

NDN Codebase and Tools

Introduction and getting started info

ALEX AFANASYEV
Florida International University
aa@cs.fiu.edu

Starting Point: <https://named-data.net/> ➔ Codebase

The screenshot shows the homepage of the Named Data Networking (NDN) website. The header features the NDN logo and navigation links for Project, Architecture, Codebase (which is highlighted), Testbed, Publications, and Discussion. The main content includes sections for the NDN Retreat 2016/Hackathon, Tutorial Videos, and a prominent "Named Data Networking (NDN) Project Newsletter for Summer 2016". A sidebar on the right provides links to various NDN components like NFD, NLSR, and ndnSIM, along with a FAQ section and a video player showing a Q&A session.

NAMED DATA NETWORKING

NDN RETREAT 2016 / HACKATHON

See presentation slides and the project results from the 6th NDN Retreat and 2nd NDN Hackathon in March 2016.

[Read More](#)

TUTORIAL VIDEOS

Watch tutorial videos about the NDN project and NDN technologies.

[Read More](#)

Named Data Networking (NDN) Project Newsletter for Summer 2016

<https://named-data.net/codebase/platform/>

Codebase

- Libraries/NDN Platform
- NFD: Forwarding Daemon
- NLSR: Link-state routing protocol
- Mini-NDN
- ndnSIM: NDN simulator
- Tools and Applications
- Documentation
- Github Source
- Redmine Issue Tracking System

Testbed

- ndn-cxx: C++ library
- NDN-CCL: Common Client Libraries
- ChronoSync
- ndnrte: Real Time Conferencing
- Consumer/Producer API
- FAQ
- Video: [A question about NDN answered on video by faculty, students, staff researchers, and colleagues.](#)
- Europe. [Read More](#)

Publications

Discussion

Events

search this site GO

NDN Codebase Overview

Infrastructure Software

NFD

NFD-
android

NDN-
RIOT

μNFD

NDN
Tools

NLSR

Repo-ng,
repo-sql

NDN
Control
Center

NDN Libraries

ndn-
cxx

NDN-
CPP

NDN-
JS

PyNDN

jNDN

Chrono
Sync

PSync

Vector
Sync

NDN-
RTC

Apps

ChronoChat

ndns

ndncert

ndn-
flow

NdnCon

ndn-fs

ndn-
atmos

Many
others

Evaluation Frameworks

ndnSIM

miniNDN

NDN Testbed

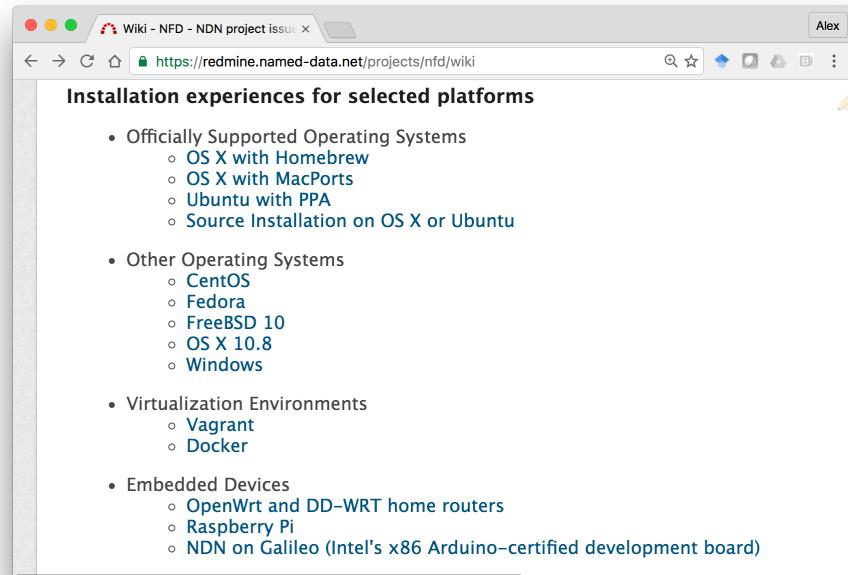
Where to Find Source Code for NDN Codebase

- Most linked from <https://named-data.net> ➔ Codebase
- Github organizations
 - <https://github.com/named-data>
 - NFD, core libraries, and other general use software
 - <https://github.com/named-data-mobile>
 - Android and related software
 - <https://github.com/named-data-iot>
 - IoT related software
 - <https://github.com/named-data-ndnsim>
 - ndnSIM core, example and real simulation scenarios

Supported Platforms

- Desktop Systems
 - Ubuntu, OSX, FreeBSD and other Linux distributions
- Home routers
 - OpenWRT, DD-WRT
- Mobile:
 - Android
 - iOS (library only)
- IoT:
 - Arduino, ESP8266
 - RIOT-OS
 - Raspberry Pi (runs NFD, available binary packages)
- Web browser
 - NDN-JS library + microforwarder
 - Firefox extension to support ndn:// URLs

<https://redmine.named-data.net/projects/nfd/wiki>



NDN Forwarding Daemon (NFD)

- The reference implementation of NDN forwarder
- <https://named-data.net/doc/NFD/current/>
 - Overview
 - Getting started
 - NFD Developer's Guide
 - Manpages
 - Wiki
 - API documentation (doxygen)

The screenshot displays three web pages from the NFD documentation site:

- Getting Started with NFD**: This page provides an overview of NFD, its installation on various platforms (MacPorts, NDN PPA, OS X), and how to start and stop it.
- NFD - Named Data Networking Forwarding Daemon**: A detailed guide for users, covering topics like NFD Overview, Getting Started with NFD (instructions for MacPorts and NDN PPA), and Additional documentation (FAQ, Manpages, NFD Developer's Guide).
- API Documentation (doxygen)**: A comprehensive API reference for NFD, listing classes, structures, unions, and interfaces with their descriptions.

