ns-3 Training

Emulation overview

MNM Workshop
May 2015

Outline

- Emulation modes
  - Tap Bridge
  - FdNetDevice
- Direct Code Execution (DCE)
  - Applications
  - Linux Kernel
  - DCE Cradle

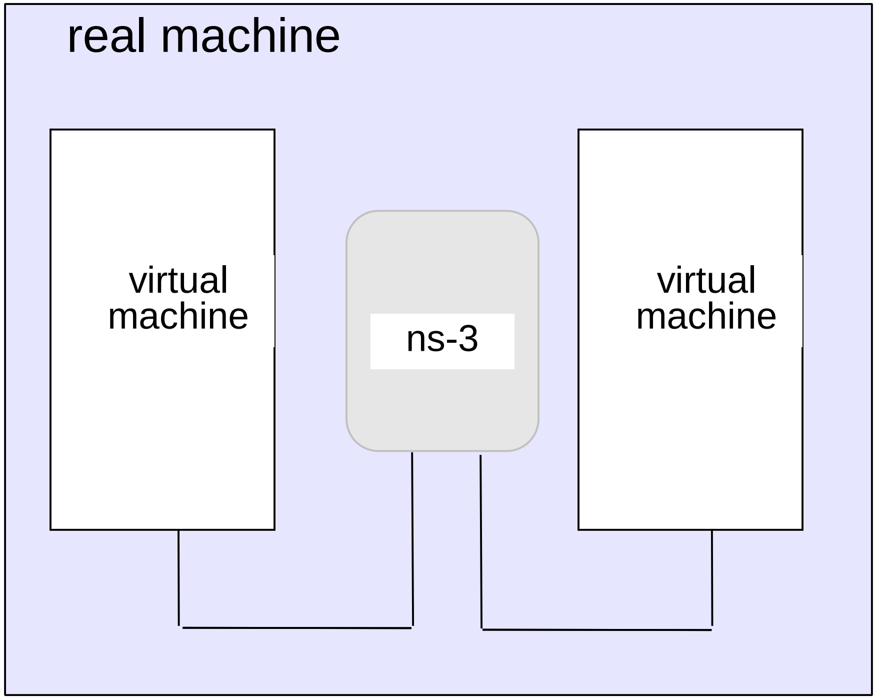
Emulation support

- Support moving between simulation and testbeds or live systems
- A real-time scheduler, and support for two modes of emulation
- Linux is only operating system supported
- Must run simulator in real time

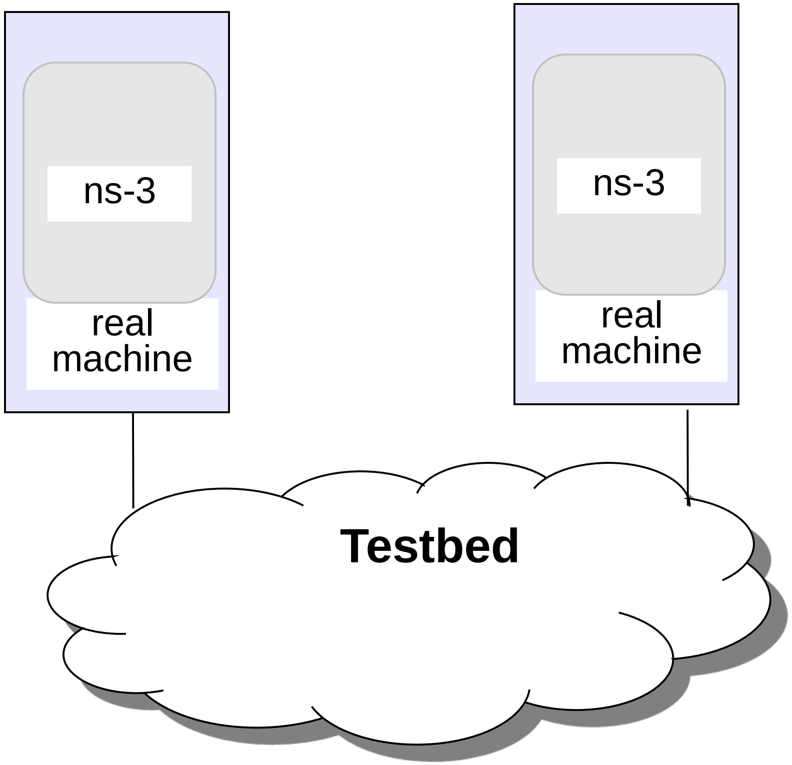
  -
    ```
    GlobalValue::Bind (“SimulatorImplementationType”,
    StringValue (“ns3::RealTimeSimulatorImpl”));
    ```
- Must enable checksum calculations across models

