

ns-3 training

Tom Henderson

ns-3 annual meeting 2019

June 17-21, Florence, Italy



Next steps

- > Code organization and build system
- > Documentation system
- > Packet objects and queues
- > Walkthrough of 'mm1-queue.cc' example
 - Simple experiment management
 - Objects, attributes, tracing
 - Logging and debugging



Software orientation

Key differences from other network simulators:

1) Command-line, Unix orientation

- vs. Integrated Development Environment (IDE)

2) Simulations and models written directly in C++ and Python

- vs. a domain-specific simulation language

ns-3 not written in a high-level language

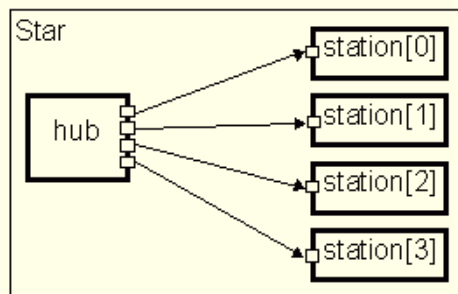
Submodule vectors, gate vectors and multiple connections are illustrated in the following example:

```
simple Hub
  gates:
    out: output[];
endsimple

simple Station //...

module Star
  submodules:
    hub: Hub
    gatesizes: output[4];
    station: Station[4];
  connections:
    for i=0..3 do
      hub.output[i] --> station[i].in;
    endfor
endmodule
```

The result of the above is depicted in Fig.4.



Example of OMNeT++ Network Description (NED) language

Figure excerpted from <http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm>

ns-3 does not have a graphical IDE

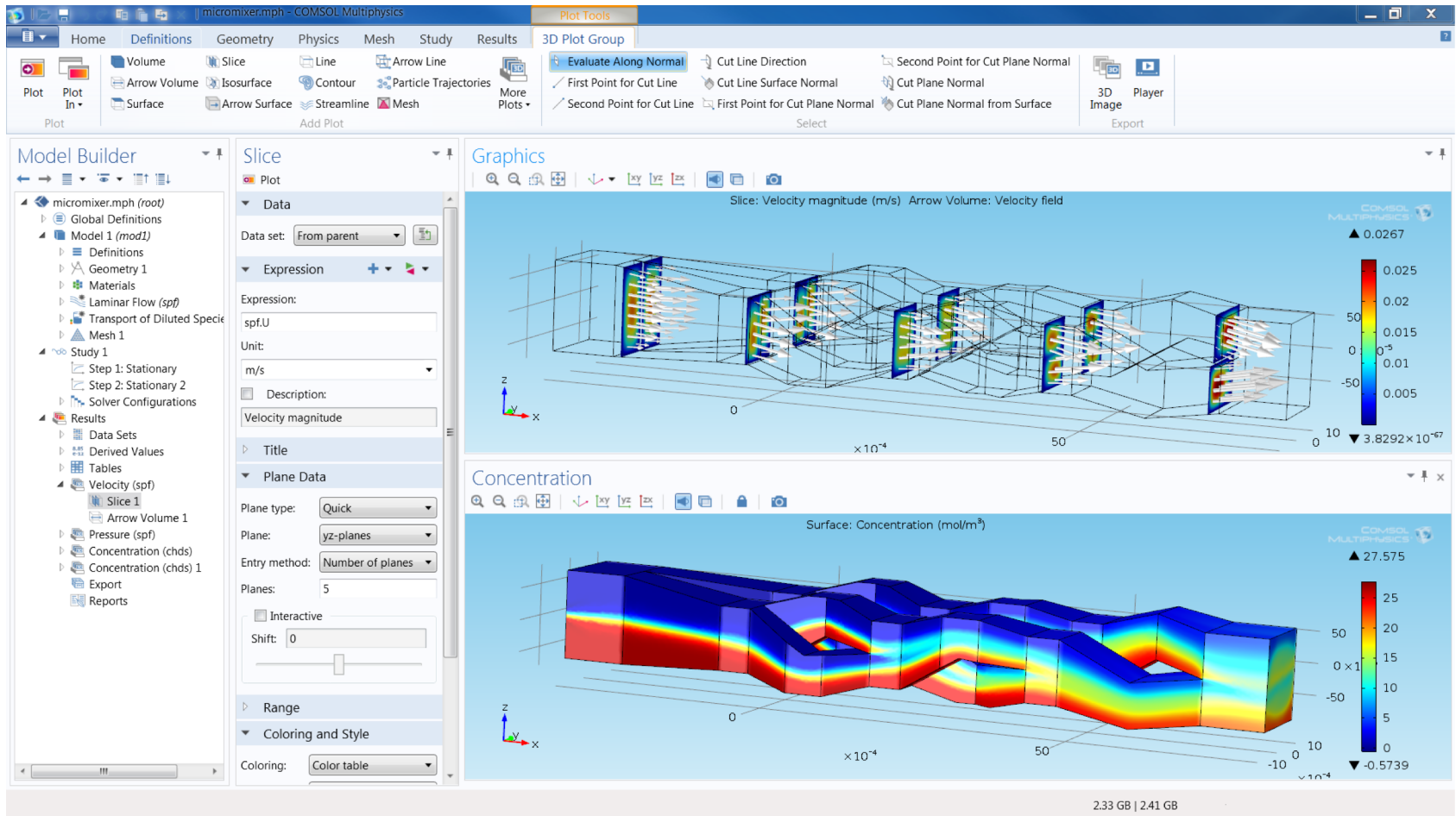
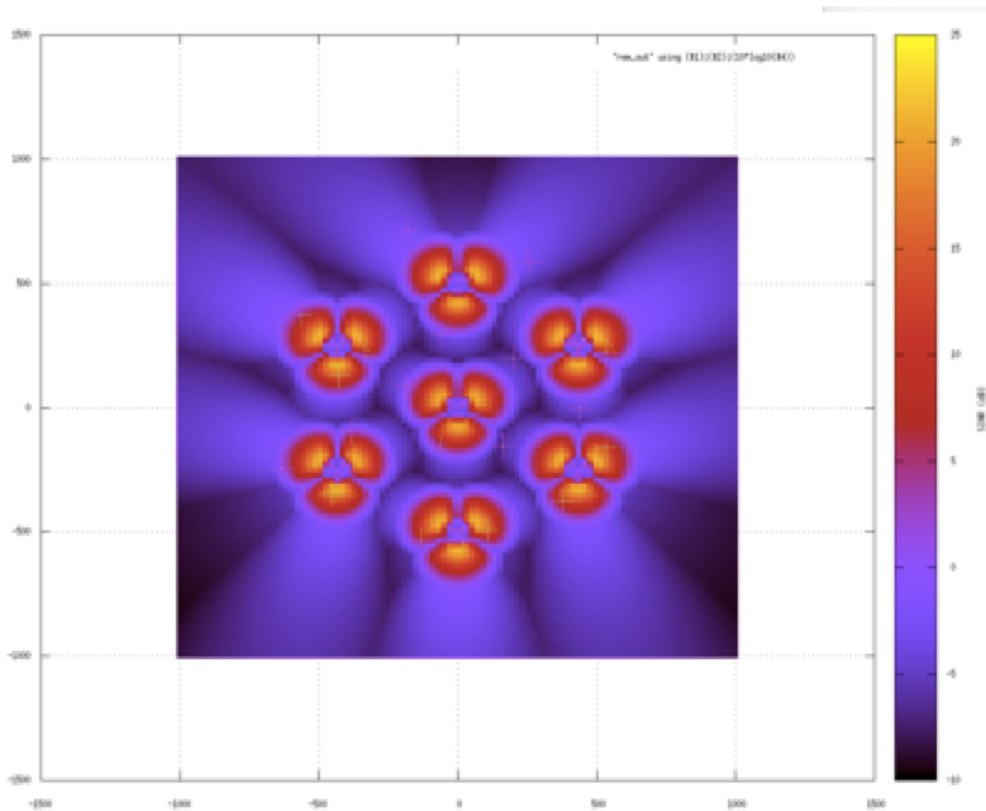


