Beginning with NFD

Joao Igor Pereira*

* LASSE - 5G & IoT Research Group, Federal University of Pará, Brazil
Email: igorsantos95@live.com

***Abstract*—Despite being a very useful tool for developers and researchers in NDN [3] area, NFD is a not very didactic tool, we can easily find how to install, articles about how it works, about the logical part of the program, even topics about some usual problems, but it is still poor the documentation about how to, in fact, use the tool, basic commands and about how to see a code working through it. The purpose of this document is to help NFD beginners that still doesn't had the "breakthrough" the hard start occasioned for the lack by the documentation about how to start using this tool, this document intends to be a practical guide for beginning to run codes in small NDN networks.**

***Keywords*—NFD,ICN, ndnSIM, NDN, ndn-cxx.**

I. Introduction

NFD is one of the most powerful tools for NDN researchers and developers since it can emulate the core of an NDN node, making the user able to test algorithms for all the jobs the NDN node core have to do and generate thought it more realistic results than many other tools. While NFD can emulate a more realistic NDN network than ndnSIM, for example, the amount of data about how to use it goes the opposite, while ndnSIM have a wide community with articles about how to use it, not just for being based in NS-3 a network simulator already widely used, but for this simulator itself, having even video lessons on youtube with hours about how to develop simple simulations, NFD is still corrected when searched on google. NFD is not an undocumented project it actually have a very well detailed guide, the NFD Developers Guide [5] which detail the program and its classes with a lot of graphs demonstrating about how the program works besides some commands, syntax and meanings and is the appropriate document to be read if you want to be an NFD master. But for a beginner, read all that data can be not just tiring, but there is the risk to read all of that, understand all the logic of the program and at the end still doesn't know how to do your own simulation.

II. Installation

You can find all the details about how to install NFD on NFD official site [2] but doing a synthesis, for Linux users the steps are below.

A. ndn-cxx

ndn-cxx [1] is a library written in C++ that implements NDN primitives and is being used in many NDN applications such as NFD, so to use NFD we have to install this library first.

1) Prerequisites:

- `python >= 2.6`
- `libsqlite3`
- `libcrypto++`
- `OpenSSL >= 1.0.1`
- `pkg-config`
- `Boost libraries >= 1.54`

You can install all the Prerequisites on Linux using the following command: sudo apt-get install build-essential libcrypto++-dev libsqlite3-dev libboost-all-dev libssl-dev. After installing the prerequisites, you can download ndn-cxx source code from the github ndn-cxx official repository with this command: git clone https://github.com/named-data/ndn-cxx. To build the library, enter in the folder that contains the ndn-cxx source code and use this commands in this order:

- `./waf configure`
- `./waf`
- `sudo ./waf install`

This commands check if all the configurations and prerequisites are ok, compile the codes and install the library. After the Installation, still in the source code folder, if you are a Linux user, type the command:

- sudo ldconfig

Now your ndn-cxx library should be installed and prepared to run.

B. NFD

After installing the ndn-cxx library we can move forward starting the installation of NFD itself

1) Prerequisites:

- `ndn-cxx`
- `pkconfig`
- `libpcap`

Commands to install these prerequisites:

- ndn-cxx installation is explained in the subsection above.
- `sudo apt-get install pkg-config`
- `sudo apt-get install libpcap-dev`

After installing the prerequisites, NFD source code can be downloaded from the NFD official repository with the command: git clone –recursive https://github.com/named-data/NFD (the –recursive attribute in the command is very important, don't forget to type it or you will have problems with the websocket folder). Having the prerequisites installed and the source code downloaded, to install NFD, use the same commands you used on the installation of ndn-cxx, but now on NFD source code folder

- `./waf configure`
- `./waf`
- `sudo ./waf install`

Now your NFD should be up to run. Run the NFD with the command: nfd-start If it shows any fatal errors, check if all the steps were executed right or check the NFD official site [2] for more information.

