Experimenting with NDN Apps using Mini-NDN

NDN Tutorial – ACM ICN 2016

September 26th, 2016, Kyoto, Japan

Davide Pesavento
Pierre and Marie Curie University, Paris, France

https://named-data.net/icn2016-tutorial

Overview

- Mini-NDN provides a network emulation environment for NDN experimentation
- A full NDN network can be run on a single system (laptop, server, etc.)
- Each node in the network can run forwarding, routing, and NDN applications
- Independent of changes in NDN platform
- NDN experiments can be performed more easily and much quicker than previously used tools

Mininet

- A popular network emulation tool
  - Process-based virtualization (containers); abstracts a single host as multiple nodes in a network
- Easy-to-use API to build desired networks
  - Nodes run as bash process inside Linux network namespaces (netns)
  - Links are virtual Ethernet pairs (veth)
- Provides tools to configure CPU and memory allocation for each node
- Can be installed on a VM (official images based on Ubuntu are provided), or natively using distro packages or Mininet install script

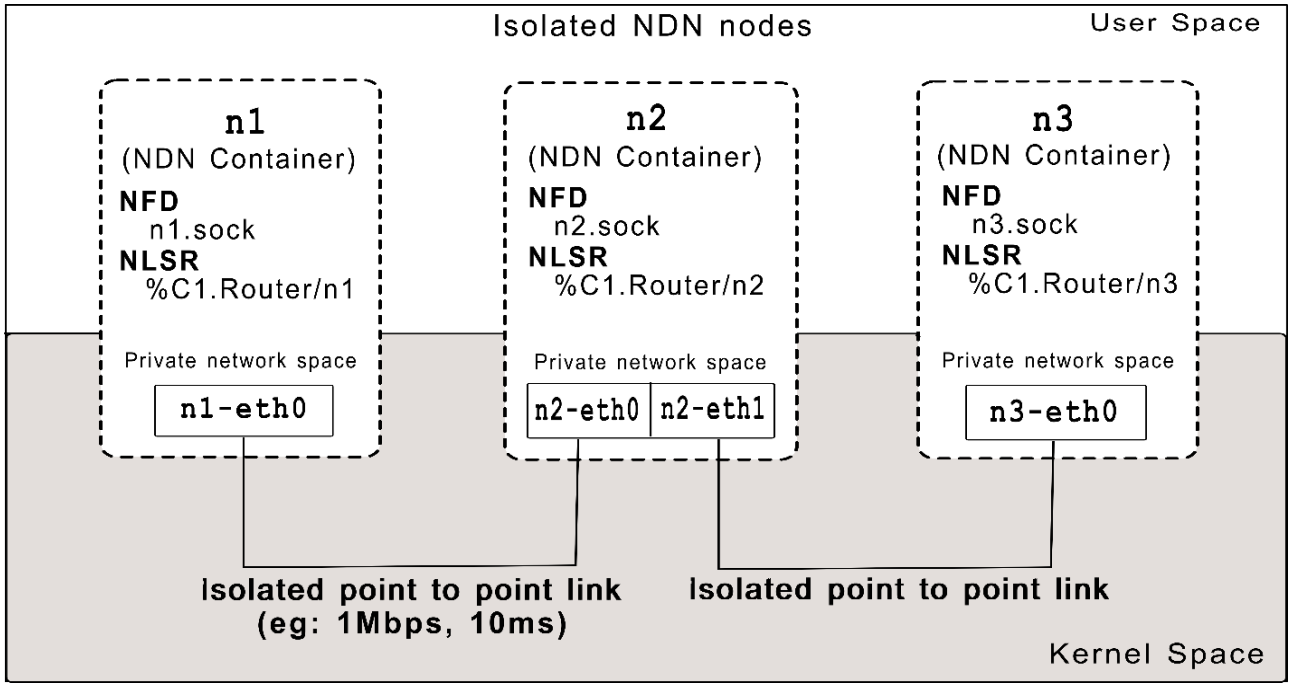
http://mininet.org/
https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

Mini-NDN

- Enables NDN research on top of Mininet
- Mini-NDN loads a user defined topology file to configure a network and runs NFD+NLSR on each node
- Topology file can be used to configure static routes
- A GUI tool (MinindnEdit – an extension of MininetEdit) is provided to help generate the topology file

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

Mini-NDN


Isolated NDN nodes
User Space
n1
(NDN Container)
NFD
n1.sock
NLSR
%C1.Router/n1
Private network space
n1-eth0
n2
(NDN Container)
NFD
n2.sock
NLSR
%C1.Router/n2
Private network space
n2-eth0
n2-eth1
n3
(NDN Container)
NFD
n3.sock
NLSR
%C1.Router/n3
Private network space
n3-eth0
Isolated point to point link
(eg: 1Mbps, 10ms)
Isolated point to point link
Kernel Space

Running NFD on each node

- Need to run multiple copies of NFD on one system
  - NO mount namespaces
  - Use a different home folder for each NFD instance:
    - `/tmp/<nodename>`
  - Use different socket file for each NFD instance in nfd.conf
  - Add the socket file path to client.conf used by NDN applications

Scalability

- Since Mini-NDN runs multiple instances of NFD and NLSR, scalability depends on the number of CPUs available
- The more CPU available to the processes, the better the results
- For example, an 8-core machine is able to scale to 100 nodes with decent traffic (say each node pinging 10 other nodes)
- Ongoing work to improve scalability

