What is ns-3

ns-3

- is a series of discrete event network simulators.
- primarily used in research and teaching.
- is built using C++ and Python with scripting capability.

Real World vs ns-3

In real world, we want to connect two computers.
1.Build a channel to connect each other (ex:ethernet, wifi)
2.Need a Net Device to help us use channel
3.Use Application to transmit data

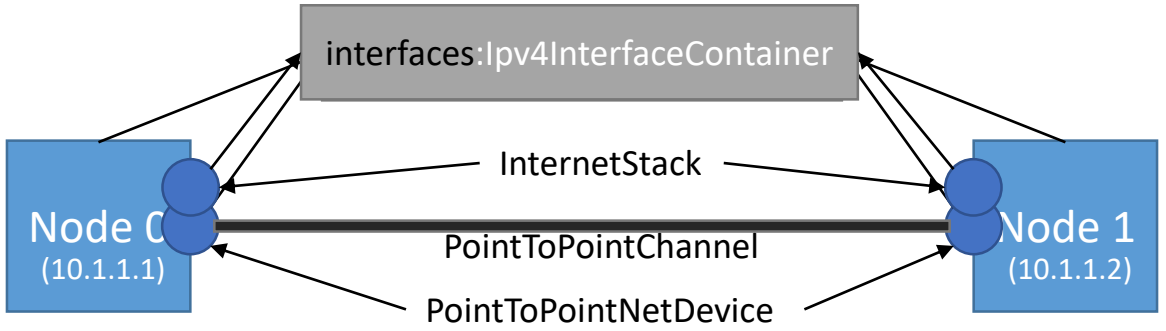
Real World vs ns-3

Topology

```
NodeContainer nodes;
nodes.Create (2);

PointToPointHelper pointToPoint;
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);

InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```



Real World vs ns-3

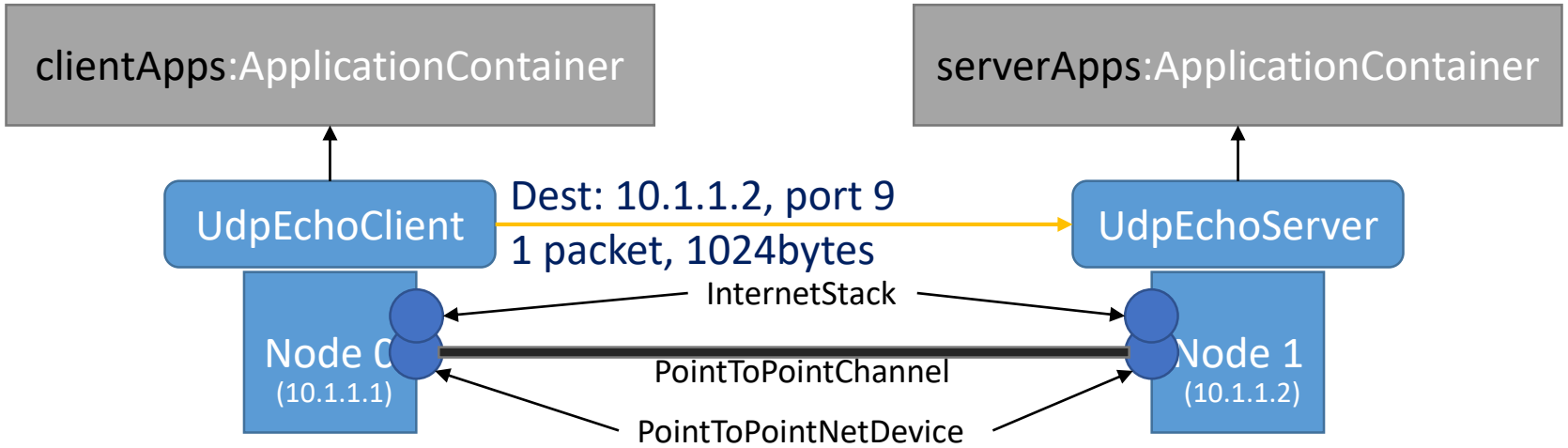
ApplicationContainer

We can use some application helper to help us build a application and install it on the nodes, then we can add to container.
A ApplicationContainer may includes different kind of application in a container.

Application

```
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```



Key Abstractions

- **Node:**
  - In real world, a computing device that connects to a network is called a host or sometimes an end system, but in ns-3, the basic computing device abstraction is called the node.
  - The Node class provides methods for managing the representations of computing devices in simulations.

