A beginners guide to installing and testing NLSR on Fedora

P. David Arjona-Villicaña*†, Octavio I. Renteria-Vidales*, Ashlesh Gawande‡
*Facultad de Ingeniería, Universidad Autónoma de San Luis Potosí
‡University of Memphis
†Email: david.arjona@uaslp.mx
November, 2018

Abstract

This is an easy to follow guide on how to install and test the NLSR routing protocol for NDN using two Fedora 28 machines. The commands and the paths used in this guide have been tested only in Fedora, although a similar procedure should be used for other linux systems. Specifically, the testing section (§V) is platform independent. Therefore users who have NLSR already installed may skip to this section. Since neither NDN, NLSR nor Fedora are static software developments, it is expected that some commands and features in this guide will change over time. However, this guide is as accurate as it could be at the time it was published.

I. Introduction

The following instructions are based on the information provided at the Named Data Networking project web page [1]. Before installing NLSR it is necessary to install different libraries and programs: ndn-cxx, NFD and ChronoSync. This document describes the necessary steps to correctly install these programs (§II, III and IV) and a brief guide on how to configure and test NLSR using a simple two-node network (§V).

Many of the commands in this guide need to be executed as *root*. Starting from Fedora 28, there is no need to set a root user [2]. However it is possible to define it, using the following command:

```
$ sudo -i
```

And then using the *passwd* command to set up a password. After setting up the root user, it is possible to become root by employing the following command and providing the root password when prompted:

```
$ su -
```

Fedora employs the *dnf* command to install and verify installed packages. The following commands may become useful during the installation process and should be executed as root:

- To verify that all installed packages are up to date
  `$ dnf update`
- To verify if a package is already installed in the system
  `$ dnf list --installed <package-name>`
- To search for information about a package or family of packages (not necessarily installed in the system).
  `$ dnf info <package-name>`
- To install a package in the system.
  `$ dnf install <package-name>`

It is also a good installation practice to download and install all the NDN programs in a common directory. Therefore, it is recommended to create a directory at the home directory where all the following programs may get installed. The name of this directory is not important. However, the following are

provided as suggestions: project, NDNproject, NDNprograms. Each of the programs in the following sections should be downloaded to their own directory using the *git* command, and then compiled in this directory.

II. Installing ndn-cxx

A. Before installing ndn-cxx

An updated list of the packages and programs that need to be installed before installing ndn-cxx, is provided at [3]. This list is also reproduced bellow with the commands to verify that all the packages have been installed in the system. The following commands should be run as root:

1) gcc and gcc-c++
   ```
   $ dnf list --installed gcc
   $ dnf list --installed gcc-c++
   ```
2) Python version 2.6 or later
   ```
   $ python -V
   ```
3) libsqlite3
   ```
   $ dnf list --installed sqlite
   $ dnf list --installed sqlite-devel
   ```
4) openssl version 1.0.1 or later
   ```
   $ openssl version
   $ dnf list --installed openssl-devel
   ```
5) pkgconf that replaces pkg-config
   ```
   $ dnf list --installed pkgconf*
   ```
6) boost libraries version 1.54 or later
   ```
   $ dnf list --installed boost
   $ dnf list --installed boost-devel
   ```
7) doxygen
   ```
   $ dnf list --installed doxygen
   ```
8) graphviz
   ```
   $ dnf list --installed graphviz
   ```
9) python2-sphinx that replaces python-sphinx
   ```
   $ dnf list --installed python2-sphinx
   ```
10) After verifying that the python-sphinx package has been installed, it is necessary to run the following two commands:
    ```
    $ pip-3 install sphinxcontrib-doxylink
    $ pip install sphinxcontrib-googleanalytics
    ```
    Note: When the two previous commands are run before correctly installing python-sphinx, the following error message is produced when trying to install this latter package: “Error unpacking rpm package python2-urllib3-1.22-3.fc27.noarch”. To fix this problem, it is necessary to run the following commands before installing python-sphinx again:
    ```
    $ dnf remove python2-sphinx
    $ pip uninstall sphinxcontrib-googleanalytics
    $ pip-3 uninstall sphinxcontrib-doxylink
    $ pip uninstall urllib3
    ```
11) dpkg-architecture
    ```
    $ dnf list --installed dpkg
    $ dnf list --installed dpkg-dev
    ```