Figure source: <https://www.comsol.com/comsol-multiphysics>

ns-3 uses outside programs for graphics



gnuplot

LTE radio environment
map (REM)

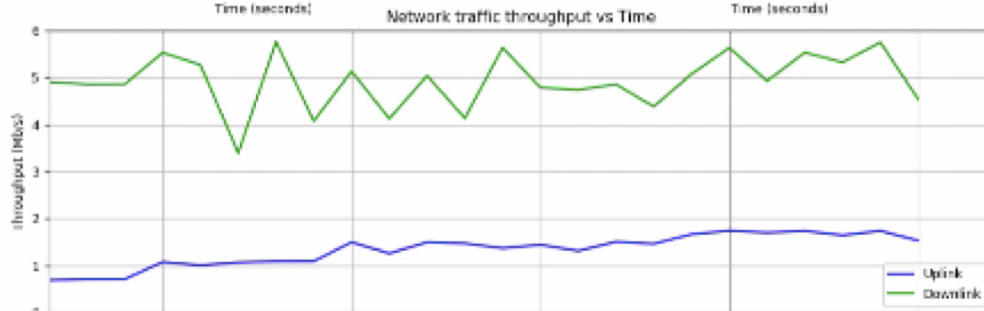
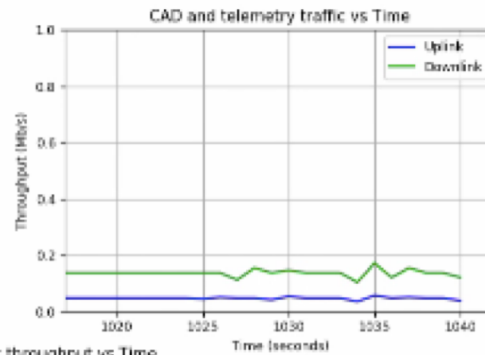
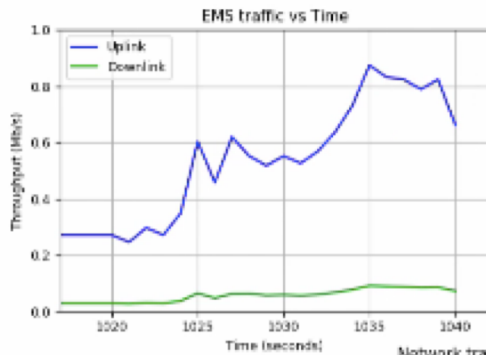
Will experiment with
this on Tuesday

ns-3 users typically write scripts to plot

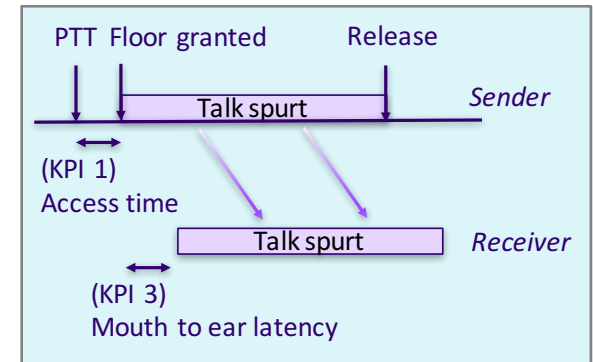
Animated

EmsVideo_1_Server 942.36392953 RX 1012 1061 U 2395
 EmsVideo_1_Client 942.37317727 TX 1012 1061 U 2398
 EmsVideo_1_Client 942.377 RX 64 113 U 2397
 WebBrowsingGraphics_0_Server 942.38092876 TX 1024 1073 U 2399
 WebBrowsingGraphics_0_Client 942.394 RX 1024 1073 U 2399
 AvlAssetPerimeter_1_Server 942.42492988 RX 1408 1457 U 2401