Comparison with other tools

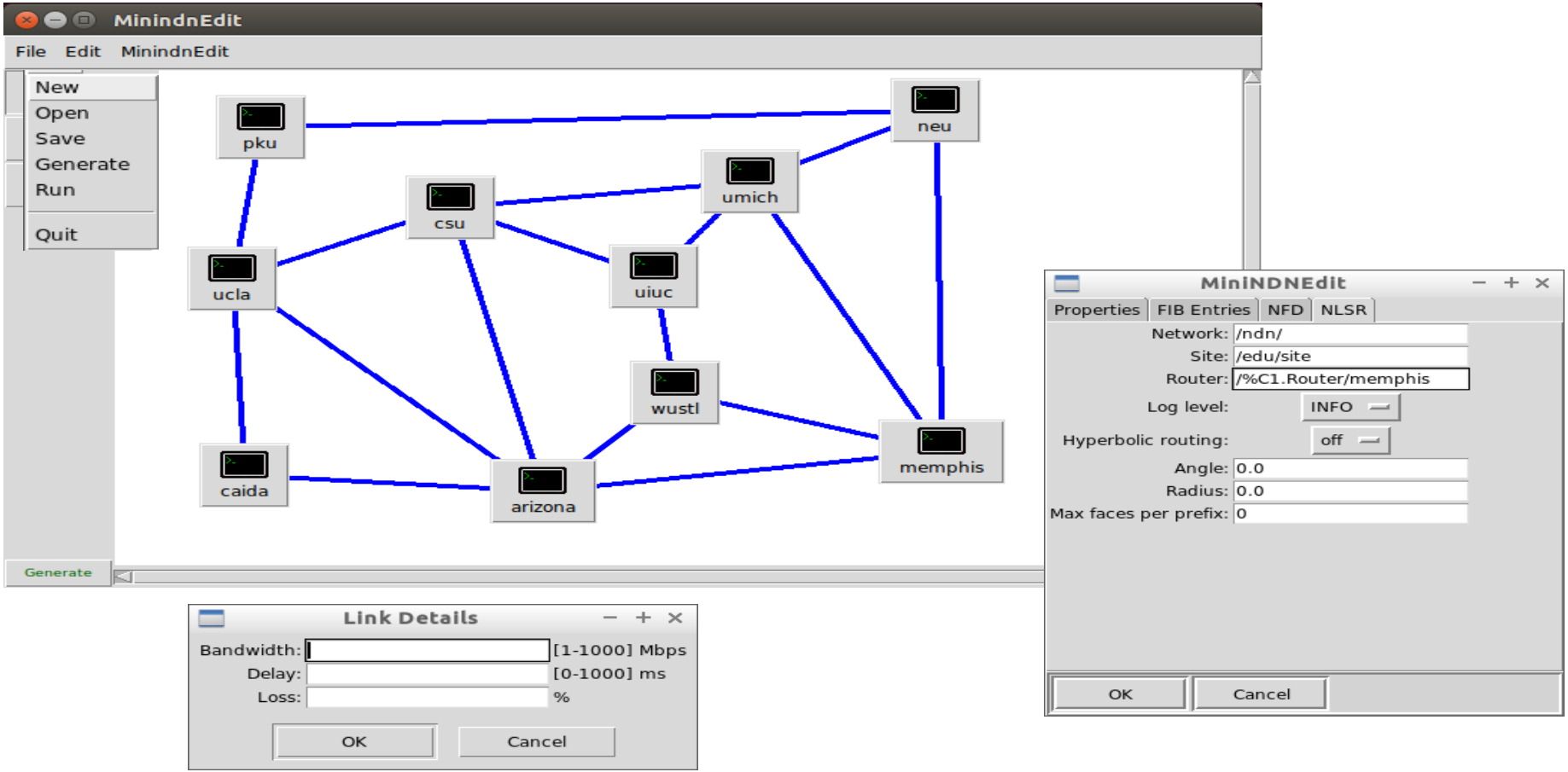
- ndnSIM
  - ndnSIM scales better than Mini-NDN
  - Applications require porting to run on ndnSIM
  - ndnSIM uses discrete time steps; applications may run differently from real world
  - Mini-NDN allows for applications with user interaction (e.g. GUI)

- Virtual machines, Vagrant, Docker, …
  - VMs need more resources
  - Requires scripts to setup networking
  - Manual NFD and NLSR configuration, or need to write a script to automate the process
  - No central controller – experiments require synchronization

- Testbed
  - Requires to maintain up-to-date images
  - Slower to boot/restart
  - Scales poorly
  - Resources may not be available

Experiment workflow (demo)

1. Create topology file
2. Write experiment class
3. Start experiment
   - `sudo minindn <topology> --experiment=<expname>`
4. (optionally) Interact with the emulated network
5. Collect/process results
   - By default, logs produced by one node are written to that node's home directory (`/tmp/<nodename>`)
   - Otherwise, use absolute paths


MinindnEdit
File
Edit
MinindnEdit
New
Open
Save
Generate
Run
Quit
pku
neu
umich
csu
ucla
uiuc
wustl
caida
arizona
memphis
Generate
Link Details
Bandwidth:
[1-1000] Mbps
Delay:
[0-1000] ms
Loss:
%
OK
Cancel
MiniNDNEdit
Properties
FIB Entries
NFD
NLSR
Network:
/ndn/
Site:
/edu/site
Router:
/%C1.Router/memphis
Log level:
INFO
Hyperbolic routing:
off
Angle:
0.0
Radius:
0.0
Max faces per prefix:
0
OK
Cancel

```
class TutorialExperiment(Experiment):
    def __init__(self, args):
        Experiment.__init__(self, args)

    def run(self):
        for host in self.net.hosts:
            host.cmd("nfd-status > nfd-status.out &")

Experiment.register("tutorial", TutorialExperiment)
```