• **Feedback, suggestions, and contributions are welcome.**

Getting Started with NFD

- To enable NDN communication in Ubuntu Linux:
 - sudo add-apt-repository ppa:named-data/ppa
 - sudo apt-get update
 - sudo apt-get install ndf
- Done. Now you have enabled new generation of networking on your machine
- Next required steps
 - Managing Identities for mandatory data-centric security
 - Self-signed certificate for local trust operations (home networking)
 - Authority-based
 - <https://yoursunny.com/t/2016/ndncert/>

```
vps3 $ ndnsec list -c
* /ndn/edu/arizona/cs/shijunxiao
  +->* /ndn/edu/arizona/cs/shijunxiao/ksk-1457557007329
  +->* /ndn/edu/arizona/KEY/cs/shijunxiao/ksk-1457557007329/ID-CERT/%FD%00%00%01S%5D+%B3
```

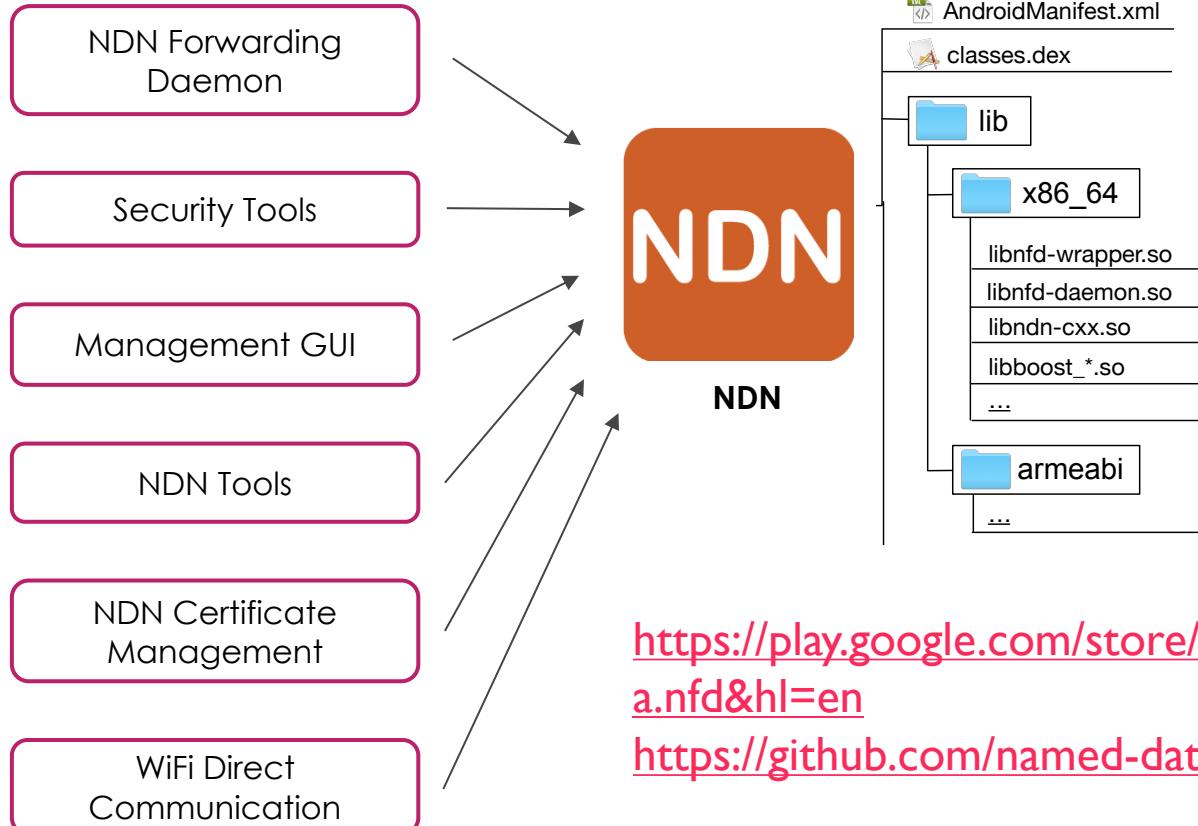
I want to issue a certificate to my other machine, sunnyq. To do that, I first generate a certificate request on sunnyq:

```
sunnyq $ ndnsec key-gen -tr /ndn/edu/arizona/cs/shijunxiao/sunnyq > sunnyq.ndncertreq
```

Then I copy the certificate request file to vps3, and issue the certificate by signing it with my existing trusted certificate:

```
vps3 $ ndnsec cert-gen -N '' -s /ndn/edu/arizona/cs/shijunxiao - < sunnyq.ndncertreq > sunnyq
```