  -
    ```
    GlobalValue::Bind (“ChecksumEnabled”, BooleanValue
    (true));
    ```
- Must run as root

ns-3 emulation modes



1) ns-3 interconnects real or virtual machines

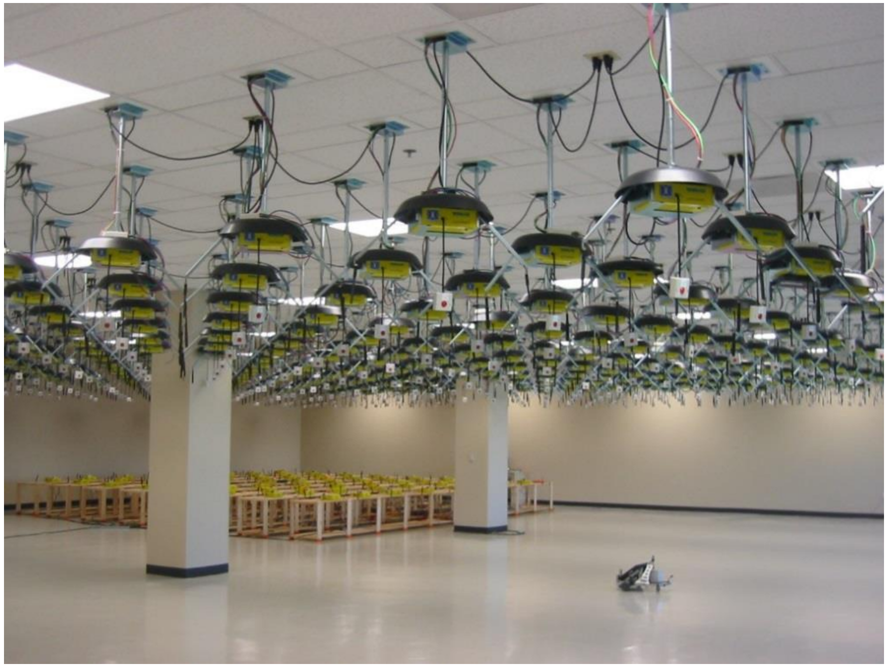


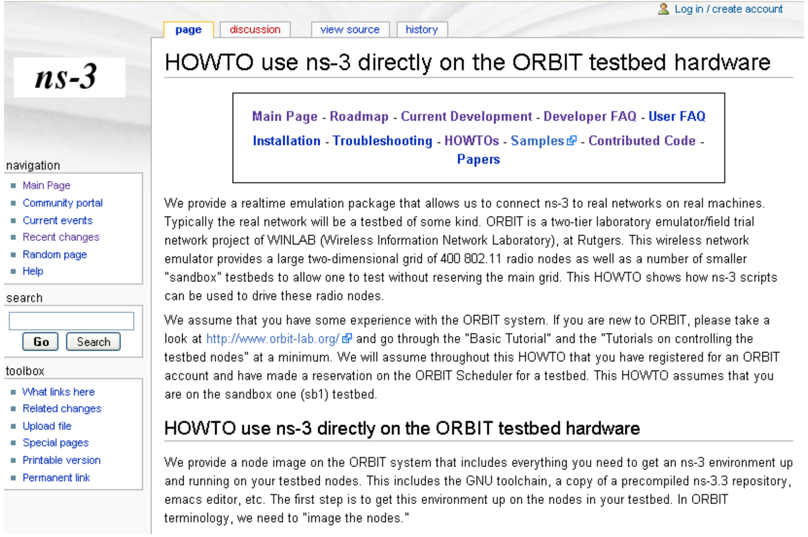
2) testbeds interconnect ns-3 stacks

Various hybrids of the above are possible

Example use case: testbeds

- Support for use of Rutgers WINLAB ORBIT radio grid





HOWTO use ns-3 directly on the ORBIT testbed hardware

Main Page - Roadmap - Current Development - Developer FAQ - User FAQ
Installation - Troubleshooting - HOWTOs - Samples - Contributed Code - Papers

We provide a realtime emulation package that allows us to connect ns-3 to real networks on real machines. Typically the real network will be a testbed of some kind. ORBIT is a two-tier laboratory emulator/field trial network project of WINLAB (Wireless Information Network Laboratory), at Rutgers. This wireless network emulator provides a large two-dimensional grid of 400 802.11 radio nodes as well as a number of smaller "sandbox" testbeds to allow one to test without reserving the main grid. This HOWTO shows how ns-3 scripts can be used to drive these radio nodes.

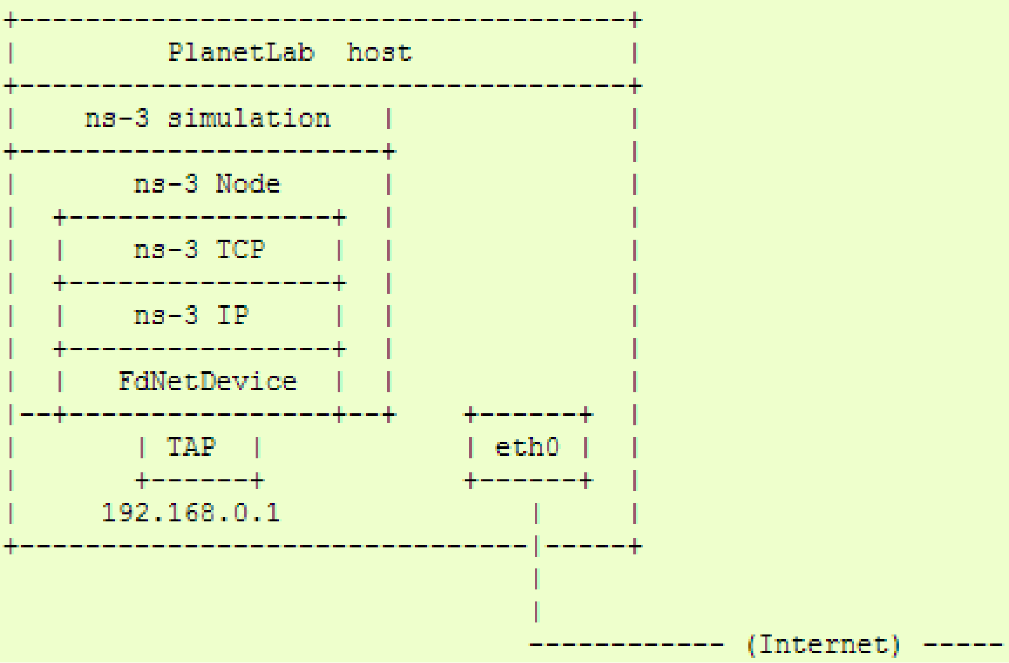
We assume that you have some experience with the ORBIT system. If you are new to ORBIT, please take a look at http://www.orbit-lab.org/ and go through the "Basic Tutorial" and the "Tutorials on controlling the testbed nodes" at a minimum. We will assume throughout this HOWTO that you have registered for an ORBIT account and have made a reservation on the ORBIT Scheduler for a testbed. This HOWTO assumes that you are on the sandbox one (sb1) testbed.