Throughput vs. time for incident scenario



Used to measure KPIs



Visualization

- No preferred visualizer for ns-3
- Two tools have been developed over the years, with some scope limitations
 - Pyviz
 - FlowMonitor (statistics with Pyviz linkage)
 - NetAnim (George Riley and John Abraham)
- Support is lagging for these tools (help wanted)

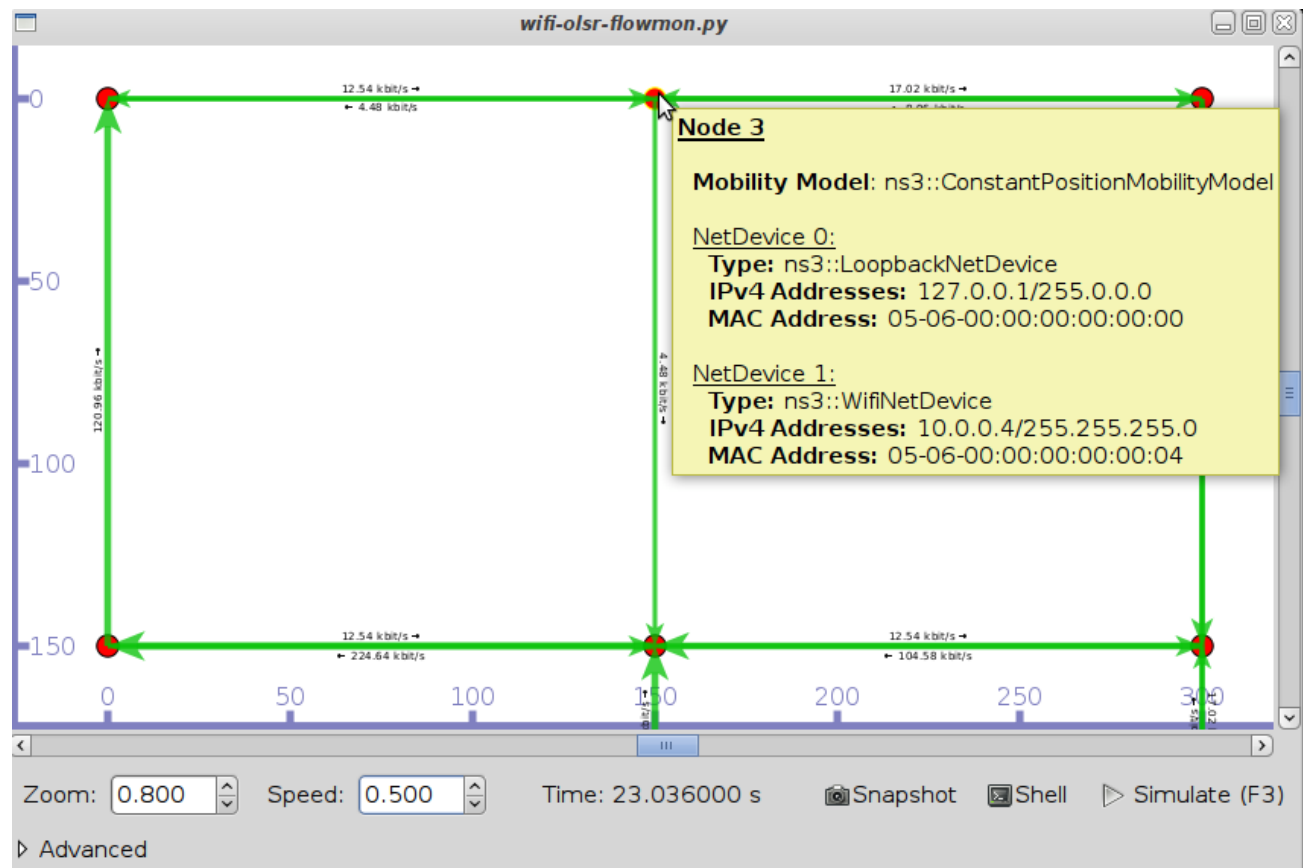
PyViz overview

- Developed by Gustavo Carneiro
- Live simulation visualizer (no trace files)
- Useful for debugging
 - mobility model behavior
 - where are packets being dropped?
- Built-in interactive Python console to debug the state of running objects
- Works with Python and C++ programs

Pyviz and FlowMonitor

- Example screenshot from:

```
./waf --run src/flow-monitor/examples/wifi-olsr-flowmon.py  
--vis
```



Enabling PyViz in your simulations

- Make sure PyViz is enabled in the build

```
Sqlite stats data output      : not enabled (library 'sqlite3' not found)
Tap Bridge                    : enabled
PyViz visualizer              : enabled
Use sudo to set suid bit      : not enabled (option --enable-sudo not selected)
```

- If program supports CommandLine parsing, pass the option

```
--SimulatorImplementationType=
ns3::VisualSimulatorImpl
```

- Alternatively, pass the "--vis" option

FlowMonitor

- Network monitoring framework found in `src/flow-monitor/`
- Goals:
 - detect all flows passing through network
 - stores metrics for analysis such as bitrates, duration, delays, packet sizes, packet loss ratios

Plan to discuss more on Tuesday

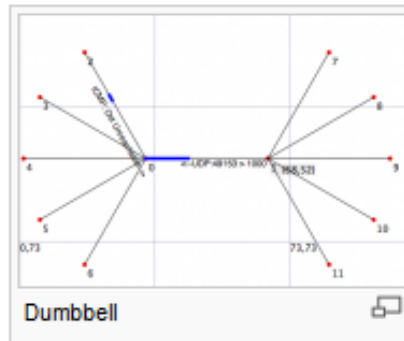
G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.

