual machine.

A detailed installation guide can be found here.

Install NetworKit via Pip

Requirements

You will need the following software to install NetworKit as a python package:

- A modern C++ compiler, e.g.: g++ (>= 4.8) or clang++ (>= 3.7)
- Python 3 (>= 3.4 is recommended, 3.3 supported)
- Pip
- SCons: Please note that SCons is only available for Python 2. For installation via pip, we have a script that builds the C++ part of NetworKit, so you can try it without SCons.
- Cython (>= 0.21): Only needed by developers.

NetworKit uses some additional external Python packages. While you do not need them to run NetworKit, it is strongly recommended to install them in order to use all the features of NetworKit:

- scipy
- numpy
- readline
- matplotlib
- networkx
- tabulate

You can use the command `pip3 install scipy numpy readline matplotlib networkx tabulate` on your terminal to install all packages at once. During the installation of NetworKit, the setup will check if the external packages NetworKit uses are available and print warnings at the end of the installation process. If you do not see any warnings, your system should be ready to use NetworKit.

Install NetworKit

Run `[sudo] pip[3] install [--user] networkit` from your command line to install the Python package *networkit*.

You can remove NetworKit completely by using the command `[sudo] pip[3] uninstall networkit`.

To check that everything works as expected, open a python terminal and run the following lines:

```
import networkit
G = networkit.Graph(5)
G.addEdge(0,1)
G.toString()
```

Build NetworKit from Source

You can clone NetworKit from AlgoHub with Mercurial or download the source code as a Zip file.

Requirements

You will need the following software to install NetworKit as a Python package:

- A modern C++ compiler, e.g.: g++ (>= 4.8) or clang++ (>= 3.7)
- SCons: Please note that SCons is only available for Python 2. For the different build targets, SCons is mandatory.
- Google Test (only needed if you want to build the unit tests, which is recommended)

Building NetworKit

This section describes how to build NetworKit including the Python functionality. If you do not wish to install NetworKit as a Python package, please refer to Building Only the C++ Core.

For building NetworKit including the Python functionality, make sure to also install the software from the Python Requirements listed in the Pip install.

After all requirements are installed, switch to the top folder of NetworKit and run the script *setup.py* with the following options:

```
python3 setup.py build_ext --inplace [--optimize=V] [-jX]
```

The script will call SCons to compile NetworKit as a library and then build the extensions in the folder *src/python*. By default, NetworKit will be built with the amount of available cores in optimized mode. It is possible to add the options `--optimize=V` and `-jN` the same way it can be done to a manual SCons call, to specify the optimization level and the number of threads used for compilation. The setup script provides more functionality and can be used with pip aswell:

```
pip3 install -e ./
```

will compile NetworKit, build the extensions and on top of that temporarily install NetworKit so that it is available on the whole system. This can be undone by calling `pip3 uninstall networkit`.

```
python3 setup.py clean [--optimize=V]
```

will remove the extensions and its build folder as well as call SCons to remove the NetworKit library and its build folder specified by `--optimize=V`.

Note: All of the above installation command may require root privileges depending on your system, so try this accordingly. If you do not have root privileges, add `--user` to your command.

Building Only the C++ Core

In case you do not need NetworKit's Python functionality, this section describes how to build the C++ parts only.

We recommend SCons for building the C++ part of NetworKit. Individual settings for your environment will be read from a configuration file. As an example, the file *build.conf.example* is provided. Copy this to *build.conf* and edit your environment settings. Then call Scons.

The call to SCons has the following options:

```
scons --optimize=<level> --target=<target>
```

where `<level>` can be

- `Dbg` debug
- `Opt` optimized
- `Pro` profiling

and `target` can be

- `Core` build NetworKit as a library, required for the Python extenstion through Cython.
- `Tests` build executable for the unit tests (requires GoogleTest).
- `Lib` build NetworKit as a library and create symbolic links.

For example, to build NetworKit as an optimized library, run

```
scons --optimize=Opt --target=Lib
```

To speed up the compilation on a multicore machine, you can append `-jX` where X denotes the number of threads to compile with.