III. Starting to Use

This section will demonstrate how to start using NFD, but first let's remember that NFD is not a simulator, it is a forwarder, a program that emulates an NDN node, so to actually emulate an NDN network, we will need more than one NFD(emulate node) running, which means that we have to install it in at least two different computers or if you have just one computer available and to make the process easier, virtual machines. Normal virtual machines use too much hardware resources to be a suitable way to emulate a more wide network, in that case, the best tool to have some virtual machines running NFD without making them competing for all your hardware resources is docker [6].

A. Docker

Docker [6] is a software container platform, its containers are able to run softwares without needing to emulate the whole operational system, permitting to emulate machines without needing too much resources, that is why docker is a suitable solution to emulate networks with NFD in the case you have just one computer available. All the instructions about how to install and use docker are on Docker official website. (https://www.docker.com). All the commands below should be used with "sudo" since Docker needs root privileges. There are ways do give docker root privileges, but that's not the focus here.

1) Containers and Images: To use docker is important to know the difference between image and container since these are the main work objects of this tool.

- Image: Image is the "static" data of the machine, the saved part of a container that was running and can be run as a container with the run command
- Container: Container is the emulated machine itself, it is built from an image and works like a normal machine.

After installing docker and getting an image, to run that image in a container use: `docker run --rm --name <containerName> -it <imageName>:<version>`. The attribute –rm is optional, it makes the container stop running in the background when you exit it, if you don't want to kill the container after exit it don't use the –rm attribute. Into the terminal of at least one container, you have to do all the steps in the installation section, after that use: `docker commit <containerID> <imageName>:<version>` to save the current container as an image. To see the container ID use in a terminal outside a container the command: `docker ps` to see the containers that are running and its codes. Now you should have a docker image saved with NFD installed and you can use it to create as many containers you need or can to simulate your NDN network.

To create a docker network use `docker creat network <networkName>`, since you create a docker network once, you don't need to create again, like images, the networks structures are saved and can be used at anytime after the creation, but the containers connected to it, if killed, do not remain there. To connect a container to a created network the command `docker network connect <networkName> <containerName>`, to simulate an NDN network you should probably run and connect more than one container to the same network.

2) Main Docker Commands:

- `docker run --name <containerName> -it <imageName>:<version>` Run a saved image as a container
- `docker ps` Shows the containers which are running

- `docker images` Shows the saved images
- `docker commit <containerID> <imageName>:<version>` Commit the changes in the container as an image
- `docker network create <networkName>` Create a docker network
- `docker network connect <networkName> <containerName>` Connect a container to an existent docker network
- `docker rm <containerName>` Remove/kill a container running
- `docker rmi <imageName>` Delete a saved/-committed image
- `exit` Exit the terminal or the current container

B. NFD

NFD have a bunch of useful commands, but having all the steps above done, the main command you will have to use is `nfd-start` to get NFD running and `nfdc register /<interestName> udp4://<containerIP>` to connect the current container or machine to the container or machine with the given IP, specifically register in the FIB the IP to send the given interest, it is important to make sure that the machines or containers linked with this command are in the same network. To check the I of the docker containers you can use inside the container the same commands that are used in normal terminals, such as `ifconfig` .

IV. Starting to run codes

Before actually writing codes, it is recommended to first see a NFD application running and investigate its code to be aware of how the program and the applications work in fact. One application recommended for that is available on NFD official site, it is called `NDN Essential Tools`. All the prerequisites to run this tool should be ok if you installed the previous tools correctly. You can download the tool through its github repository. `git clone https://github.com/named-data/ndn-tools` after that enter the folder where you cloned it and do the same installations process as before.

- `./waf configure`
- `./waf`
- `sudo ./waf install`

NDN essential tools contains some basic applications that are described on its github repository as:

- peek: transmit a single packet between a consumer and a producer
- chunks: segmented file transfer between a consumer and producer
- ping: test reachability between two nodes
- dump: analyze traffic on wire
- dissect: inspect TLV structure of NDN packet format
- dissect-wireshark: Wireshark extension to inspect TLV structure of NDN packets
- pib: a service to manage the public information of keys and publish certificates

A. Running a sample

To see NDN Essential Tools working, lets use one of its tools, one of the most basic ones is the ping. To run the NDN Essential Tools ping, you will need two machines or two containers one acting like the ping client and the other like the ping server. To configure the ping server, enter in one of the two containers that you will use, start the NFD with the `nfd-start` command and connect the two containers with the `sudo docker network connect <> <>` command.