NetAnim

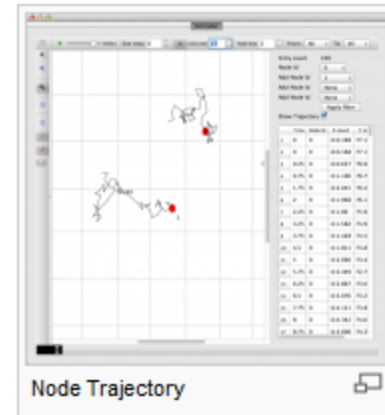
- "NetAnim" by George Riley and John Abraham

No	Time	Event Name	From Node	To Node	Details
1	2.5e-05	0	5	5	WIR MGT_BEACON From DS: 0 to DS: 0 DA: 00:00:00:00:00:00
2	2.3e-05	0	6	6	WIR MGT_BEACON From DS: 0 to DS: 0 DA: 00:00:00:00:00:00
3	2.5e-05	0	7	7	WIR MGT_BEACON From DS: 0 to DS: 0 DA: 00:00:00:00:00:00
4	0.000167003	5	6	WIR MGT_ASSOCIATION_REQUEST From DS: 0 to DS: 0	
5	0.000167003	5	7	WIR MGT_ASSOCIATION_REQUEST From DS: 0 to DS: 0	
6	0.000167003	5	0	WIR MGT_ASSOCIATION_REQUEST From DS: 0 to DS: 0	
7	0.000179066	0	5	WIR CTL_ACK RA:00:00:00:00:00:00	
8	0.000179066	0	6	WIR CTL_ACK RA:00:00:00:00:00:00	
9	0.000179066	0	7	WIR CTL_ACK RA:00:00:00:00:00:00	
10	0.000492183	6	5	WIR MGT_ASSOCIATION_REQUEST From DS: 0 to DS: 0	
11	0.000492183	6	0	WIR MGT_ASSOCIATION_REQUEST From DS: 0 to DS: 0	
12	0.00051414	0	5	WIR CTL_ACK RA:00:00:00:00:00:00	
13	0.00051414	0	6	WIR CTL_ACK RA:00:00:00:00:00:00	
14	0.00051414	0	7	WIR CTL_ACK RA:00:00:00:00:00:00	