12) git and valgrind, which is used when installing NFD (Section III).

```
$ dnf list --installed valgrind
$ dnf list --installed valgrind-devel
$ dnf list --installed git
```

B. Downloading and installing ndn-cxx

The *git* command allows to download the ndn-cxx library in its own folder, also called *ndn-cxx*. Therefore it is recommended to execute this command at the directory created at §I:

```
$ git clone --depth 1 https://github.com/named-data/ndn-cxx.git
```

Move to the ndn-cxx directory:

```
$ cd ndn-cxx
```

Use the following command to verify that everything is ready to compile. If an error message displays, it is necessary to fix it before continuing:

```
$ ./waf configure --with-examples
```

Compile the ndn-cxx library and install the compiled files at the system's directories:

```
$ ./waf
$ sudo ./waf install
```

C. Configuring ndn-cxx

The commands in this subsection need to be run as root. First, it is necessary to create file *local.conf*, which contains a line with the location for the ndn-cxx library:

```
$ echo /usr/local/lib64 >> /etc/ld.so.conf.d/local.conf
```

The following command configures the libraries:

```
$ ldconfig -v | grep ndn
```

This command should display a line similar to the following:

```
libndn-cxx.so.0.6.1 -> libndn-cxx.so.0.6.1
```

Configure the NDN path:

```
$ echo export PKG_CONFIG_PATH=/usr/local/lib64/pkgconfig >>
/etc/profile.d/ndn.sh
```

After this command has been executed, it is necessary to apply the changes by either logging out and back in, and then running the following command:

```
$ printenv | grep PKG
```

Or by executing the following command:

```
$ ./etc/profile.d/ndn.sh
```

For more information and examples about how to compile and configure this library, users should read the ndn-cxx guide at [3].

III. Installing NFD

A. Before installing NFD

An updated list of the packages and programs that need to be installed before NFD is provided at [4]. Before installing NFD it is necessary to verify that the following packages are installed:

1) libpcap libraries

```
$ dnf list --installed libpcap
$ dnf list --installed libpcap-devel
```

B. Downloading and installing NFD

This library is downloaded and installed in a folder called *NFD*, which should be created at the directory defined at §I. The following commands need to be run as a regular user:

```
$ git clone --depth 1 https://github.com/named-data/NFD.git
$ cd NFD
$ ./waf configure
```

If the previous command prints an error message saying that waf cannot find WebSocket, it is necessary to follow the instructions provided by this same output, which tells the user to execute either of the following two set of commands:

```
$ git submodule update --init
```

Or:

```
$ mkdir -p websocketpp
$ curl -L https://github.com/cawka/websocketpp/archive/
  0.8.1-hotfix.tar.gz > websocketpp.tar.gz
$ tar xf websocketpp.tar.gz -C websocketpp/ --strip 1
```

After executing either of these instructions, complete the configuration by running *./waf configure* again. Then complete the installation by means of the following commands:

```
$ ./waf
$ sudo ./waf install
```

C. Configuring NFD

Create a configuration file by running the following command as root:

```
$ cp /usr/local/etc/ndn/nfd.conf.sample /usr/local/etc/ndn/nfd.conf
```

After the configuration file has been created, NFD's behavior may be changed by modifying this file. Once the configuration file has been created, it is recommended to start NFD by using the following command:

```
$ nfd-start
```

This command does not properly allow to employ the command window to enter new commands; however it displays the NFD logs. Therefore, it is recommended to open a new command window. This second window may be used to verify NDF's status and then stop NFD by using the following commands:

```
$ nfd-status
$ nfd-stop
```

IV. Installing NLSR

A. Installing ChronoSync

Before installing NLSR, it is necessary to first download and install ChronoSync, which is a library that allows NLSR routers to verify that their routing information coincides. More information about ChronoSync may be found at [6]. This library may be installed by running the following commands as a regular user and at the directory defined at §I:

```
$ git clone --depth 1 https://github.com/named-data/ChronoSync.git
$ cd ChronoSync
$ ./waf configure
$ ./waf
$ sudo ./waf install
```

The following command needs to be used again to configure the libraries:

```
$ sudo ldconfig -v | grep -i chronosync
```