NDN-Android: NDN Stack for Android

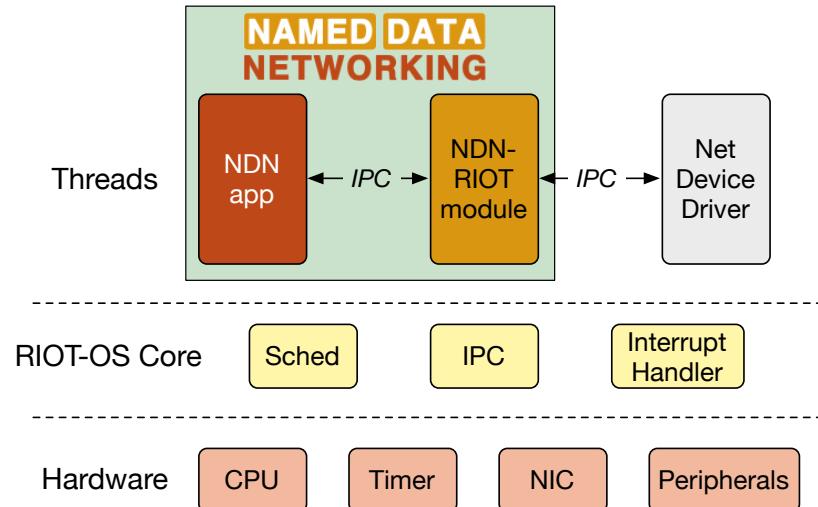


- Embeds actual NFD, compiled using NDK
- Works with all (non-rooted) Android devices

https://play.google.com/store/apps/details?id=net.named_data.nfd&hl=en
<https://github.com/named-data/mobile>

NDN-RIOT: NDN for RIOT-OS

- Optimized for IoT apps
- Support
 - Data-centric security
 - Stateful NDN packet forwarding
 - Replaceable forwarding strategies
 - 802.15.4 and Ethernet
- Simple application APIs
- Several simple examples to get started



<https://github.com/named-data-iot>

Getting Started with NDN-RIOT Examples

- Downloading

- `mkdir riot`
- `cd riot`
- `git clone https://github.com/named-data-iot/RIOT`
- `git clone https://github.com/named-data-iot/ndn-riot`
- `git clone https://github.com/named-data-iot/ndn-riot-examples`

- Compiling an example

- `cd ndn-riot-examples/<APP>`
- For host architecture (for debugging)
 - `make`
- For a specific RIOT board
 - `make BOARD=samr21-xpro`
 - `make flash BOARD=samr21-xpro # to flash firmware`
 - `make term BOARD=samr21-xpro # to access board via serial interface`

[ndn-benchmark](#)

[ndn-consumer](#)

[ndn-ping](#)

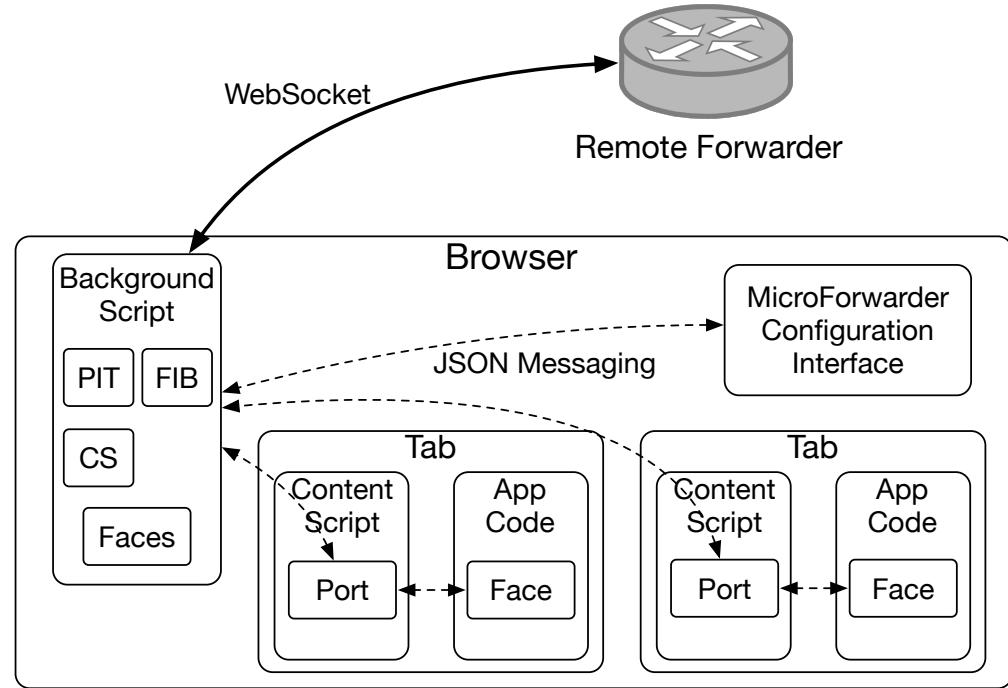
[ndn-producer](#)

[ndn-rtt](#)

[ndn-template](#)

NDN Micro Forwarder in browsers

- NDN forwarder as a Firefox/Chrome extension, written in JavaScript
- The cross-browser plugin (built upon NDN.js and the WebExtensions API) provides shared connectivity to remote forwarders and enables shared data cache
- Allows browser tabs to communicate with each other through a local channel even when remote connectivity is lost