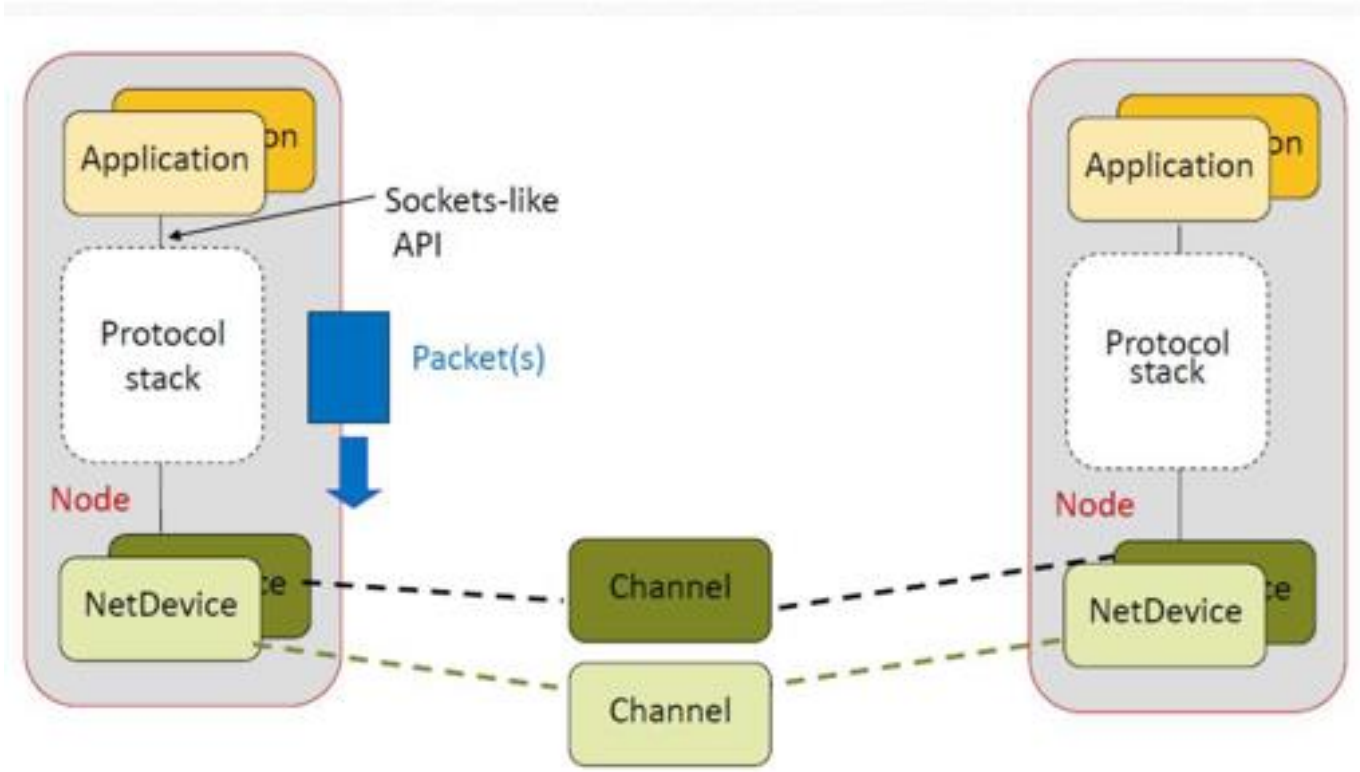
- **Channel:**
  - In the simulated world of *ns-3*, one connects a Node to an object representing a communication channel. Here the basic communication subnetwork abstraction is called the channel.
  - The Channel class provides methods for managing communication subnetwork objects and connecting nodes to them.
  - We will use specialized versions of the Channel called CsmaChannel, PointToPointChannel and WifiChannel in this tutorial.

- **Net Device:**
  - In ns-3 the net device abstraction covers both the software driver and the simulated hardware. A net device is “installed” in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels.
  - The Net Device class provides methods for managing connections to Node and Channel objects.
  - We will use the several specialized versions of the NetDevice called CsmaNetDevice, PointToPointNetDevice, and WifiNetDevice in this tutorial.