This command should display a line similar to the following:

```
libChronoSync.so.0.5.0 -> libChronoSync.so.0.5.0
```

B. Installing PSync

PSync has similar functionality than ChronoSync and it should replace this latter library in the future. More information about PSync may be found at [7]. This library may be installed by running the following commands as a regular user and at the directory defined at §I:

```
$ git clone --depth 1 https://github.com/named-data/PSync.git
$ cd PSync
$ ./waf configure
$ ./waf
$ sudo ./waf install
```

The following command needs to be used again to configure the libraries:

```
$ sudo ldconfig -v | grep -i psync
```

This command should display a line similar to the following:

```
libPSync.so.0.5.0 -> libPSync.so.0.5.0
```

C. Downloading and installing NLSR

NLSR is downloaded and installed in a folder called *NLSR* which should be created at the directory defined at §I. The following commands need to be run as a regular user:

```
$ git clone --depth 1 https://github.com/named-data/NLSR.git
$ cd NLSR
$ ./waf configure
$ ./waf
$ sudo ./waf install
```

D. Configuring NLSR

Create and configure the following directory by running the following commands as root:

```
$ mkdir /var/lib/nlsr
$ chmod 777 /var/lib/nlsr
```

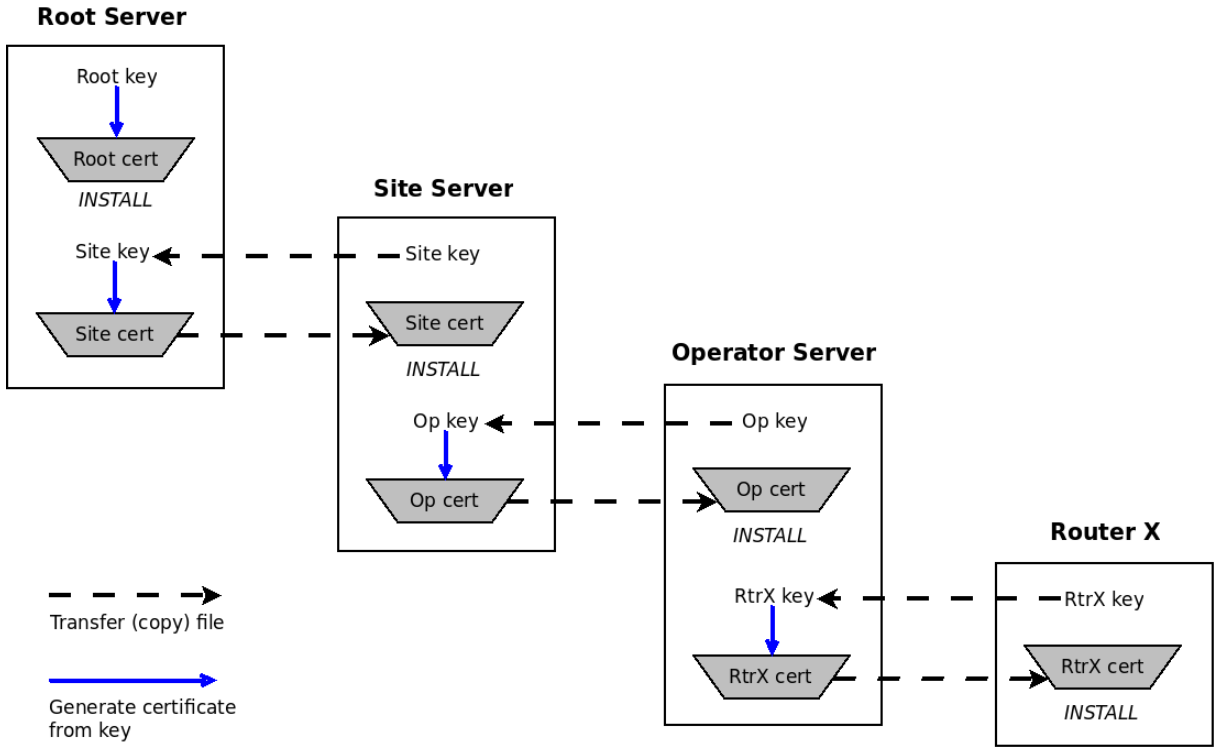

Fig. 1. Security configuration example for Router X.