Logging is enabled by default. If you want to disable logging functionality, add the following to your scons call:

```
--logging=no
```

Use NetworKit as a library

It is also possible to use NetworKit as a library. Therefore, choose the target Lib when compiling NetworKit. The include directives in your C++-application look like the following

```
#include <NetworKit/graph/Graph.h>
```

NetworKit in the directory include is a symlink to the directory networkit/cpp, so the directory structure from the repository is valid. To compile your application, you need to add the paths for the header files and the location of the library. Note, that it is possible to link the different builds (debug, profiling, optimized) of the library. There is a simple source file to demonstrate this. Feel free to compile LibDemo.cpp as follows:

```
g++ -o LibDemo -std=c++11 -I/path/to/repo/include -L/path/to/repo LibDemo.cpp -lNetworKit -fopenmp
```

Test

You actually do not need to build and run our unit tests. However, if you experience any issues with NetworKit, you might want to check, if NetworKit runs properly. Please refer to the Unit Tests and Testing section in our NetworKit Development Guide.

Known Issues

- Mac OS X 10.10 "Yosemite": Some users have reported compilation problems on Yosemite with g++ 4.9. The compiler errors mention register problems. While the exact reason remains unclear, the actual issue seems to be that the compiler tries to perform a dual architecture build. Fix: Enforce a 64-bit build by prepending `ARCHFLAGS="-arch x86_64"` to your setup/pip command, e.g. as in `sudo ARCHFLAGS="-arch x86_64" python3 setup.py build_ext --inplace -j4` or `sudo ARCHFLAGS="-arch x86_64" pip3 install networkit`.
- NetworKit has not yet been successfully built on **Windows**. This is partially due to the fact that Windows ships without a C++ compiler which is necessary to build the Python extensions. Even with the Visual C++ Redistributable our attempts were not successful. Any help is appreciated. It may be possible to build NetworKit as a library on Windows in environments like MinGW or Cygwin.

Contributions

We would like to encourage contributions to the NetworKit source code. See the NetworKit Development Guide for instructions. For support please contact the mailing list.

Use NetworKit with IPython

First make sure you have installed IPython, e.g. via pip: `pip3 install ipython`.

IPython Terminal

If you want to use NetworKit in the IPython terminal, type the following commands in your OS terminal:

```
ipython3
```

```
from networkit import *
```

The first line opens the IPython terminal. The second line imports the *networkit* Python module. After that, you should be able to use NetworKit interactively. For usage examples, refer to the NetworKit UserGuide.

IPython Notebook/jupyter

Additionally, we recommend that you familiarize yourself with NetworKit through experimenting with the interactive IPython Notebook NetworKit_UserGuide.ipynb located in the folder Doc/Notebooks. The user guide also introduces a large portion of NetworKits functionality with usage examples. To display and work with these notebooks, you have to install jupyter and start a local notebook server from the terminal with:

```
jupyter/ipython3 notebook
```

If you run into any problems with jupyter, head over to the jupyter documentation. If the notebook server starts as it is supposed to, your default browser should open a web interface or you have to open it manually. Then you can add NetworKit_UserGuide.ipynb from the above mentioned location or browse to the location through the web interface.

To show plots within the notebooks, place the following two lines at the beginning of your notebook:

```
%matplotlib inline
matplotlib.pyplot as plt
```

Note: Instead of running jupyter, it may still be possible to run `ipython3 notebook`. However, the notebook functionality of the ipython package is deprecated and has been moved to jupyter, which we strongly recommend.

Usage Example

Now that you are done installing NetworKit, you might want to try the following example:

```
>>> from networkit import *
>>> g = generators.HyperbolicGenerator(1e5).generate()
>>> overview(g)
Network Properties for:     G#5
nodes, edges            100000, 300036
directed?           False
weighted?           False
isolated nodes          1815
self-loops          0
density             0.000060
clustering coefficient      0.720003
min/max/avg degree      0, 1174, 6.000720
degree assortativity        0.001383
number of connected components  4026
size of largest component   78387 (78.39 %)

>>> communities = community.detectCommunities(g, inspect=True)
PLM(balanced,pc,turbo) detected communities in 0.14902853965759277 [s]
solution properties:
-------------------  -----------
```

```
# communities          4253
min community size        1
max community size     1821
avg. community size      23.5128
modularity                0.987991
-------------------  -----------

>>>
```