HOWTO use ns-3 directly on the ORBIT testbed hardware

We provide a node image on the ORBIT system that includes everything you need to get an ns-3 environment up and running on your testbed nodes. This includes the GNU toolchain, a copy of a precompiled ns-3.3 repository, emacs editor, etc. The first step is to get this environment up on the nodes in your testbed. In ORBIT terminology, we need to "image the nodes."

Example use case: PlanetLab

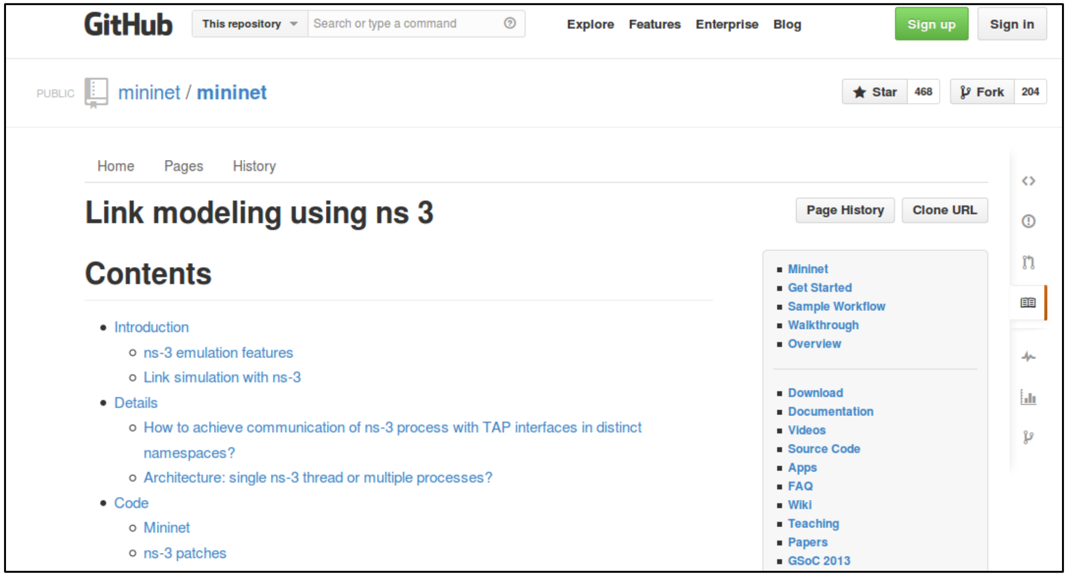
- The PlanetLabFdNetDeviceHelper creates TAP devices on PlanetLab nodes using specific PlanetLab mechanisms (i.e. the vsys system), and associates the TAP device to a FdNetDevice in ns-3.



```
+-------------------------------------+
|         PlanetLab  host             |
+-------------------------------------+
|    ns-3 simulation   |              |
+----------------------+              |
|       ns-3 Node      |              |
|  +----------------+  |              |
|  |    ns-3 TCP    |  |              |
|  +----------------+  |              |
|  |    ns-3 IP     |  |              |
|  +----------------+  |              |
|  |   FdNetDevice  |  |              |
|--+----------------+--+    +------+  |
|       | TAP  |            | eth0 |  |
|       +-------+           +------+  |
|     192.168.0.1               |     |
+-------------------------------|-----+
                                |
                                |
                                ------------ (Internet) -----
```

Example use case: mininet

- Mininet is popular in the Software-Defined Networking (SDN) community
- Mininet uses "TapBridge" integration
- https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3