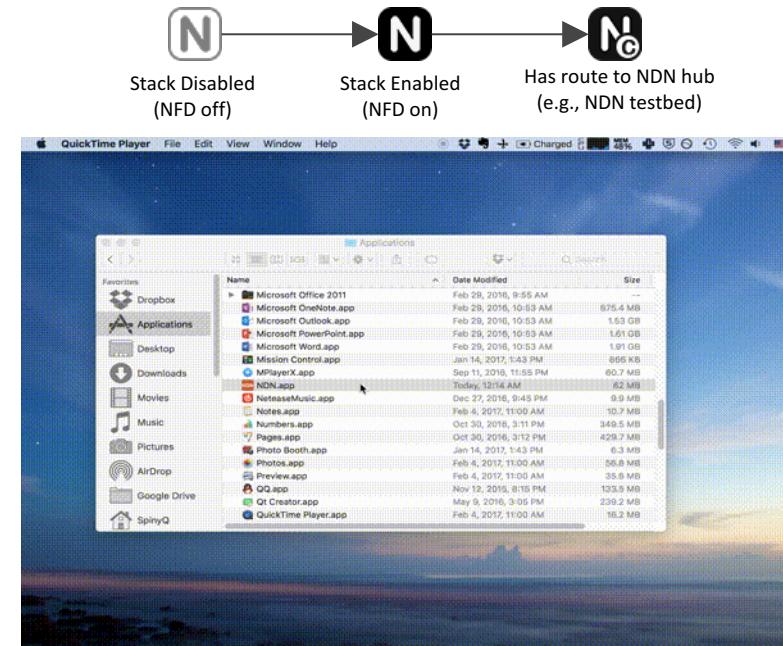
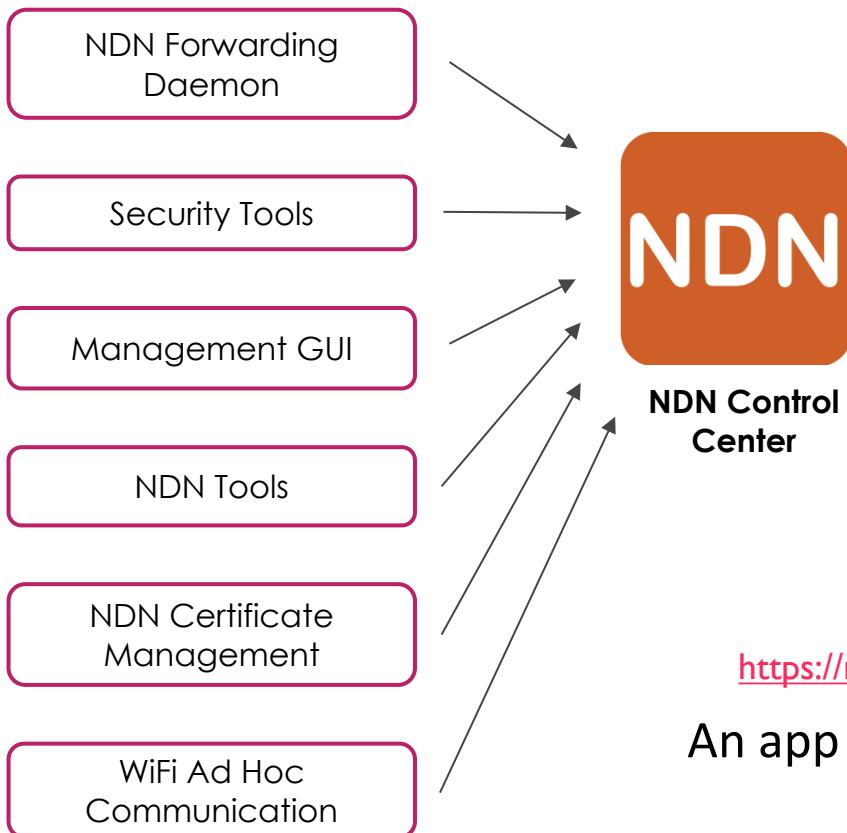
<https://github.com/named-data/ndn-js/tree/master/tools/micro-forwarder>

NDN Tools

- ndnping, ndnpingserver
 - Reliability testing tools
- ndncatchunks, ndnputchunks
 - Segmented file transfer between a consumer and producer
- ndnpeek, ndnpoke
 - Transmit a single packet between a consumer and a producer
- ndndump, dissect, wireshark-dissect
 - Debug NDN packet flow
- repo-ng, repo-sql: NDN repositories providing managed persistent storage



NDN Control Center (macOS, Linux)



<https://named-data.net/codebase/applications/ndn-control-center/>

An app that provides a simple way to start playing with NDN apps

Libraries

ndn-cxx: NDN C++ library with eXperimental eXtensions

- C++11
- The reference library and security library implementation
- Used in: NFD, NLSR, ndn-tools, ChronoChat, etc.
- <https://named-data.net/doc/ndn-cxx/current/>
 - Overview
 - Getting started
 - Trivial applications
 - Tutorials
 - Specifications
 - Manpages
 - API documentation (doxygen)
- Feedback, suggestions, and contributions are welcome.

The screenshot shows two browser windows. The top window displays the Redmine issue tracker for the ndn-cxx project, showing a list of issues with columns for Subject, Status, and Description. The bottom window displays the ndn-cxx documentation website, featuring a navigation bar with 'Namespaces' selected, a table of contents on the left, and detailed pages for various components like 'Namespaces' and 'Trivial applications' on the right.



- NDN protocol stack
 - NFD as NDN packet mux/demuxer
 - Name prefixes / interface (routing) configuration
 - (Discovery of local hub & prefixes)
 - (Local data prefixes propagation)
- **Identity/Certificate**
 - To ensure data-centric security

- Client

- `ndn::Face face;`
 - // start async
 - `face.expressInterest(ndn::Interest("/some/name").setMustBeFresh(true), onData, onNack, onTimeout);`
- `face.processEvents();`
- // async onData
- // async onNack
- // async onTimeout

- Server

- `ndn::Face face;`
 - // start async
 - `face.setInterestFilter("/some/name", onInterest, onSuccess, onFailure);`
- `face.processEvents();`
- // async onInterest
 - `ndn::Data("/some/name");`
 - `data.setContent(...);`
 - **keyChain.sign(data);**
 - `face.put(data);`