V. Configuring and Testing NLSR

To test NLSR, the first step is to configure the keys and certificates that implement a secure communication between the routers. Then it is necessary to verify that the computers in the test network are connected, that NFD is running and the faces between the computers are configured. Finally, the NLSR configuration file has to be edited before running NLSR. The following subsections are provided as a guide to define and configure a simple computer network between two computers: router1 and router2.

A. Setting up the security

Configuring security in an NDN network requires to generate, exchange and install, keys and certificates between the root, site, operator and router computers that form the network [8], [9], although in practice, it is possible to keep more than one of these entities in a single machine. The following example and Figure 1 show how to configure security for a single router, called Router X. In this example, the root, site, operator and Router X are in different computers:

1) At the root server, generate the root key:
   ```
   $ ndnsec-key-gen /ndn/ > root.key
   ```
2) Generate the certificate for the root key at the root server:
   ```
   $ ndnsec-cert-dump -i /ndn/ > root.cert
   ```
3) Install the root certificate at the root server:
   ```
   $ ndnsec-cert-install -f root.cert
   ```
4) At the site server, generate the site key:
   ```
   $ ndnsec-key-gen /ndn/edu/uaslp > site.key
   ```
5) Copy the site key to the root server and generate the certificate for the site server:
   ```
   $ ndnsec-cert-gen -s /ndn/ site.key > site.cert
   ```
6) Copy the site certificate to the site server and install it:

```
$ ndnsec-cert-install -f site.cert
```

7) At the operator server, generate the operator key:
```
$ ndnsec-key-gen /ndn/edu/uaslp/%C1.Operator/op > op.key
```
8) Copy the operator key to the site server and generate the certificate for the operator server:
```
$ ndnsec-cert-gen -s /ndn/edu/uaslp op.key > op.cert
```
9) Copy the operator certificate to the operator server and install it:
```
$ ndnsec-cert-install -f op.cert
```
10) At the router, generate the router key:
```
$ ndnsec-key-gen /ndn/edu/uaslp/%C1.Router/routerX > routerX.key
```
11) Copy the router key to the operator server and generate the certificate for the router:
```
$ ndnsec-cert-gen -s /ndn/edu/uaslp/%C1.Operator/op routerX.key >
  routerX.cert
```
12) Copy the router certificate to the router and install it:
```
$ ndnsec-cert-install -f routerX.cert
```

In the previous steps, the *%C1.Router* and *%C1.Operator* labels are NDN keywords and should not be changed. These labels will be also used by the configuration file (§V-D)

The following command may be used to verify that the certificates have been installed in a computer:

```
$ ndnsec-list
```

This guide recommends that one machine functions as the root, site, operator and router1, while a different computer only functions as router2. Figure 2 shows this configuration. For router1, the twelve steps described before need to be executed except for exchanging files between computers. For the router2, only steps 10 to 12 are needed to generate this router's certificate.

Additionally, the following command may be used to print a list and a brief description of all the *ndnsec* commands:

```
$ man ndnsec
```

B. Configuring the network

The first step is to configure the physical network. If two computers are going to get connected using a single Ethernet cable, it is necessary to verify that this cable is a crossover. The other option is to employ