Link modeling using ns 3

Contents

- 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

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

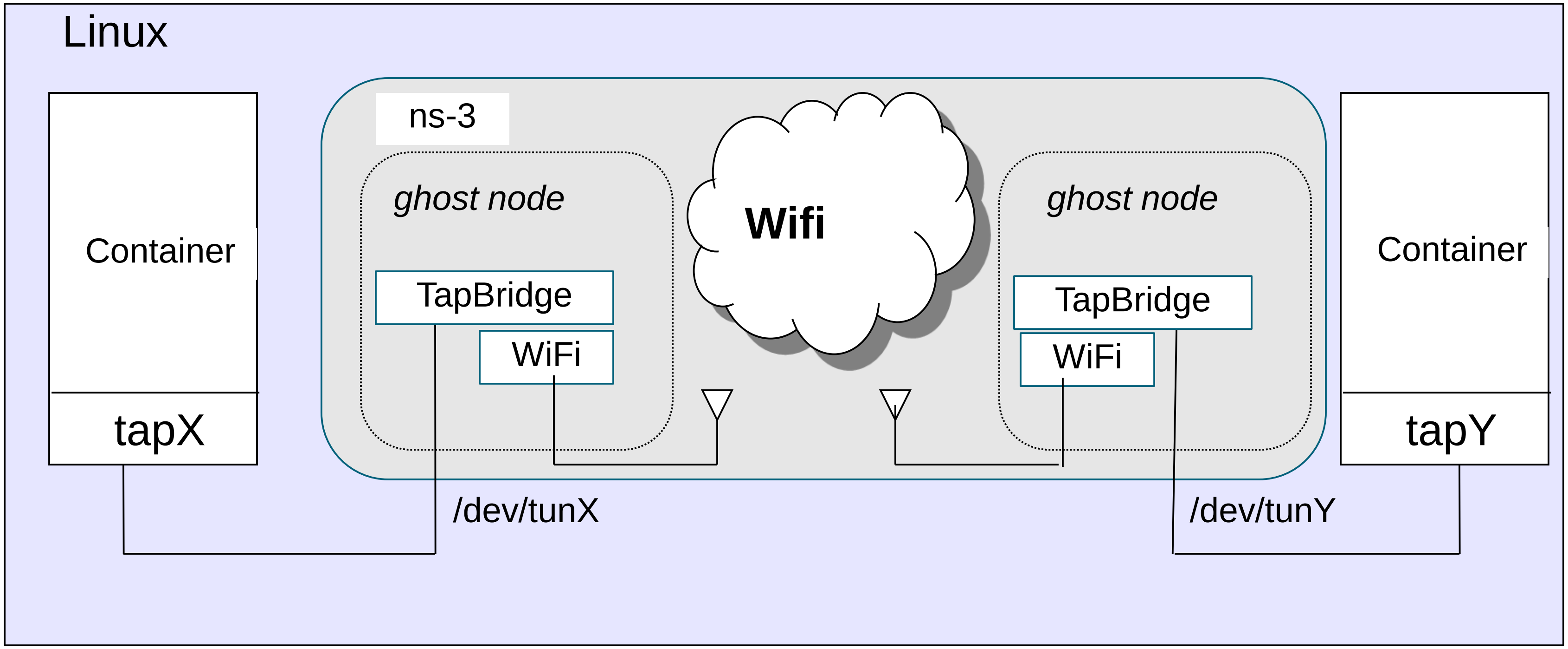
- Download
- Documentation
- Videos
- Source Code
- Apps
- FAQ
- Wiki
- Teaching
- Papers
- GSoC 2013

Emulation Devices

Device models

- File Descriptor Net Device (FdNetDevice)
  - read and write traffic using a file descriptor provided by the user
  - this file descriptor can be associated to a TAP device, to a raw socket, to a user space process generating/consuming traffic, etc.
- Tap Bridge
  - Integrate Tun/Tap devices with ns-3 devices