NDN Common Client Libraries (NDN-CPP, NDN-JS, jNDN, PyNDN)

- C++, Java, Python, JavaScript, C#, Squirrel
- Used in: NDN-RTC, NdnCon, NFD-Android, etc.
- <https://named-data.net/codebase/platform/ndn-ccl/>
 - NDN Common Client Libraries API
 - NDN-CPP API
 - PyNDN API
 - NDN-JS API
 - jNDN API

The screenshot displays two browser windows side-by-side, both showing the "NDN Common Client Libraries API" documentation. The left window shows the main "NDN Common Client Libraries (NDN-CCL) Documentation" page, which includes an introduction, a list of libraries implementing the API (C++ - NDN-CPP, Python - PyNDN, JavaScript - NDN-JS, Java - jNDN), function and class documentation, and potential contributors' guidelines. The right window shows a detailed view of the "Name Class" section under the "NDN Common Client Libraries API" table of contents, listing various constructor methods: "Name Constructor (array of components)", "Name Constructor (from URI)", and "Name Constructor (copy)". A vertical sidebar on the right lists numerous other NDN classes and components.

NDN Common Client Libraries (NDN-CCL) Documentation

The NDN Common Client Libraries (NDN-CCL) are written in C++, Python, JavaScript and Java and provide a common API for client applications to use NDN. Any library in NDN-CCL suite allows an application to send interests to and receive data from an NDN forwarding daemon (NFD) and provide a large set of other functions necessary for any NDN application.

Libraries implementing the NDN Common Client Libraries API:

- C++ – [NDN-CPP](#), [language-specific issues]
- Python – [PyNDN](#)
- JavaScript – [NDN-JS](#)
- Java – [jNDN](#)

Function and class documentation: [NDN-CPP](#), [PyNDN](#), [NDN-JS](#), [jNDN](#).

Potential contributors to the NDN-CCL should review the [NDN-CCL Development Guidelines](#).

Supported Features

Feature	NDN-CPP	PyNDN	NDN-JS	jNDN	Notes
MemoryContentCache	✓	✓	✓	✓	
ChronoSync2013	✓	✓	✓	✓	
Name.Component from*	✓	✓	✓	✓	
Name.Component Is*	✓	✓	✓	✓	
Name.Component to*	✓	✓	✓	✓	
ImplicitSha256DigestComponent	✓	✓	✓	✓	
ProtobufTlv	✓ API	✓ API	✓ API	✓ API	
SegmentFetcher	✓ API	✓ API	✓ API	✓ API	

- Client

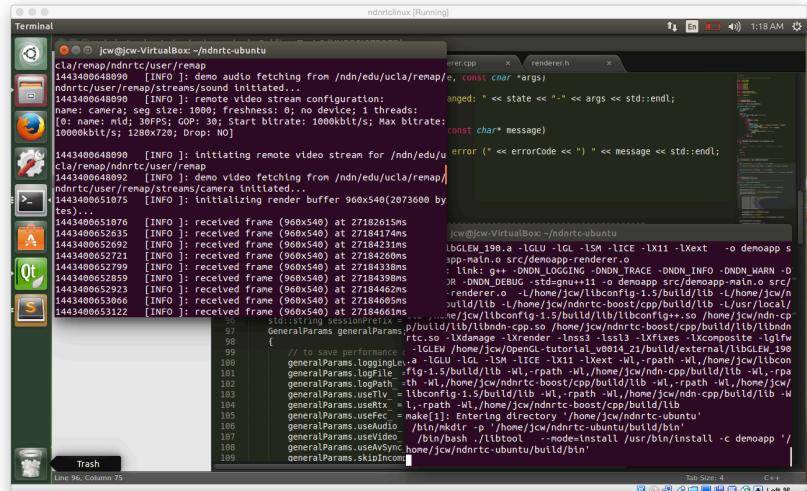
- `loop = asyncio.get_event_loop()`
- `face = ThreadsafeFace(loop, None)`
 - // start async
 - `face.expressInterest(ndn.Interest(ndn.Name("/some/name")), onData, onNack, onTimeout)`
- `loop.run_forever()`
- `face.shutdown()`
- // async onData
- // async onNack
- // async onTimeout

- Server

- `loop = asyncio.get_event_loop()`
- `face = ThreadsafeFace(loop, None)`
- // start async
 - `face.registerPrefix(ndn.Name("/some/name"), onInterest, onFailure)`
- `loop.run_forever()`
- `face.shutdown()`
- // async onInterest
 - `data = ndn.Data("/some/name");`
 - `data.setContent(...);`
 - **keyChain.sign(data, ...);**
 - `face.putData(data);`

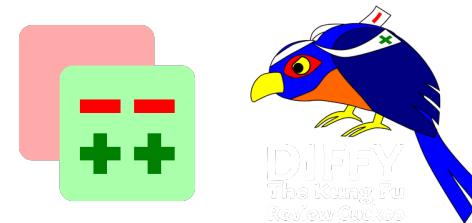
NDN-RTC

- C++ library for low-latency audio/video streaming over NDN
 - VP8/9 encoder
 - WebRTC audio processing pipelines
 - Forward error correction (OpenFEC)
 - Pull-based streaming control by consumers



