README - Installation guideline for ndnSIM 2.3 and usage of the voicemail scenario

Sandra-Nadine Bergmann

10. April 2022

1 Install ndnSIM

- An installation guideline can be found here (`http://ndnsim.net/2.2/getting-started.html`).
- Platform: Ubuntu 16.04 (64 bit)
- Requirements and dependencies:

```
python >= 2.6
libsqlite3
libcrypto++
pkg-config
Boost libraries >= 1.53
```

- Install core-dependencies (required tools and libraries):

```
sudo apt-get install build-essential libsqlite3-dev
  libcrypto++-dev libboost-all-dev
```

- Install requirements for NS-3 Python Bindings:

```
sudo apt-get install python-dev python-pygraphviz
  python-kiwi
sudo apt-get install python-pygoocanvas python-gnome2
sudo apt-get install python-rsvg ipython
```

- Install git:

```
sudo apt-get install git
```

- Download ndnSIM source code:
- NdnSIM contains 3 parts (NS-3 custom branch, customized Python Bindings generation library for the visualizer module, the ndnSIM source code and a modified source code of ndn-cxx library and NDN Forwarding Daemon (NFD)) as git-submodules.
- The following commands download all parts from GitHub repositories:

```
mkdir Dokumente/uni/ndnSIM
cd Dokumente/uni/ndnSIM
git clone https://github.com/named-data-ndnSIM/
  ns-3-dev.git ns-3
git clone https://github.com/named-data-ndnSIM/
  pybindgen.git pybindgen
git clone --recursive https://github.com/
  named-data-ndnSIM/ndnSIM.git ns-3/src/ndnSIM
```

- When --recursive is not used, then the following command has to be used in the git folder:

```
cd Dokumente/uni/ndnSim/ns-3
git submodule update --init
```

- Build and run ndnSIM with the following command (build ndnSIM with Python Bindings enabled):

```
cd Dokumente/uni/ndnSim/ns-3
./waf configure --enable-examples
./waf
```

- Sometimes Python Bindings, XmlIo and PyViz visualizer are not enabled after the previous command. This can be seen in the cmd output.
- The following commands can be used to enable Python Bindings (which is not described in the online guideline):

```
sudo apt-get install gccxml
sudo apt-get install python-pygccxml
sudo apt-get install libxml2-dev
```

- With these commands, Python Bindings, XmlIo und PyViz visualizer are enabled.
- But that is not always enough. A file version.py has to be created as well with a specific content when this file does not exist. This file should be located in the folder Dokumente/uni/ndnSim/pybindgen/pybindgen with the filename version.py. If this file does not exist, then perform the following steps:

- Create this file:

```
cd Dokumente/uni/ndnSim/pybindgen/pybindgen
touch version.py
```

- Copy the content into the version.py file when creating the file. This step should not be performed when this file already exists.

- The content of the version.py file:

```
cd Dokumente/uni/ndnSim/pybindgen/pybindgen
cat version.py
# coding: utf-8
# file generated by setuptools_scm
# don't change, don't track in version control
version = '0.17.0.post45+ng4806e4f'
```

- If the following works, then files can be compiled the following way:

```
cd Dokumente/uni/ndnSim/pybindgen
./waf configure
./waf
```

- Perform the following step:

```
cd Dokumente/uni/ndnSim/ns-3
./waf configure --enable-examples
./waf
```

- Simulation with ndnSIM:

- Simulation scenarios can be created and executed in a folder (scenario). Therefore, the following commands can be performed:

```
cd Dokumente/uni/ndnSIM/ns-3
sudo ./waf install
cd ..

git clone https://github.com/named-data-ndnSIM/
  scenario-template.git scenario
cd scenario
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
export LD_LIBRARY_PATH=/usr/local/
  lib:$LD_LIBRARY_PATH

./waf configure --debug

./waf --run <scenario>
```

- It is important that ./waf configure is executed with the parameter --debug. This is needed when ./waf configure --enable examples is executed in ns-3 folder before (and not ./waf configure -d optimized). With ./waf configure -d optimized no log outputs are created.
- Compiling with ./waf configure --debug --logging is also possible.
- The examples from the tutorial can be executed the following way:

```
cd Dokumente/uni/ndnSIM/scenario
./waf
./waf --run <scenario>
```

- Therefore, the scenario files (.cpp files) and topology files (.txt) have to be copied to the folders Dokumente/uni/ndnSIM/scenario/scenarios and Dokumente/uni/ndnSIM/scenario/scenarios/topologies. Predefined example scenarios can be found in the folder Dokumente/uni/ndnSIM/ns-3/src /ndnSIM/examples. These files can be copied in the scenario folder.
- The command ./waf can also be ommited.
- Example (normal execution and with logs):

```
cd Dokumente/uni/ndnSIM/scenario
./waf --run=ndn-simple
NS_LOG=ndn.Producer:ndn.Consumer ./waf --run=
  <scenario>
NS_LOG=nfd.Forwarder ./waf --run=ndn-simple --vis
```

- Simulations can also be executed with the vizualizer:

```
cd Dokumente/uni/ndnSIM/scenario
./waf --run=ndn-simple --vis
```

- The complete installation guideline can be found here (`http://ndnsim.net/2.2/getting-started.html`). This installation guideline is slightly different.
- Notice: The current link to the installation guideline can be found on `ndnsim.net/`.
- Notice: The following command and change has been performed as well:

```
vim ~/.bashrc
```

- Therefore, the following line is added at the bottom:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

- This is not needed when the scenario is compiled with ./waf configure --debug. Then everything works fine without this command.
- The file bashrc is located in the directory /home/sandra (hidden file).
- Additional installations for ndnSIM 2-3:

```
sudo apt-get install libssl-dev
sudo apt-get install python-setuptools
```