“TapBridge": netns and ns-3 integration

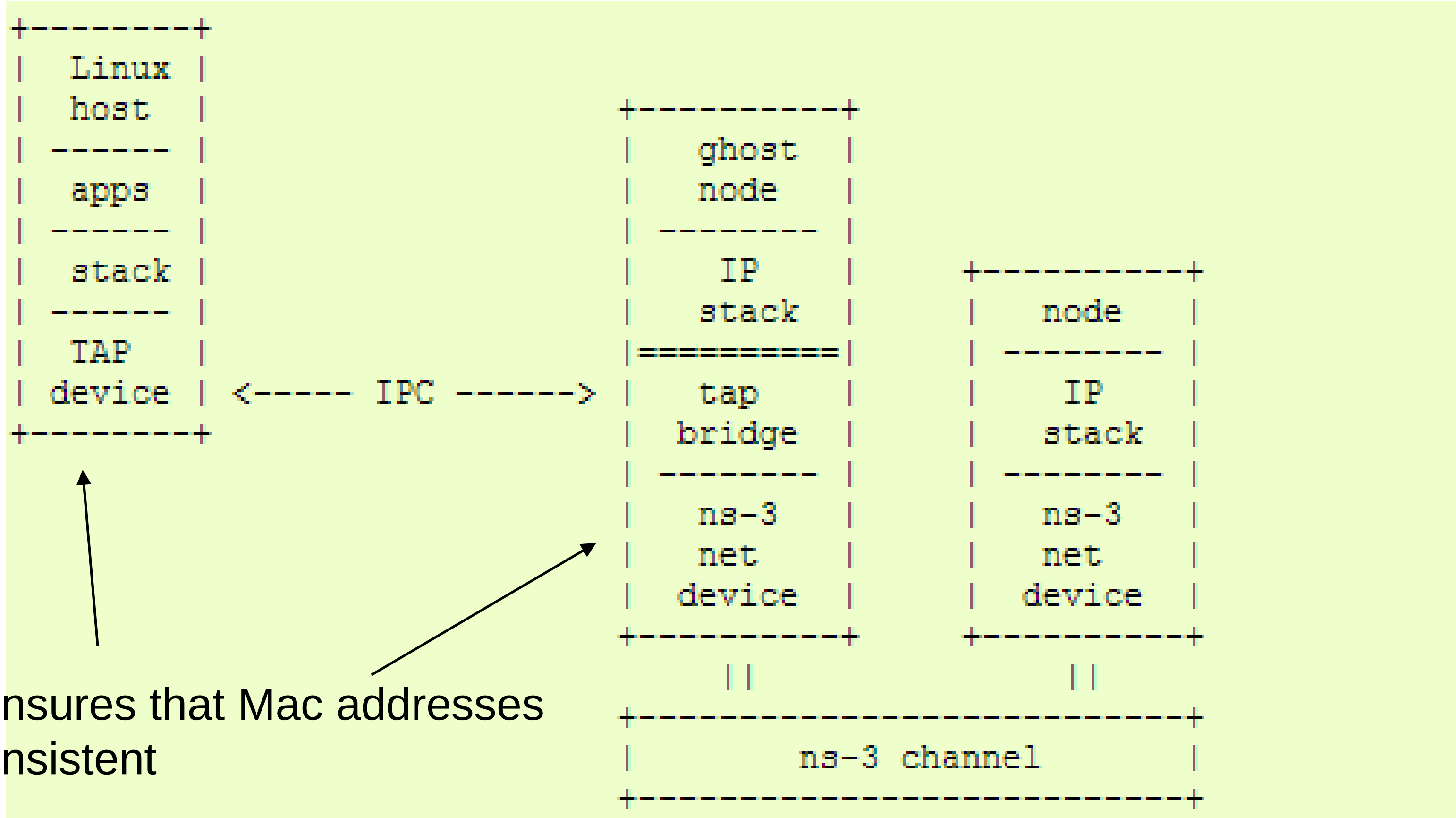


Tap device pushed into namespaces; no bridging needed

TapBridge modes

- ConfigureLocal (default mode)
  - ns-3 configures the tap device
  - useful for host to ns-3 interaction
- UseLocal
  - user has responsibility for device creation
  - ns-3 informed of device using “DeviceName” attribute
- UseBridge
  - TapDevice connected to existing Linux bridge

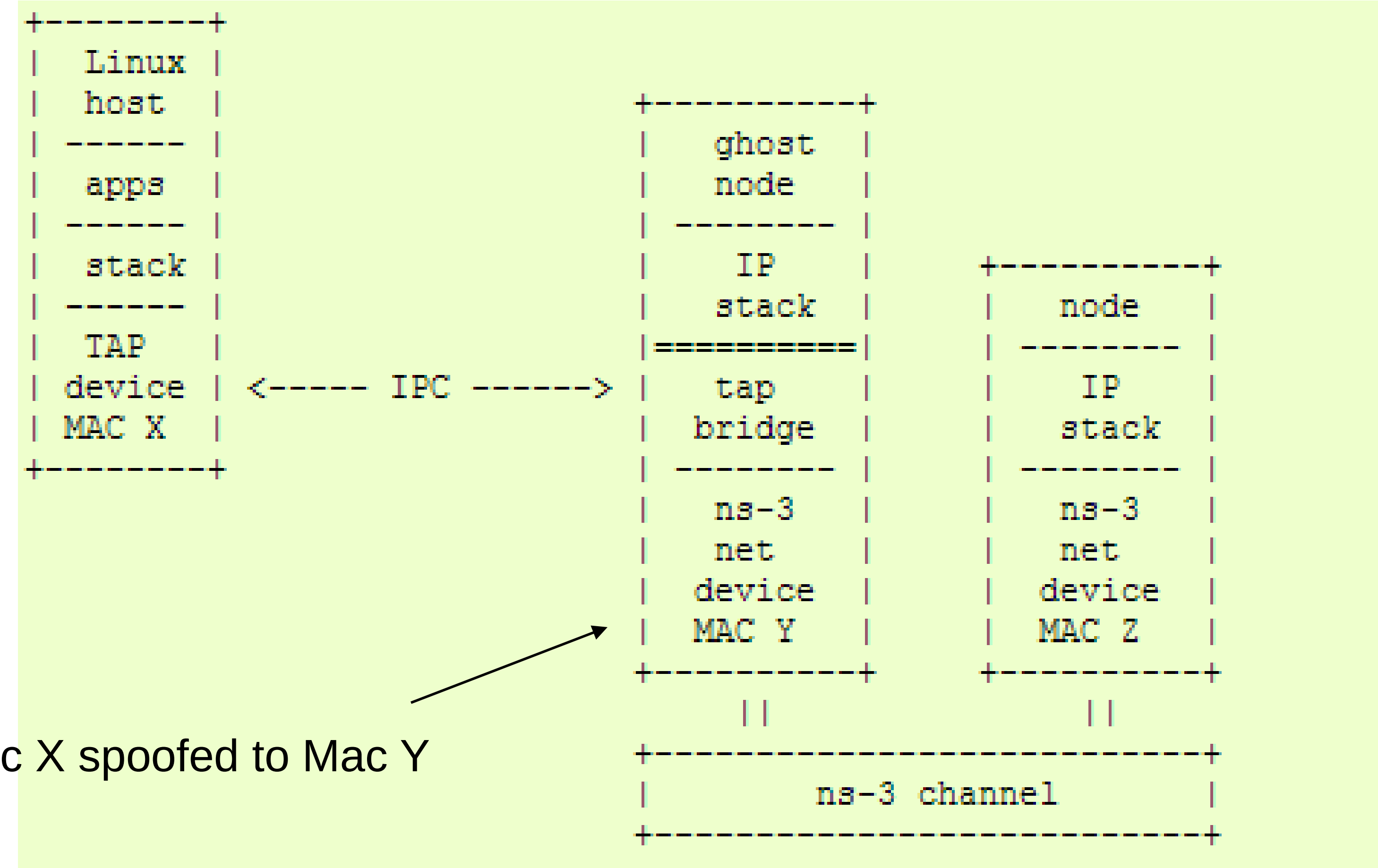
ConfigureLocal



```
+---------+
|  Linux  |
|  host   |                     +-----------+
| ------- |                     |   ghost   |
|  apps   |                     |   node    |
| ------- |                     | --------- |
|  stack  |                     |    IP     |     +-----------+
| ------- |                     |   stack   |     |   node    |
|  TAP    |                     |===========|     | --------- |
| device  | <----- IPC ------>  |   tap     |     |    IP     |
+---------+                     |  bridge   |     |   stack   |
                                | --------- |     | --------- |
                                |   ns-3    |     |   ns-3    |
                                |   net     |     |   net     |
                                |  device   |     |  device   |
                                +-----------+     +-----------+
                                      ||                ||
                                +-----------------------------+
                                |          ns-3 channel       |
                                +-----------------------------+
```