Key Abstractions

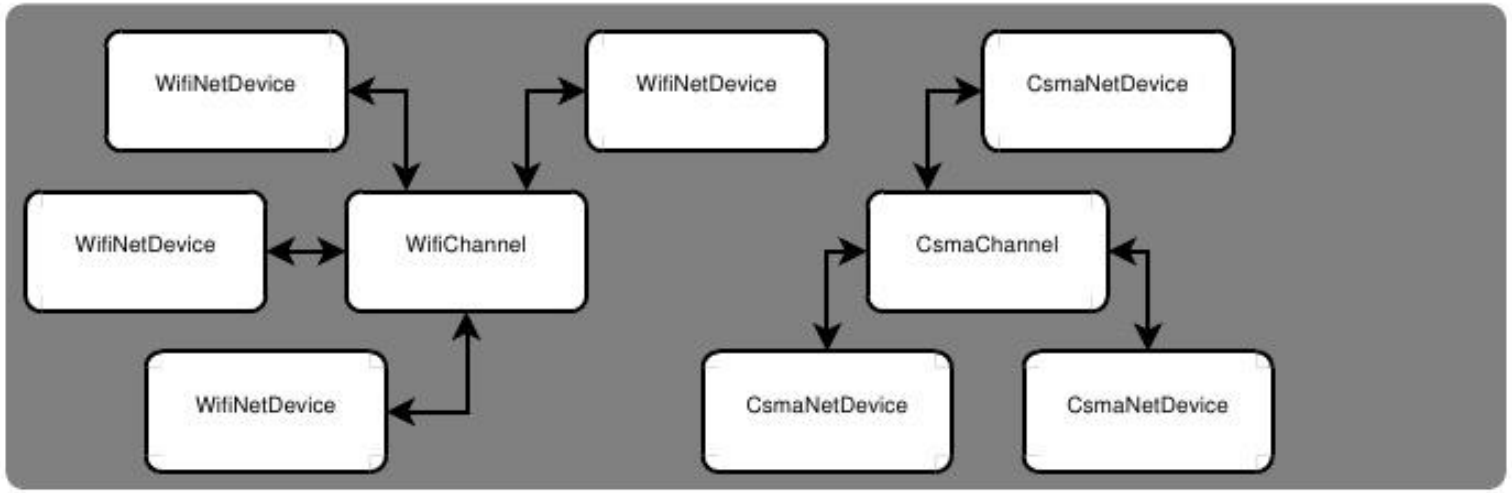
- **Topology Helpers:**
  - In ns-3 you will find Nodes with attached NetDevices. In a large simulated network you will need to arrange many connections between Nodes, NetDevices and Channels.
  - Since connecting NetDevices to Nodes, NetDevices to Channels, assigning IP addresses, etc., are such common tasks in ns-3, we provide what we call topology helpers to make this as easy as possible.
- **Application:**
  - We do have the idea of an application. Just as software applications run on computers to perform tasks in the "real world," *ns-3* applications run on *ns-3 Nodes* to drive simulations in the simulated world.
  - In *ns-3* the basic abstraction for a user program that generates some activity to be simulated is the application.
  - The Application class provides methods for managing the representations of our version of user-level applications in simulations.