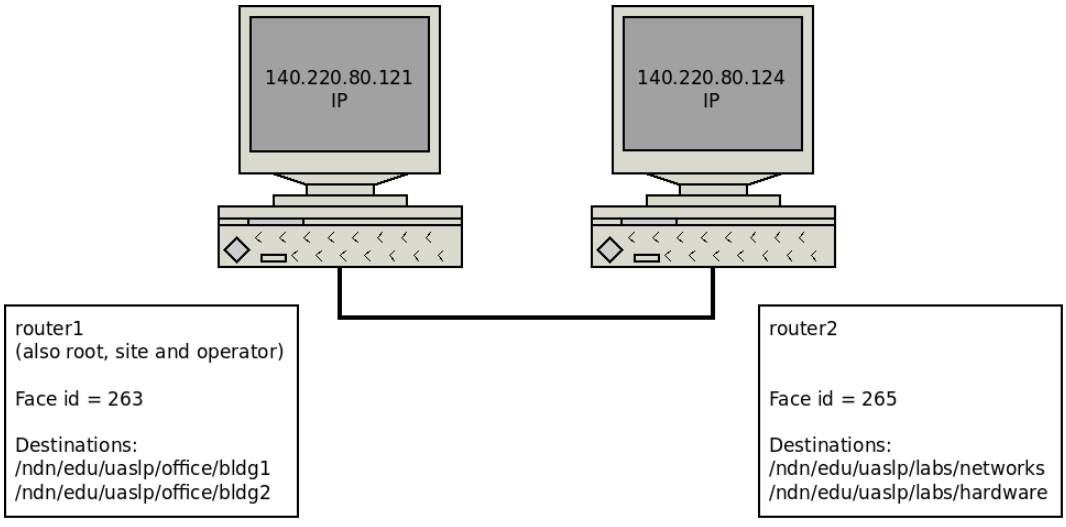

Fig. 2. Example network.

a switch between two computers that are then connected using two regular Ethernet cables.

After the physical network has been assembled, it is necessary to configure the network addresses and cards for all the computers in the network. It is important to remember that computers that are connected to each other should use the same subnetwork address. It is possible to verify the network configuration in a Linux computer by means of the *ip addr* command

Once the physical network and network cards have been configured, it is necessary to verify that the computers can communicate with each other. The simplest way to do this is by using the *ping* command:

```
$ ping <remote-ip-address>
```

C. Starting and configuring NFD

To start and configure NFD it is necessary to open two terminal windows. The first one will be used to start NFD by means of the *nfd-start* command. This terminal will also display the logs that NFD generates. By default, NFD only generates informational logs (INFO). However, it is possible to obtain different levels of verbosity for these logs. These levels can be set before NFD starts by editing the */usr/local/etc/ndn/nfd.conf* file. Open this file using a regular text editor, read the information provided about logging and then modify the *default-level* variable at the *log* section according to the instructions provided in the file. Additional information about NFD configuration may be found at [5].

The second terminal will be used to monitor the NFD status:

```
$ nfd-status
```

Employ the following command to configure each face that a computer uses to connect to a neighboring computer:

```
$ nfdc face create udp4://<remote-ip-address>
```

The face id may be displayed by running either *nfd-status* or:

```
$ nfdc face list
```

The status of the face may be verified by using the following command:

```
$ nfdc face show id <face-id>
```

After finishing NLSR testing, it is necessary to destroy the face before stopping NFD. This operation is described at §V-F. For the two computer network provided as an example (Figure 2), it is necessary that both machines run NFD and that each one configures a face that connects to the other machine.

D. Setting up the configuration file

Instructions on how to use the configuration file are already provided at the NLSR's Router Configuration page [10]. Read the information in this page to understand NLSR router configuration. The following text describes the instructions that have been modified at the default nlsr.conf file for router1:

```
; AT general SECTION:
{
  network /ndn/                    ; name of the network
  site /edu/uaslp                  ; name of the site
  router /%C1.Router/router1       ; name of the router: router1
}

;AT neighbors SECTION:
neighbors
{
```

```
  neighbor
  {
    name /ndn/edu/uaslp/%C1.Router/router2   ; Neighbor router: router2
    face-uri  udp://140.220.80.124           ; face to the neighbor
    link-cost 30                             ; cost of the link
  }
}

; AT advertising SECTION:
advertising
{
  prefix /ndn/edu/uaslp/office/bldg1         ; Advertising destinations
  prefix /ndn/edu/uaslp/office/bldg2         ; for router1
}

; AT security SECTION:
security
{
  validator
  {
    ...
    trust-anchor
    {
      type file
      file-name "root.cert"        ; root certificate file
    }
  }

  prefix-update-validator
  {
    ...
    trust-anchor
    {
      type file
      file-name "site.cert"        ; site certificate file
    }
  }

  cert-to-publish "root.cert"      ; root certificate file

  cert-to-publish "site.cert"      ; site certificate file

  cert-to-publish "op.cert"        ; operator certificate file

  cert-to-publish "router1.cert"   ; router1 certificate file
}
```