ns-3 ensures that Mac addresses are consistent

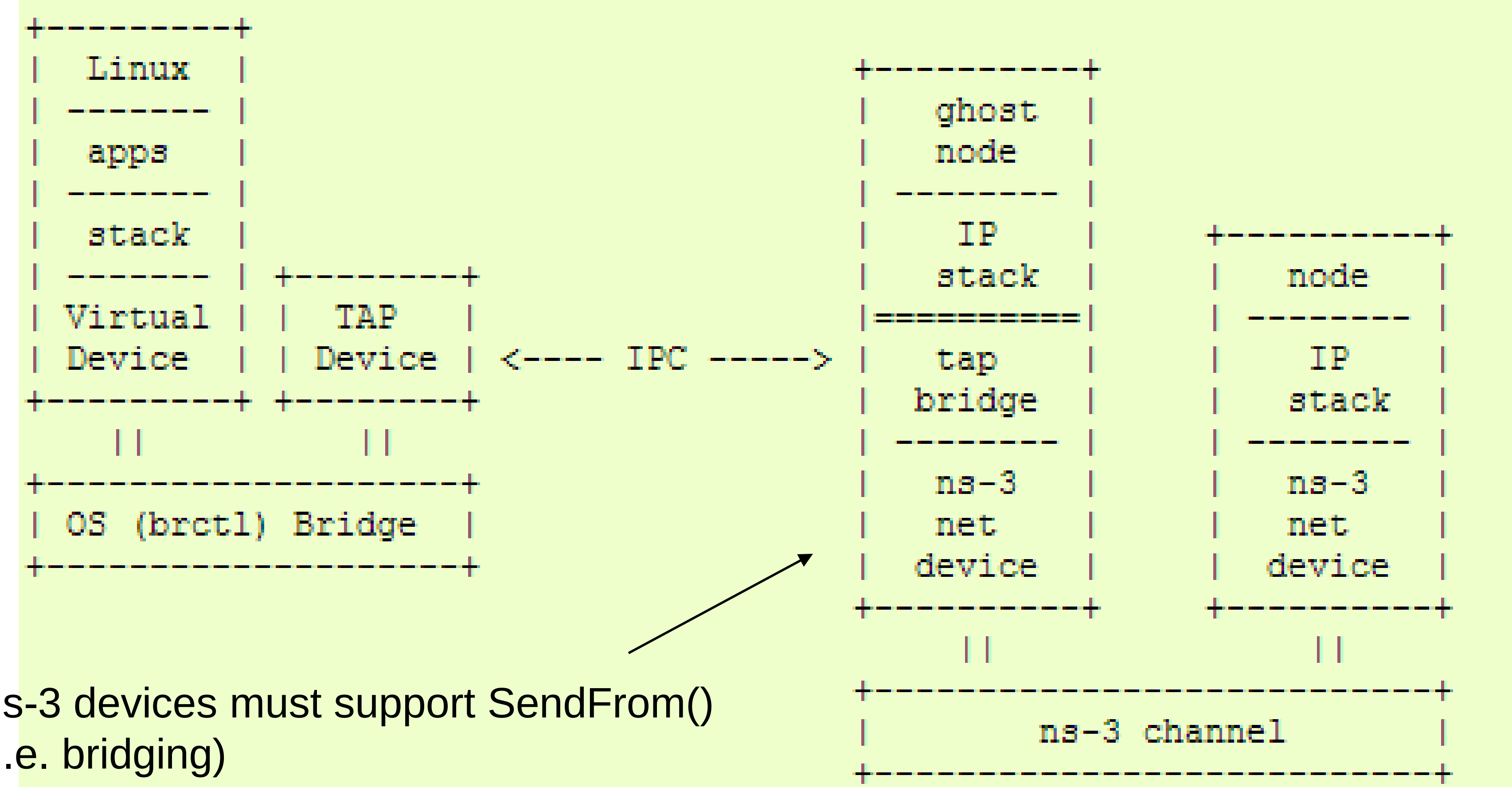
UseLocal



```
+---------+
|  Linux  |
|  host   |                      +-----------+
| ------- |                      |   ghost   |
|  apps   |                      |   node    |
| ------- |                      | --------- |
|  stack  |                      |    IP     |      +-----------+
| ------- |                      |   stack   |      |   node    |
|  TAP    |                      |===========|      | --------- |
| device  | <------ IPC ------>  |   tap     |      |    IP     |
| MAC X   |                      |  bridge   |      |   stack   |
+---------+                      | --------- |      | --------- |
                                 |   ns-3    |      |   ns-3    |
                                 |   net     |      |   net     |
                                 |  device   |      |  device   |
                                 |  MAC Y    |      |  MAC Z    |
                                 +-----------+      +-----------+
                                       ||                 ||
                                 +------------------------------+
                                 |         ns-3 channel         |
                                 +------------------------------+
```