2 Install BRITE

- For generating random networks BRITE is needed.
- For installing BRITE you have to install mecurial before:

```
sudo apt-get install mecurial
```

- After this the BRITE repository can be cloned:

```
cd Dokumente/uni/ndnSIM_2-3
hg clone http://code.nsnam.org /BRITE
cd BRITE
make
sudo cp libbrite.so /usr/lib/
```

- ns-3 has to be configured again:

```
cd ns-3
./waf configure --enable-examples
  --with-brite=../BRITE
./waf
sudo ./waf install
```

- Copy some files:

```
cd ../..
cd BRITE
sudo cp *.h /usr/local/include/ns3-dev/ns3
sudo mkdir /usr/local/include/ns3-dev/ns3/Models
cd Models
sudo cp *.h /usr/local/include/ns3-dev/ns3/Models
```

- Move into the scenario folder, configure and run the scenario:

```
cd ../..
cd scenario
./waf configure --debug --logging
./waf
./waf --run=<scenario>
```

3 Installation for Plotting

- Therefore matplotlib is used.
- Check if packages for mathplotlib are already installed:

```
apt-cache policy python-numpy
apt-cache policy python3-matplotlib
apt-cache policy python-scipy
```

- If not installed, install these packages:

```
sudo apt-get install python-numpy
sudo apt-get install python3-matplotlib
sudo apt-get install python-scipy
```

- Scripts which use matplotlib have to be executed like:

```
python3 scriptname.py
```

- A better way to install scipy:

```
sudo apt-get install python3-pip
sudo apt-get purge python-scipy
pip3 install scipy
```

4 Create a new scenario

- Move into the ndnSIM_2-3 folder and clone a template scenario repository:

```
cd Dokumente/uni/ndnSIM_2-3
git clone http://github.com/named-data-ndnSIM/
  scenario-template.git voicemail_scenario
```

- The template repository has been named voicemail_scenario.

- Configure and run a scenario:

```
./waf configure --debug
./waf --run=<scenario>
./waf --run=<scenario> --vis
```

- How scenarios can be executed with a various number of arguments is shown later.

5 Execute a simulation (without Python script)

- Run the scenario with one of the following commands:

```
./waf --run=<scenario>
./waf --run=<scenario> --vis

for instance:
NS_GLOBALVALUE="RngRun=0" NS_LOG=BigMobileVoicemail
./waf "--run=big-mobile-voicemail --logDir=results/
/best-route_noAPs_3/run_0/
--fwStrategy=best-route --noAPs=3 --xPos=20
--yPos=-150 --yPosNext=-40 --isrT=50" --vis

NS_GLOBALVALUE="RngRun=0" NS_LOG=BigMobileVoicemail
./waf "--run=big-mobile-voicemail --logDir=results/
best-route_noAPs_3/run_0/" --vis
```

- Later the scenario is executed within a Python script (see: Execute all simulations with a Python script).

6 Execute all simulations with a Python script

- The executeSimulations.py script is used to execute all simulation runs. It can be found in the following directory:

```
Dokumente/uni/ndnSIM_2-3/voicemail_scenario/
  pythonScripts
```

- Execute the executeSimulations.py script with the following command:

```
cd Dokumente/uni/ndnSIM_2-3/voicemail_scenario
python ./pythonScripts/executeSimulations.py -r=2
  -t=4
python ./pythonScripts/executeSimulations.py -r=48
  -t=10 -s=2
```

- The script has to be executed in the voicemail_scenario (scenario) folder.
- The parameter -r sets the number of runs for each scenario configuration, the parameter -t sets the number of threads and the parameter -s sets the start number of runs for skipping a certain number of runs when they are executed already in the past.

7 Calculate metrics for each simulation (without Python script)

- Execute the calculateMetrics.py script for one configuration folder:

```
cd Dokumente/uni/ndnSIM-2-3/voicemail_scenario/
  results/best-route_noAP_1
python ../../pythonScripts/calculateMetrics.py
```

8 Calculate metrics for each simulation with a Python script

- Execute the executeCalculateMetrics.py script for each folder seperately:

```
cd Dokumente/uni/ndnSIM-2-3/voicemail_scenario/
  results
python ../pythonScripts/executeCalculateMetrics.py
  -noRuns=2 -dir="best-route_noAPs_1/"
python ../pythonScripts/executeCalculateMetrics.py
  -noRuns=2 -dir="best-route_noAPs_3/"
python ../pythonScripts/executeCalculateMetrics.py
  -noRuns=2 -dir="custom_noAPs_1/"
python ../pythonScripts/executeCalculateMetrics.py
  -noRuns=2 -dir="custom_noAPs_3/"
python ../pythonScripts/executeCalculateMetrics.py
  -noRuns=2 -dir="multicast_noAPs_1/"
python ../pythonScripts/executeCalculateMetrics.py
  -noRuns=2 -dir="multicast_noAPs_3/"
```

- The parameter -noRuns sets the number of runs for each scenario configuration and the parameter -dir sets the direcory in where the files are located.

9 Calculate results and compare all forwarding strategies for each configuration

- Execute the calculateResults.py script for calculating the results:

```
cd Dokumente/uni/ndnSIM_2-3/voicemail_scenario/
   results
python3 ../pythonScripts/calculateResults.py
  -bDir="best-route_noAPs_1"
  -mDir="multicast_noAPs_1"
  -cDir="custom_noAPs_1"
  -dstDir="resultCharts_noAPs_1"
```

- Here the python3 command is needed.
- The parameter -bDir sets the directory where the best-route forwarding strategy files are located, -mDir sets the directory where the multicast forwarding strategy files are located, -cDir sets the directory where the custom forwarding strategy files are located and -dstDir sets the directory where the results charts for a specific configuration are stored.