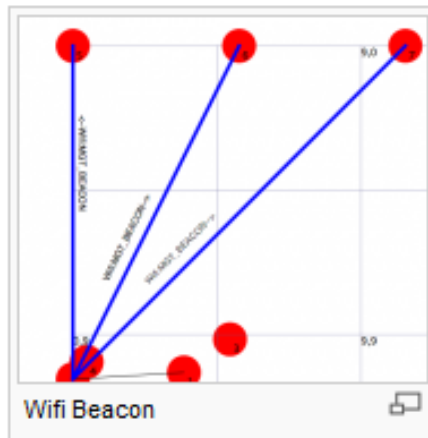
Packet Statistics



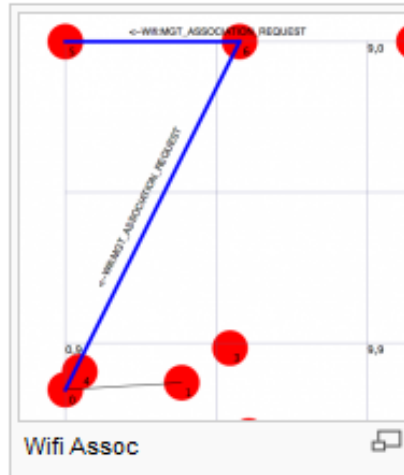
Dumbbell



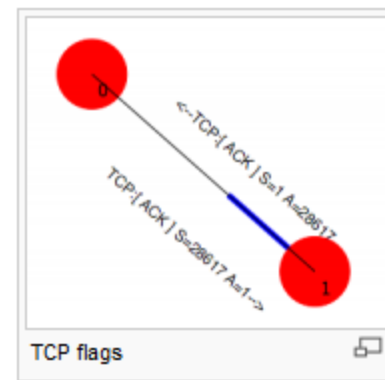
Node Trajectory



Wifi Beacon



Wifi Assoc



TCP flags

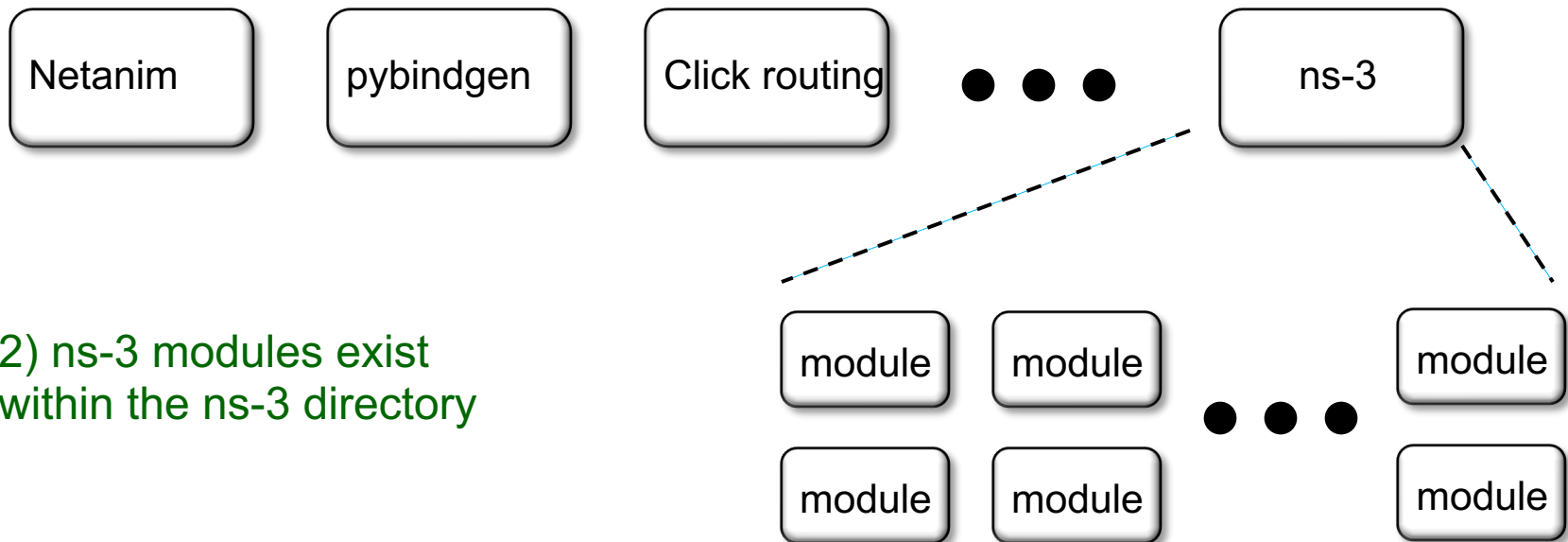
NetAnim key features

- Animate packets over wired-links and wireless-links
 - limited support for LTE traces
- Packet timeline with regex filter on packet meta-data.
- Node position statistics with node trajectory plotting (path of a mobile node).
- Print brief packet-meta data on packets

Software organization

- Two levels of ns-3 software and libraries

1) Several supporting libraries, not system-installed, can be in parallel to ns-3