Main collaboration tools

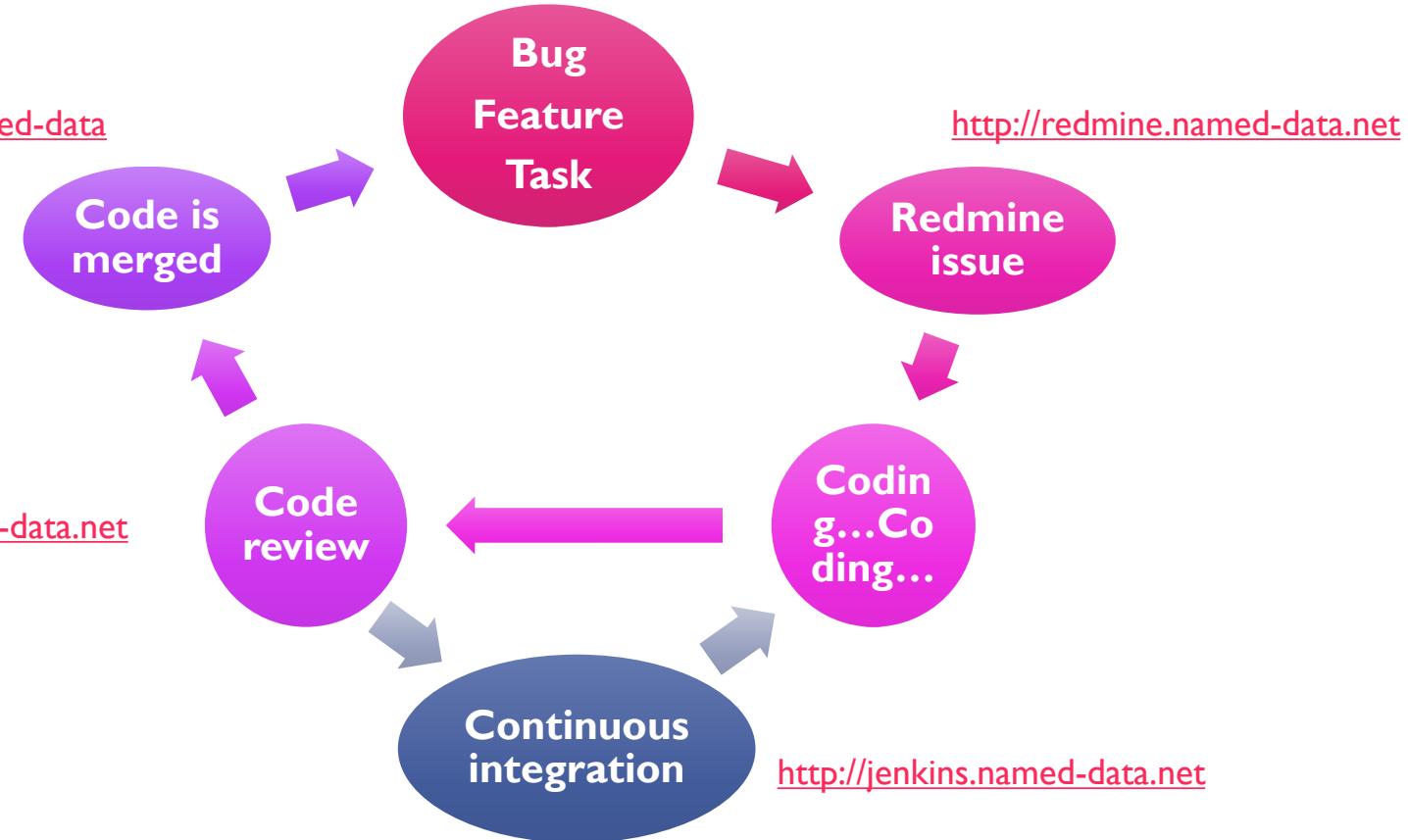
- **Redmine** – issue tracking, wiki
 - Coordination of NDN code development across the world
 - > 3100 issues (>2400 closed)
- **Gerrit** – code review
 - Ensuring code quality and high-quality coding training
 - > 4100 changes
- **Mailing lists** – wider discussions, announcements



<https://named-data.net/codebase/platform/support/mailing-lists/>

Development Model

<https://github.com/named-data>



Evaluation at Different Scales

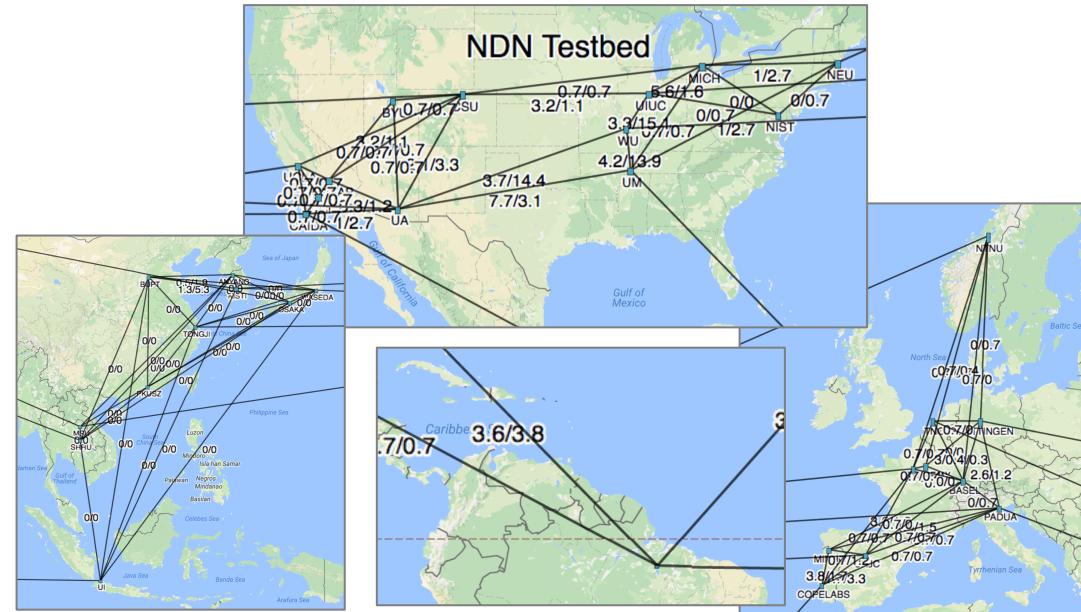
NDN Testbed

- Network of 37 sites across 4 continents, 14 countries

Open to join and use

<https://named-data.net/ndn-testbed/policies-connecting-nodes-ndn-testbed/>

- Examples applications and experiments: videoconferencing, network management, virtual machine migration, strategies, nTorrent, etc.
- Small scale evaluations