The following text shows the modified instructions for router2:

```
; AT general SECTION:
{
```

```
  network /ndn/                      ; name of the network
  site /edu/uaslp                    ; name of the site
  router /%C1.Router/router2         ; name of the router: router2
}

;AT neighbors SECTION:
neighbors
{
  neighbor
  {
    name /ndn/edu/uaslp/%C1.Router/router1   ; Neighbor router: router1
    face-uri  udp://140.220.80.121           ; face to the neighbor
    link-cost 30                             ; cost of the link
  }
}

; AT advertising SECTION:
advertising
{
  prefix /ndn/edu/uaslp/labs/networks        ; Advertising destinations
  prefix /ndn/edu/uaslp/labs/hardware        ; for router2
}

; AT security SECTION:
security
{
  validator
  {
    ...
    trust-anchor
    {
      type file
      file-name "root.cert"          ; root certificate file
    }                                ; this file needs to be copied to
  }                                  ; router2

  prefix-update-validator
  {
    ...
    trust-anchor
    {
      type file
      file-name "site.cert"          ; site certificate file
    }                                ; this file needs to be copied to
  }                                  ; router2

  ...
  cert-to-publish "router2.cert"     ; router2 certificate file
}
```

Notice that files *root.cert* and *site.cert*, which were generated at router1, need to be copied to router2. Also notice that the *%C1.Router* and *%C1.Operator* keywords employed at §V-A are also referenced by these configuration files.

E. Starting NLSR

It is recommended to open a third command terminal and run NLSR in this window. After the NLSR configuration file has been edited and saved as *nlsr.conf*, it is possible to start NLSR by running either of the following two commands:

```
$ nlsr
$ nlsr -f <configuration-file>
```

However, to verify what is NLSR doing, it becomes necessary to employ NLSR's logging facility [11]. A brief description on how to use NDN's logging facility may be displayed by entering the *man ndn-log* command. This guide recommends using one of the following two instructions to start NLSR:

```
$ export NDN_LOG=nlsr.*=TRACE && nlsr
$ export NDN_LOG=nlsr.*=TRACE && nlsr -f <configuration-file>
```

The second terminal window may be used to run *nfd-status* again and it should be possible to verify that the status has changed, specially at the FIB and RIB sections of the generated report.

F. Turning everything off

In order to stop NLSR and NFD, the following sequence of events is recommended:

1) Stop NLSR by pressing the Ctrl+C keys at the third terminal window.
2) Destroy the face to the remote computers using either of the following two commands at the second terminal window:
   ```
   $ nfdc face destroy <face-id>
   $ nfdc face destroy udp4://<remote-ip-address>
   ```
3) Stop NFD by entering the following command at the second terminal window:
   ```
   $ nfd-stop
   ```
4) The crossover Ethernet cable may be unplugged and the computers' network configuration restored to its original settings.

G. Where to go from here

Users interested in building and configuring larger networks may want to take a look at the NDN Ansible repository [12]. This repository uses Ansible, which is a configuration management tool, to manage the official NDN testbed deployment [13].

REFERENCES

[1] *Named Data Networking*, http://named-data.net/, March 2018.
[2] M. Kolman. *Anaconda improvements in Fedora 28*, Fedora Magazine, June 2018.
[3] *Getting started with ndn-cxx*, http://named-data.net/doc/ndn-cxx/current/INSTALL.html, April 2018.
[4] *Getting started with NFD*, http://named-data.net/doc/NFD/current/INSTALL.html, April 2018.
[5] *NFD usage*, http://named-data.net/doc/NFD/current/manpages/nfd.html, May 2018.
[6] Z. Zhu and A. Afanasyev. *Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking*, in IEEE ICNP, October 2013.
[7] *PSync*, https://github.com/named-data/PSync, January 2019.
[8] *NLSR Security Configuration*, http://named-data.net/doc/NLSR/current/SECURITY-CONFIG.html June 2018.
[9] V. Lehman, M. Chowdhury, N. Gordon, A. Gawande. *NLSR Developer's Guide*, University of Memphis, November 2017.
[10] *NLSR Router Configuration*, http://named-data.net/doc/NLSR/current/ROUTER-CONFIG.html, April 2018.
[11] *Getting Started with NLSR*, http://named-data.net/doc/NLSR/current/GETTING-STARTED.html, May 2018.
[12] *NDN Ansible repository*, https://github.com/WU-ARL/NDN_Ansible, October 2018.
[13] *NDN Testbed*, https://named-data.net/ndn-testbed/, October 2018.