2) ns-3 modules exist within the ns-3 directory

Typical module source code organization

model/

examples/

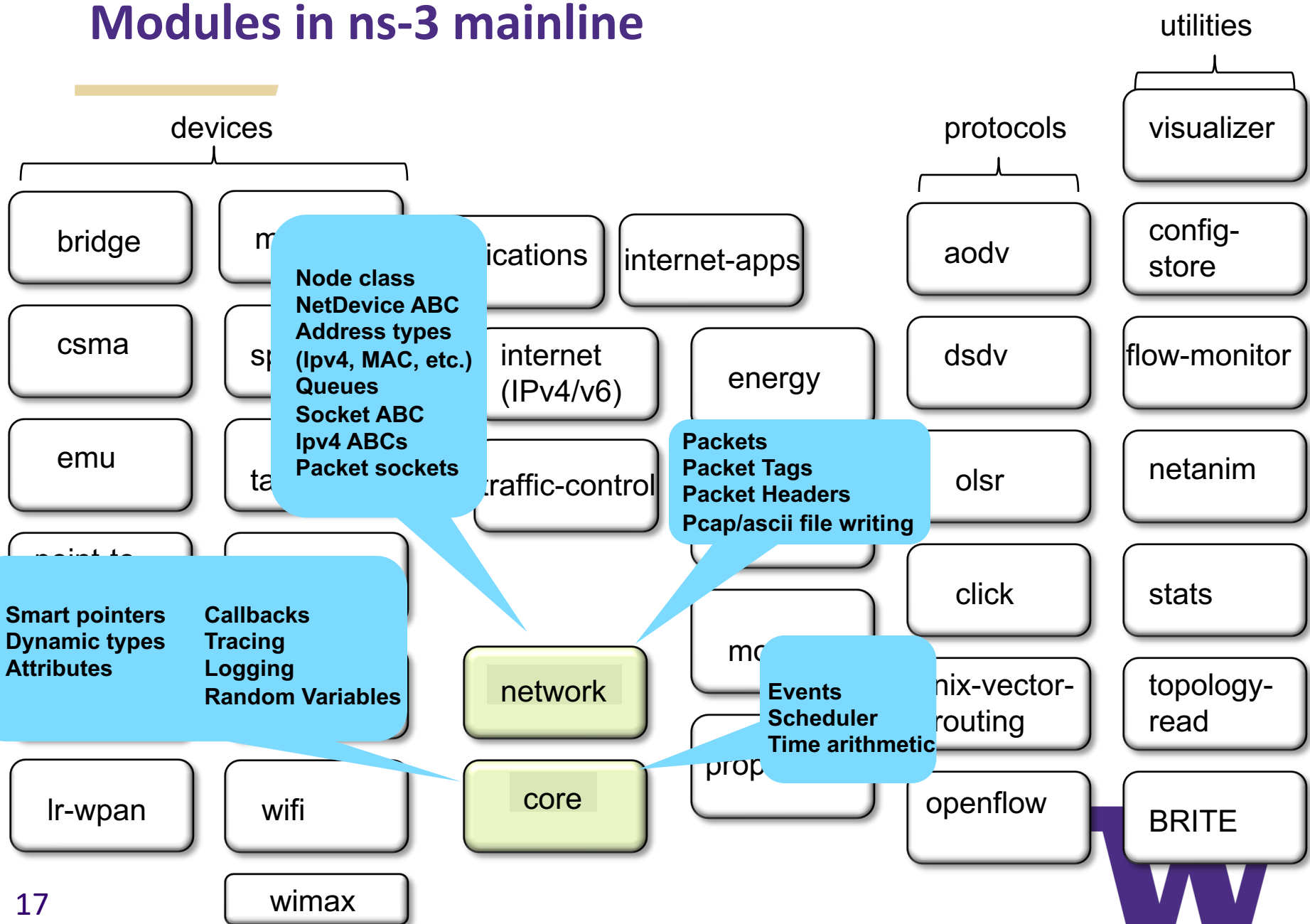
test/

bindings/

doc/

wscript/

Modules in ns-3 mainline

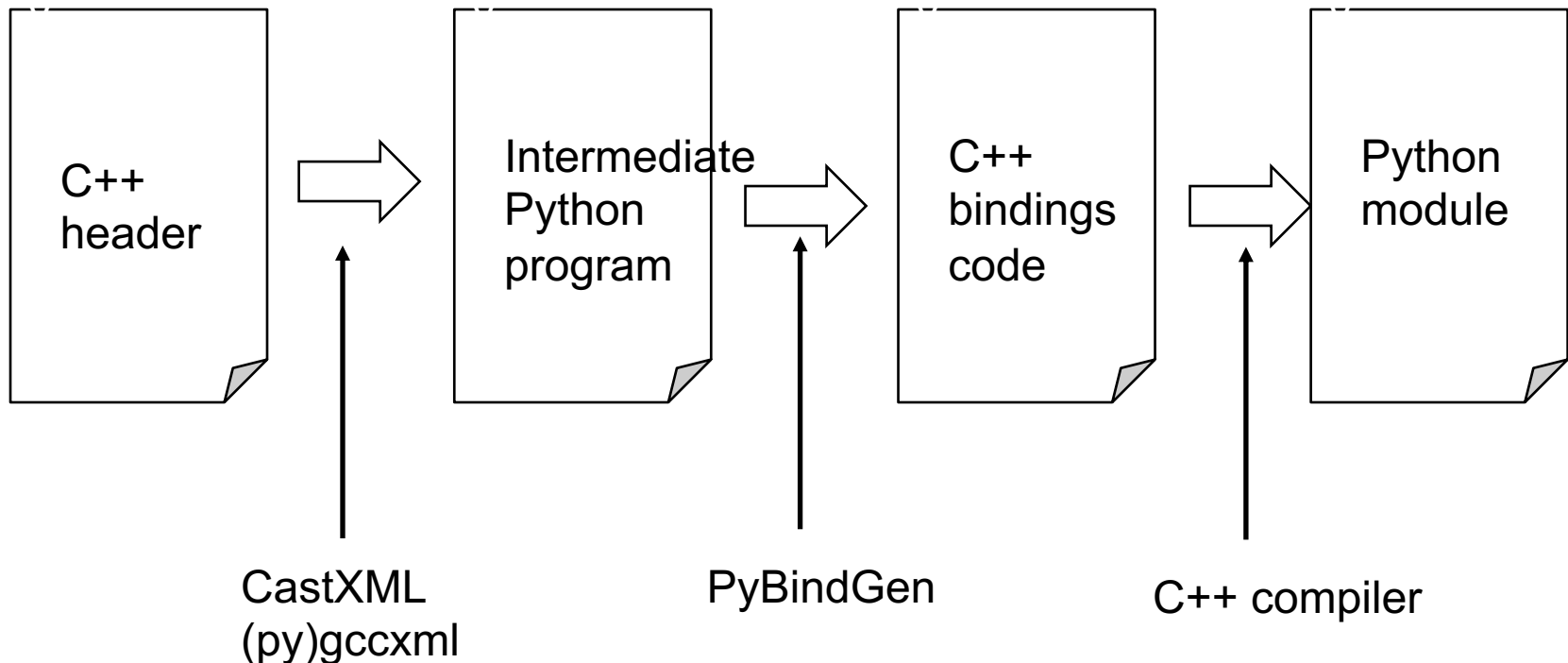


ns-3 programs

- ns-3 programs are C++ executables that link the needed shared libraries
 - or Python programs that import the needed modules
- The ns-3 build tool, called 'waf', can be used to run programs
- waf will place headers, object files, libraries, and executables in a 'build' directory

Python bindings

- ns-3 uses a program called PyBindGen to generate Python bindings for all libraries



Python bindings status

- API scanning for Python used to use a tool called gccxml
- ns-3 has moved to the successor, CastXML
 - requires a development installation of clang
- Automated testing currently only for Linux 64-bit systems
 - MacOS API scanning is not tested

waf operation

- This slide is a placeholder to demonstrate Waf operation
 - `'waf build'` will compile and link source code into executables
 - `'waf --run'` will run an executable in a special shell that knows the path to ns-3 libraries
 - New option: `'waf --run-no-build'` will skip the build step

waf configuration

- Key waf configuration examples

```
./waf configure
  --enable-examples
  --enable-tests
  --disable-python
  --enable-modules
```

- Whenever build scripts change, need to reconfigure

Demo: `./waf --help`
`./waf configure --enable-examples --enable-tests --enable-modules='core'`

Look at: `build/c4che/_cache.py`

wscript example

```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-

def build(bld):
    obj = bld.create_ns3_module('csma', ['network', 'applications'])
    obj.source = [
        'model/backoff.cc',
        'model/csma-net-device.cc',
        'model/csma-channel.cc',
        'helper/csma-helper.cc',
    ]
    headers = bld.new_task_gen(features=['ns3header'])
    headers.module = 'csma'
    headers.source = [
        'model/backoff.h',
        'model/csma-net-device.h',
        'model/csma-channel.h',
        'helper/csma-helper.h',
    ]

    if bld.env['ENABLE_EXAMPLES']:
        bld.add_subdirs('examples')

    bld.ns3_python_bindings()
```

waf build

- Once project is configured, can build via `./waf build` or `./waf`
- waf will build in parallel on multiple cores
- waf displays modules built at end of build

Demo: `./waf build`

Look at: `build/` libraries and executables

Running programs

- `./waf shell` provides a special shell for running programs
 - Sets key environment variables

```
./waf --run sample-simulator
```

```
./waf --pyrun src/core/examples/sample-simulator.py
```