Open Network Lab (ONL)

- Remotely accessible network testbed
 - Operated and maintained by Applied Research Lab in Department of Computer Science and Engineering at Washington University in St. Louis
 - Real Hardware for running repeatable network experiments with trusted results. (NOT simulations)
- Use for NDN
 - NDN installed on each host/VM
 - NFD performance study
 - NDN Testbed Emulation to test new releases
- How to join?
 - <https://onl.wustl.edu/>
 - And “Get an account”



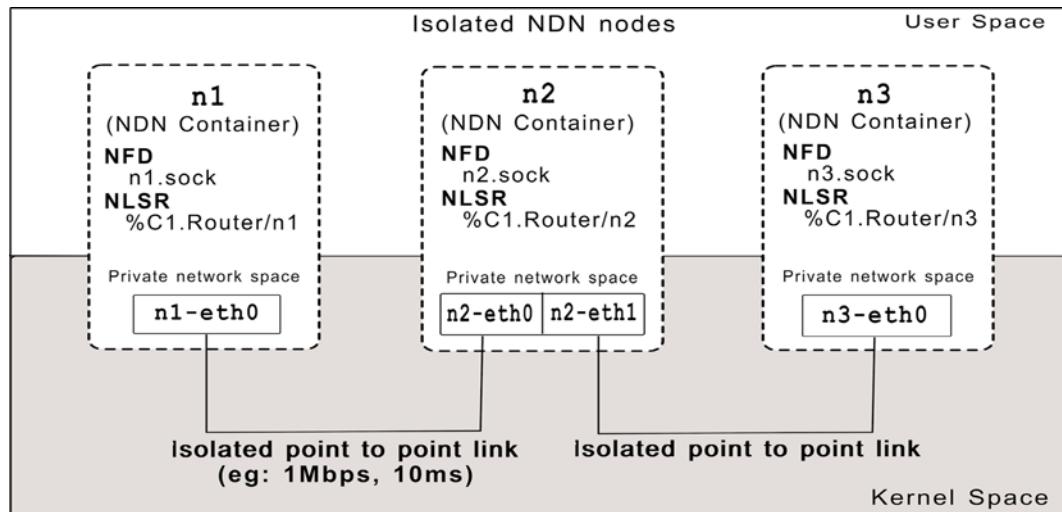
MiniNDN: NDN Emulation Framework (Based on MiniNet)

<https://github.com/named-data/mini-ndn>

Runs actual instances of NFD, NLSR

Medium-scale evaluations

- Easy to configure network emulation
- Runs any real application
- Number of emulated nodes \propto CPU power
- Cluster edition can be used to scale emulations





ndnSIM: NDN Simulation Framework (Based on NS-3)