After that type `ndnpingserver /<interestName>` to set the interest to be responded, now your ndn ping server is up to run. To configure the ping client, you have to start the NFD with the `nfd-start` and then bind this container to the other recording on its FIB the other machine with the same interest expressed in the server `nfdc register /<interestName> udp4://<pingServerIP>` after recording it you can stat the ping with the command `ndnping <interestName>`

B. investigating the code

To investigate the ndn ping operation lets look at some functions in its file core code called `ping.cpp` on `ndn-tools/tools/ping/client` folder.

The first function we are going to talk about is the function that actually sends de interest packet, sends the ping, the function `performPing()`

```
void
Ping::performPing()
{
  BOOST_ASSERT((m_options.nPings < 0) || (m_nSent < m_options.nPings));

  Name pingPacketName = makePingName(m_nextSeq);

  Interest interest(pingPacketName);
  interest.setMustBeFresh(!m_options.shouldAllowStaleData);
  interest.setInterestLifetime(m_options.timeout);

  auto now = time::steady_clock::now();
  m_face.expressInterest(interest,
                  bind(&Ping::onData, this, _1, _2, m_nextSeq, now),
                  bind(&Ping::onNack, this, _1, _2, m_nextSeq, now),
                  bind(&Ping::onTimeout, this, _1, m_nextSeq));

  ++m_nSent;
  ++m_nextSeq;
  ++m_nOutstanding;

  if ((m_options.nPings < 0) || (m_nSent < m_options.nPings)) {
    m_nextPingEvent = m_scheduler.scheduleEvent(m_options.interval,
       bind(&Ping::performPing, this));
  }
  else {
    finish();
  }
}
```

As you can see, the function creates an interest name based on the current number of the request sequence and send it, after that it increments attributes like the number of ping sent and the sequence itself and verifies if a fixed number of pings were defined with the -c attribute in `ndnping` command, if it is not (nPings is equals to -1), it schedule pings until you exit the program, if it is, it pings until the number of Pings sent is equals to the number of pings defined in -c attribute. When `finish()` is hit, the program stop scheduling pings and finish your execution.

Another important function in this file is `onData()`.

```
  void
Ping::onData(const Interest& interest,
          const Data& data,
          uint64_t seq,
          const time::steady_clock::TimePoint& sendTime)
{
  time::nanoseconds rtt = time::steady_clock::now() - sendTime;
  afterData(seq, rtt);
  finish();
}
```

onData function is called everytime the ping is responded by the ping server and the data arrives at the client, this method is a standard method that in this program is implemented to just print the RTT and the name of the data that arrived, this method should and will be used in other application to process the incoming data . The file still have functions for time out and Nack, functions that called when these events occur, they are similar to onData but serving to these events, is it possible to set the timeout time overwriting the standard one using the attribute -o on ndnping command. There are other methods on the file such as `makePingName()` which is responsible for creating the name of the interest based on the sequence of pings, but editing this 4 methods cited you already can make your own NDN code over this application.

REFERENCES

[1] ndn-cxx: NDN C++ library with eXperimental eXtensions [Online]. Available: https://named-data.net/doc/ndn-cxx/current/

[2] Named Data Networking Forwarding Daemon. [Online]. Available: http://named-data.net/doc/NFD/current/

[3] NDN Named Data Networking project. [Online]. Available: http://www.named-data.net/

[4] NS-3 based Named Data Networking (NDN) simulator [Online]. Available: http://ndnsim.net/

[5] Afanasyev, A., Shi, J., Zhang, B., Zhang, L., Moiseenko, I., Yu, Y., Fan, C. (2014). NFD developers guide. Technical Report NDN-0021, NDN.

[6] Docker platform. [Online]. Available: https://www.docker.com/