Build variations

- Configuring a build type is done at waf configuration time
- debug build (default): all asserts and debugging code enabled
`./waf -d debug configure`
- **optimized**
`./waf -d optimized configure`
- **static libraries**
`./waf --enable-static configure`

Controlling the modular build

- One way to disable modules:
 - `./waf configure --enable-modules='a','b','c'`
- The `.ns3rc` file (found in `utils/` directory) can be used to control the modules built
- Precedence in controlling build
 - 1) command line arguments
 - 2) `.ns3rc` in ns-3 top level directory
 - 3) `.ns3rc` in user's home directory

Demo how `.ns3rc` works

Building without wscript

- The scratch/ directory can be used to build programs without wscripts


Demo how programs can be built without wscripts

Integrating other tools and libraries

Other libraries

- more sophisticated scenarios and models typically leverage other libraries
- ns-3 main distribution uses optional libraries (libxml2, gsl, mysql) but care is taken to avoid strict build dependencies
 - The Waf wscripts can be consulted as examples
 - example: `sqlite3` in `src/stats/wscript`
- the 'bake' tool (described later) helps to manage library dependencies
- users are free to write their own Makefiles or wscripts to do something special

CORE emulator




Networks and Communication Systems Branch

Focus Areas | Projects | Products | Organization


/ NRL / ITD / NCS / Common Open Research Emulator (CORE)

Common Open Research Emulator (CORE)

The Common Open Research Emulator (CORE) is a tool for emulating networks on one or more machines. You can connect these emulated networks to live networks. CORE consists of a GUI for drawing topologies of lightweight virtual machines, and Python modules for scripting network emulation.



NCS Home
Focus Areas
Projects
Products
Organization



mininet emulator

The screenshot shows the GitHub repository page for `mininet/mininet`. The repository is public and has 468 stars and 204 forks. The page title is "Link modeling using ns 3". The main content area is titled "Contents" and lists the following items:

- Introduction
 - ns-3 emulation features
 - Link simulation with ns-3
- Details
 - How to achieve communication of ns-3 process with TAP interfaces in distinct namespaces?
 - Architecture: single ns-3 thread or multiple processes?
- Code
 - Mininet
 - ns-3 patches

On the right side, there is a sidebar with a table of contents for the repository:

- Mininet
- Get Started
- Sample Workflow
- Walkthrough
- Overview

Below this, there is another list of links:

- Download
- Documentation
- Videos
- Source Code
- Apps
- FAQ
- Wiki
- Teaching
- Papers
- GSoC 2013

Co-simulation frameworks have emerged

- PNNL's FNCS framework integrates ns-3 with transmission and distribution simulators