Mac X spoofed to Mac Y

UseBridge



```
+----------+
|  Linux   |                             +----------+
| -------  |                             |  ghost   |
|   apps   |                             |  node    |
| -------  |                             | -------- |
|  stack   |                             |    IP    |     +----------+
| -------  | +--------+                  |  stack   |     |   node   |
| Virtual  | |  TAP   |                  |==========|     | -------- |
| Device   | | Device | <---- IPC -----> |   tap    |     |    IP    |
+----------+ +--------+                  |  bridge  |     |   stack  |
    ||            ||                     | -------- |     | -------- |
+--------------------+                   |   ns-3   |     |   ns-3   |
| OS (brctl) Bridge  |                   |   net    |     |   net    |
+--------------------+                   |  device  |     |  device  |
                                         +----------+     +----------+
                                              ||               ||
                                         +---------------------------+
                                         |        ns-3 channel       |
                                         +---------------------------+
```

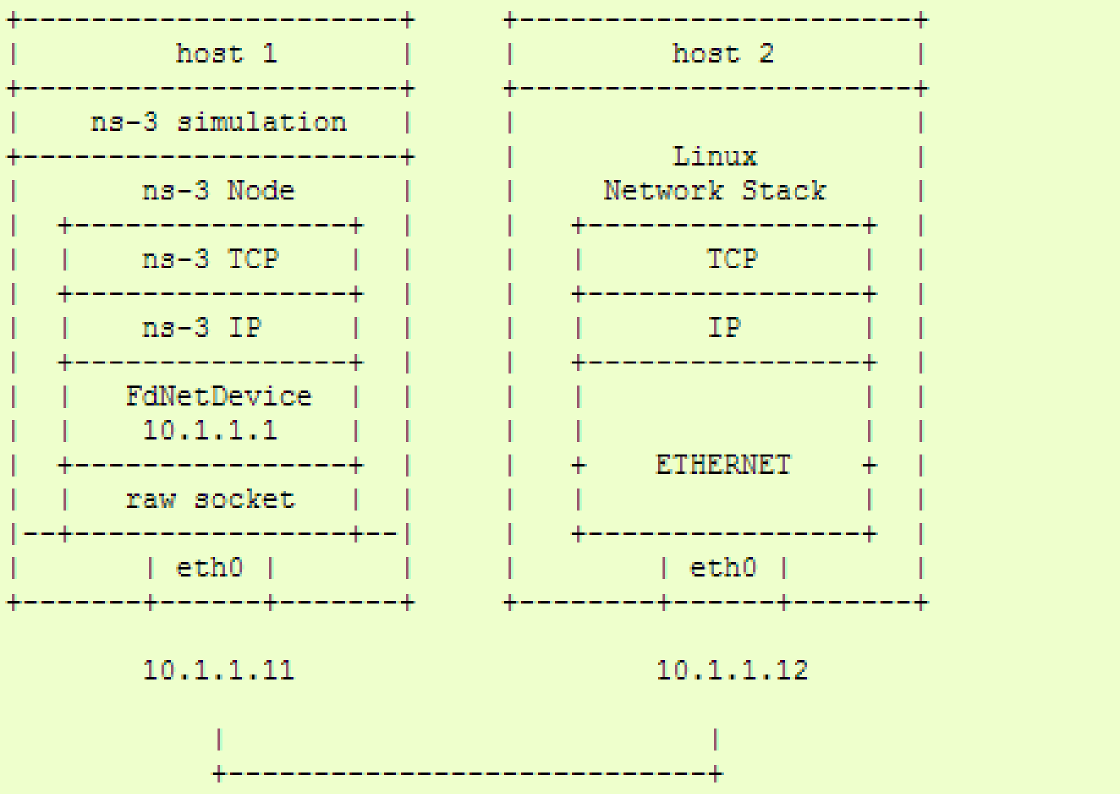
ns-3 devices must support SendFrom()
(i.e. bridging)

FdNetDevice

- Unified handling of reading/writing from file descriptor
- Three supported helper configurations:
  - EmuFdNetDeviceHelper (to associate the ns-3 device with a physical device in the host machine)
  - TapFdNetDeviceHelper (to associate the ns-3 device with the file descriptor from a tap device in the host machine) (not the same as TapBridge)
  - PlanetLabFdNetDeviceHelper (to automate the creation of tap devices in PlanetLab nodes, enabling ns-3 simulations that can send and receive traffic though the Internet using PlanetLab resource.