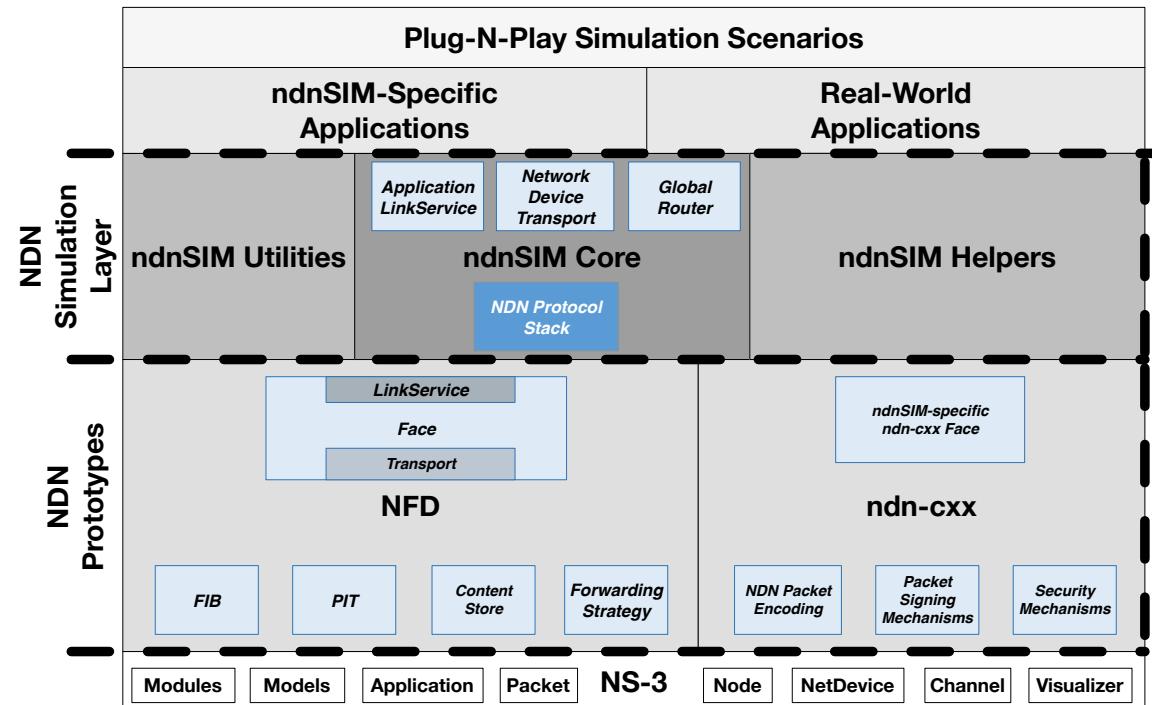
<https://ndnsim.net/>

Fully integrated with NDN

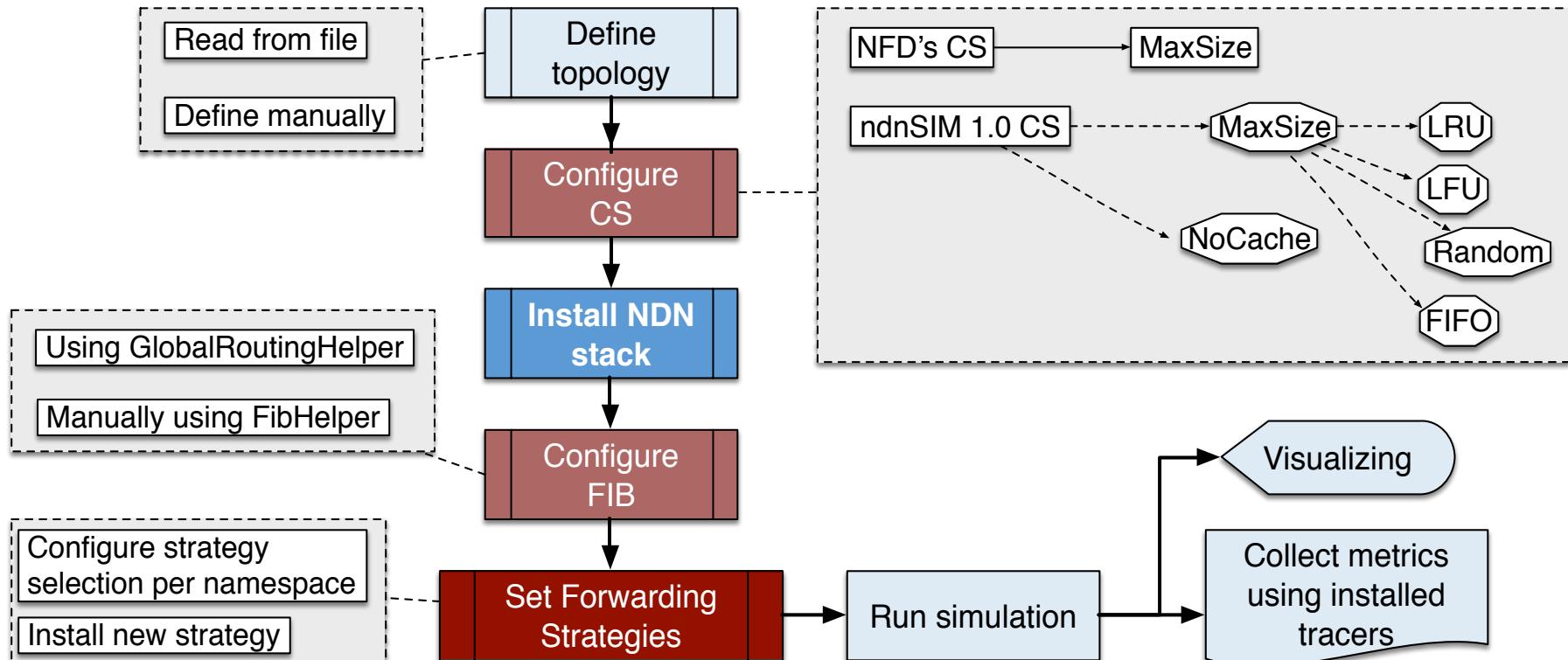
prototype implementations: NFD & ndn-cxx

Large scale evaluations

- Provide interoperability between simulation and prototyping
- Enable a two-way of experimentation and evaluation
- Enable high-fidelity NDN simulations
- 1500+ nodes with WiFi channels in the evaluation of NDN for vehicular networking



Typical Workflow with ndnSIM



Install consumer app(s)

```
NodeContainer consumerNodes;  
consumerNodes.Add(grid.GetNode(0,0));  
ndn::AppHelper cHelper("ns3::ndn::ConsumerCbr");  
cHelper.SetPrefix("/prefix");  
cHelper.SetAttribute("Frequency",  
    StringValue("10"));  
cHelper.Install(consumerNodes);
```

Install NDN stack

```
ndn::StackHelper ndnHelper;  
ndnHelper.InstallAll();
```

Install producer app(s)

```
Ptr<Node> producer = grid.GetNode(2, 2);  
ndn::AppHelper pHelper("ns3::ndn::Producer");  
pHelper.SetPrefix("/prefix");  
pHelper.SetAttribute("PayloadSize",  
    StringValue("1024"));  
pHelper.Install(producer);
```

Configure FIB (manually or like here using the helper)

```
ndn::GlobalRoutingHelper ndnGlobalRoutingHelper;  
ndnGlobalRoutingHelper.InstallAll();  
ndnGlobalRoutingHelper.AddOrigins("/prefix", producer);  
ndnGlobalRoutingHelper.CalculateRoutes();
```