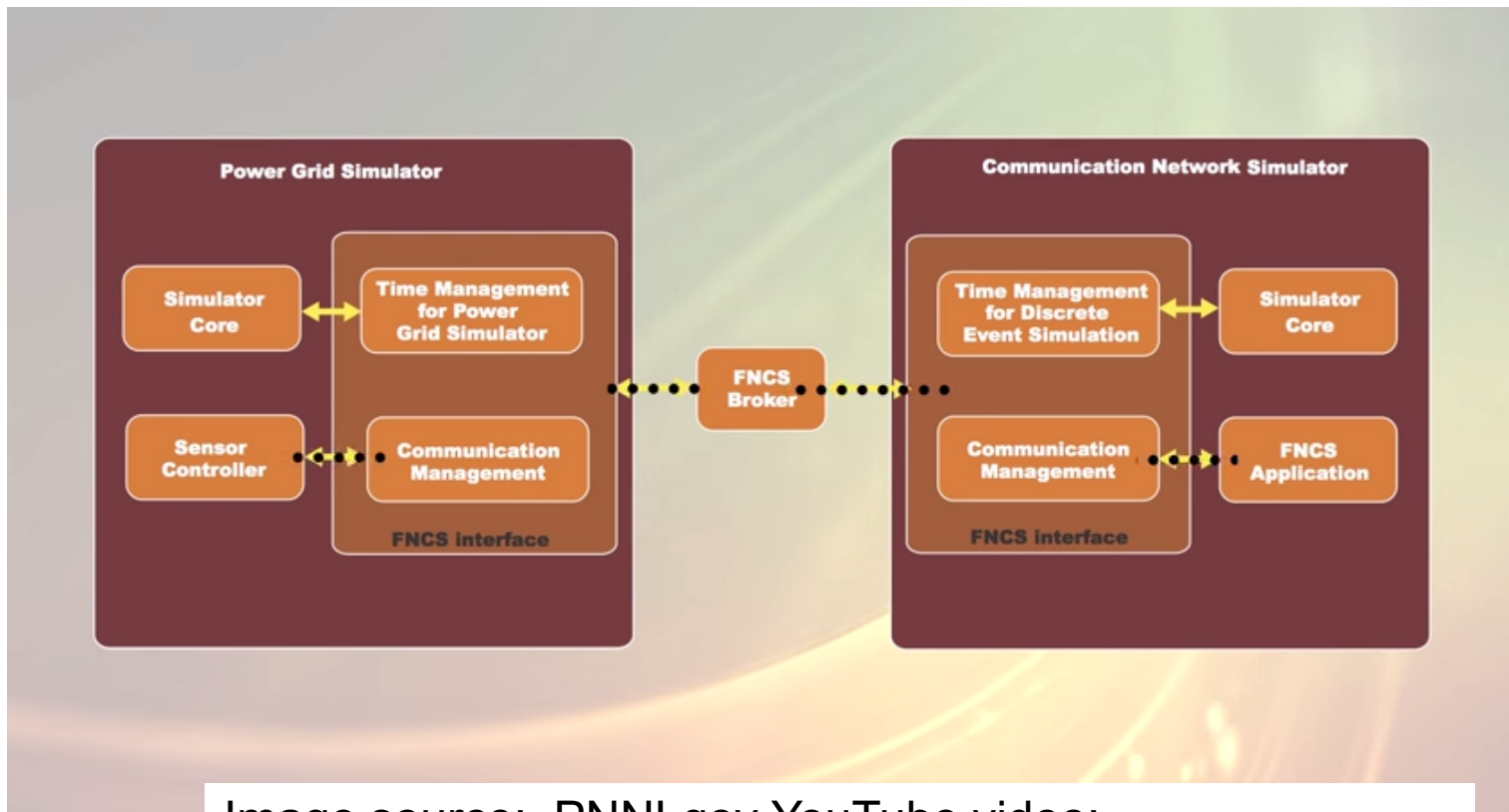
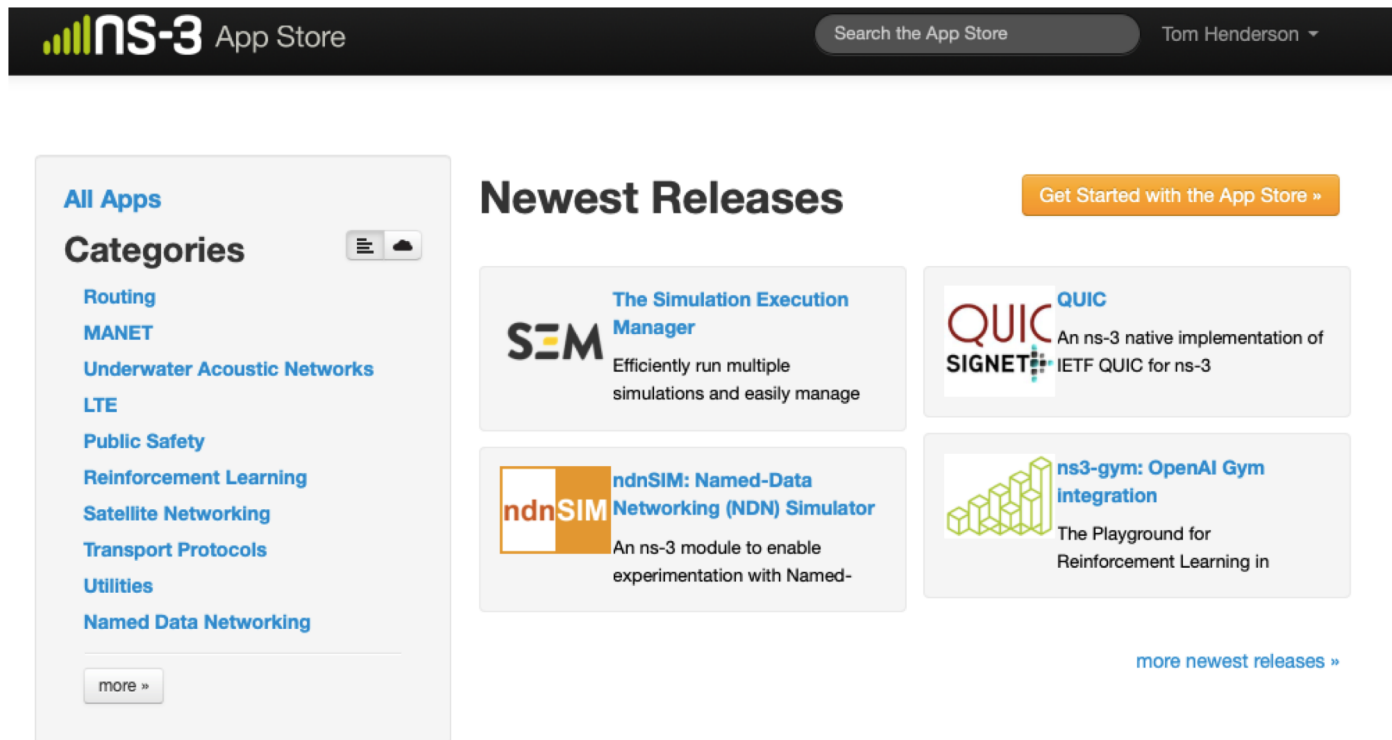


Image source: PNNLgov YouTube video:
Introducing FNCS: Framework for Network Co-Simulation

ns-3 App Store

- Project is migrating away from a centralized repository to a modular system called the 'ns-3 App Store'

– <https://apps.nsnam.org>



The screenshot shows the ns-3 App Store interface. At the top, there is a dark header with the ns-3 logo and 'App Store' text on the left, a search bar with the text 'Search the App Store' in the center, and the user name 'Tom Henderson' with a dropdown arrow on the right. Below the header, the main content area is divided into several sections. On the left, there is a sidebar titled 'All Apps' with a sub-section 'Categories' containing a list of categories: Routing, MANET, Underwater Acoustic Networks, LTE, Public Safety, Reinforcement Learning, Satellite Networking, Transport Protocols, Utilities, and Named Data Networking. A 'more »' button is at the bottom of the categories list. The main content area is titled 'Newest Releases' and features a 'Get Started with the App Store »' button in the top right. Below this, there are four app cards. The first card is for 'SEM: The Simulation Execution Manager', described as 'Efficiently run multiple simulations and easily manage'. The second card is for 'QUIC: An ns-3 native implementation of IETF QUIC for ns-3'. The third card is for 'ndnSIM: Named-Data Networking (NDN) Simulator', described as 'An ns-3 module to enable experimentation with Named-'. The fourth card is for 'ns3-gym: OpenAI Gym integration', described as 'The Playground for Reinforcement Learning in'. A 'more newest releases »' link is at the bottom right of the main content area.

Documentation overview

- Placeholder slide: online browsing of
 - Doxygen
 - ns-3 manual, model library, tutorial
 - wiki
 - command-line help