EmuFdNetDeviceHelper

- Device performs MAC spoofing to separate emulation from host traffic



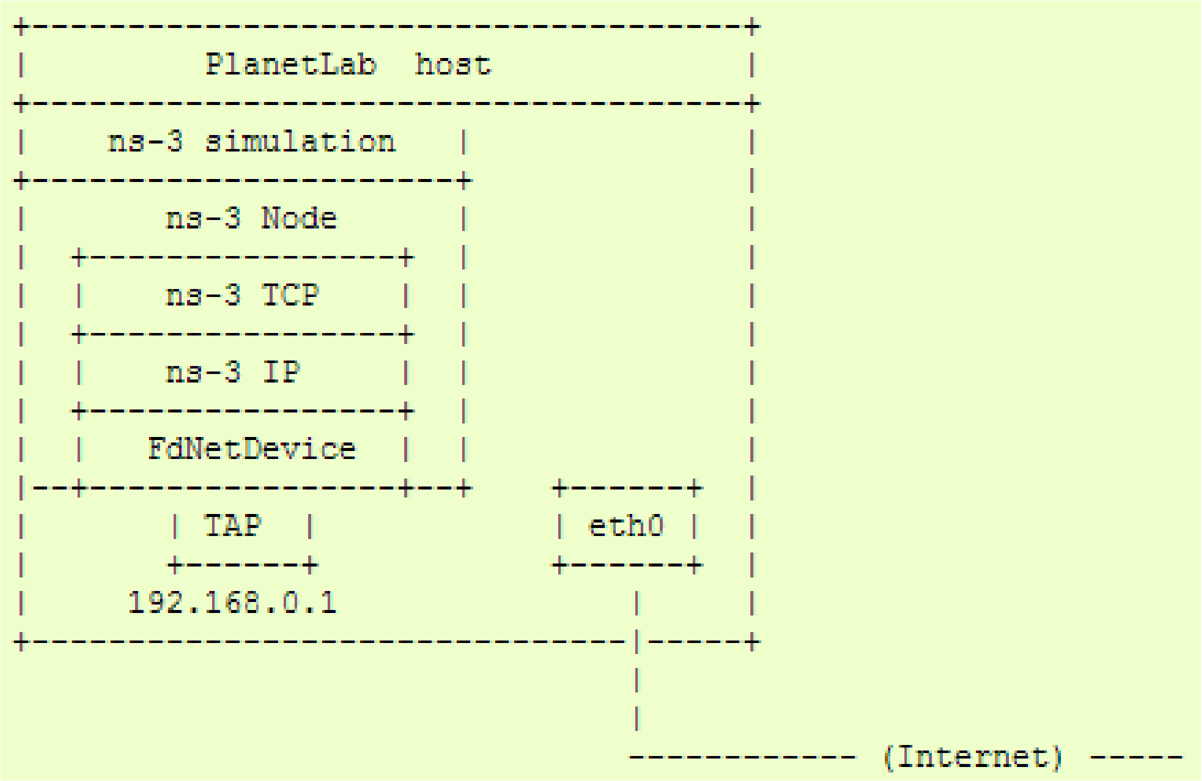
```
+----------------------+     +-----------------------+
|         host 1       |     |         host 2        |
+----------------------+     +-----------------------+
|    ns-3 simulation   |     |                       |
+----------------------+     |         Linux         |
|       ns-3 Node      |     |     Network Stack     |
|  +----------------+  |     |   +----------------+  |
|  |    ns-3 TCP    |  |     |   |       TCP      |  |
|  +----------------+  |     |   +----------------+  |
|  |    ns-3 IP     |  |     |   |       IP       |  |
|  +----------------+  |     |   +----------------+  |
|  |   FdNetDevice  |  |     |   |                |  |
|  |    10.1.1.1    |  |     |   |                |  |
|  +----------------+  |     |   +    ETHERNET    +  |
|  |   raw socket   |  |     |   |                |  |
|--+----------------+--|     |   +----------------+  |
|       | eth0 |       |     |        | eth0 |       |
+-------+------+-------+     +--------+------+-------+

        10.1.1.11                     10.1.1.12

            |                            |
            +----------------------------+
```

PlanetLabFdNetDeviceHelper

- Special case of TapFdNetDeviceHelper where Tap devices configured according to PlanetLab conventions



```
+-------------------------------------+
|         PlanetLab  host             |
+-------------------------------------+
|    ns-3 simulation    |             |
+----------------------+              |
|       ns-3 Node       |             |
|  +----------------+   |             |
|  |   ns-3 TCP     |   |             |
|  +----------------+   |             |
|  |   ns-3 IP      |   |             |
|  +----------------+   |             |
|  |  FdNetDevice   |   |             |
|--+----------------+---+    +------+ |
|       | TAP  |            | eth0 |  |
|       +------+            +------+  |
|     192.168.0.1              |       |
+-------------------------------|-----+
                                |
                                |
                                ------------ (Internet) -----
```

ns-3 over host sockets

- Two publications about how to run ns-3 applications over real hosts and sockets
  - "Simulator-agnostic ns-3 Applications", Abraham and Riley, WNS3 2012
  - Gustavo Carneiro, Helder Fontes, Manuel Ricardo, "Fast prototyping of network protocols through ns-3 simulation model reuse", Simulation Modelling Practice and Theory (SIMPAT), vol. 19, pp. 2063–2075, 2011.

Generic Emulation Issues

- Ease of use
  - Configuration management and coherence
  - Information coordination (two sets of state)
    - e.g. IP/MAC address coordination
  - Output data exists in two domains
  - Debugging can be more challenging
- Error-free operation (avoidance of misuse)
  - Synchronization, information sharing, exception handling
    - Checkpoints for execution bring-up
    - Inoperative commands within an execution domain
    - Deal with run-time errors
  - Soft performance degradation (CPU) and time discontinuities