ns-3 Model Architecture



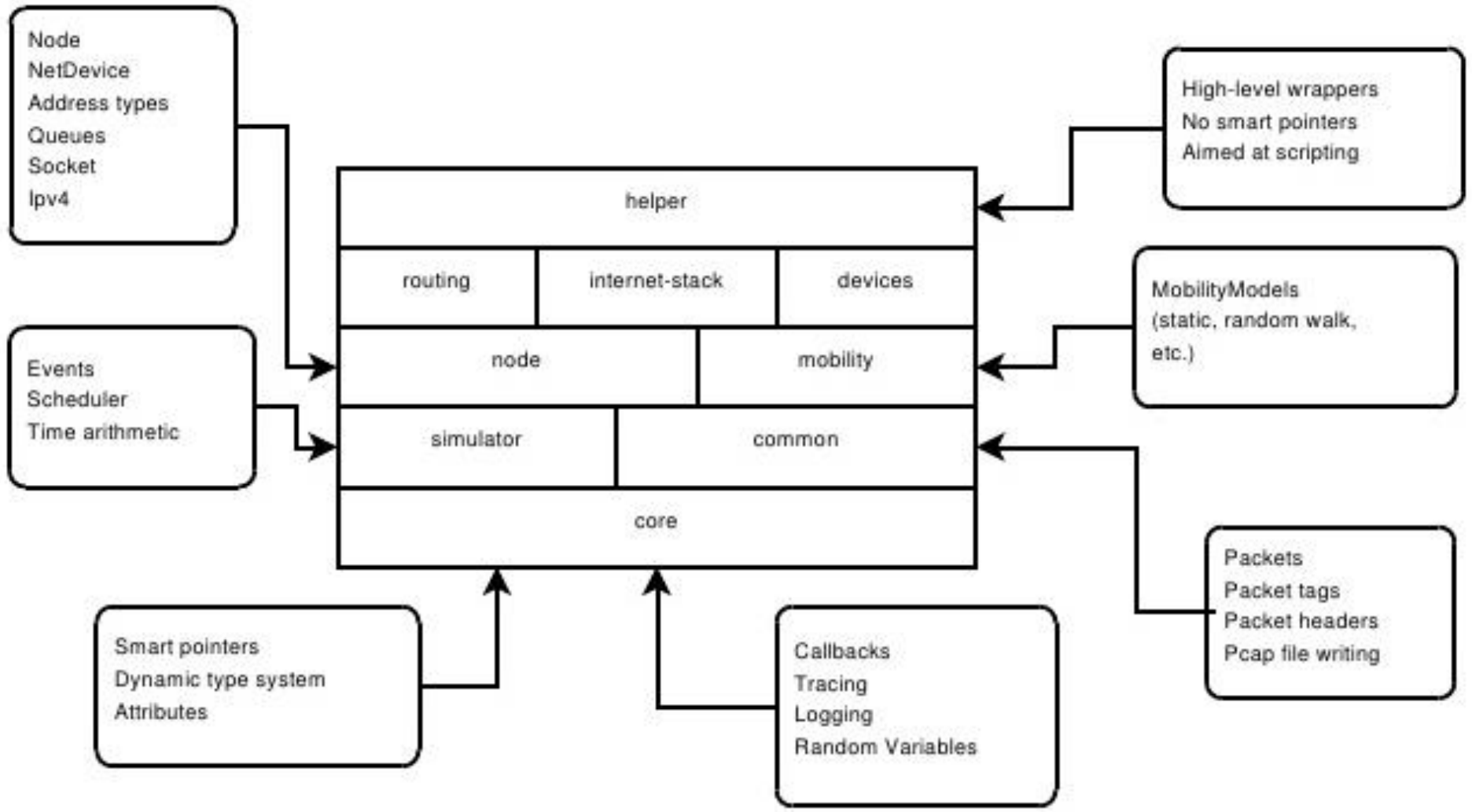
ns-3 NetDevice Models

NetDevices are strongly bound to Channels of a matching type:



PointToPointHelper: be used to the connection between two points or a node container
WifiHelper: 802.11 protocol for wireless LAN connection
CsmaHelper: 802.3 protocol for Ethernet connection
LteHelper: we can use this helper to help us create a UE or eNodeB NetDevice

ns-3 Concept


Node
NetDevice
Address types
Queues
Socket
Ipv4
High-level wrappers
No smart pointers
Aimed at scripting
helper
routing
internet-stack
devices
MobilityModels
(static, random walk,
etc.)
Events
Scheduler
Time arithmetic
node
mobility
simulator
common
core
Packets
Packet tags
Packet headers
Pcap file writing
Smart pointers
Dynamic type system
Attributes
Callbacks
Tracing
Logging
Random Variables

Installation

- Environment Recommendation
  - Ubuntu16.04
  - ns-3.26
    - Download ns-3.26 on the web and extract the folder
  - Packages requirements (Terminal):

```
apt-get install gcc g++ python python3 python3-dev
apt-get install qt5-default mercurial
apt-get install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev ipython
apt-get install autoconf cvs bzr unrar
apt-get install gdb valgrind
apt-get install uncrustify
apt-get install doxygen graphviz imagemagick
apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils texlive-lang-portuguese dvipng latexmk
apt-get install python3-sphinx dia
apt-get install gsl-bin libgsl-dev
apt-get install tcpdump sqlite sqlite3 libsqlite3-dev
apt-get install libxml2 libxml2-dev
apt-get install libgtk2.0-0 libgtk2.0-dev
apt-get install python-dev python-gnome2 python-rsvg
```

Installation

- Install ns-3 and NetAnim on ubuntu (Terminal):

```
# install ns-3
cd ns-allinone-3.26
./build.py
cd ns-3.26

# test whether the ns-3 is installed
cp -rf examples/tutorial/first.cc scratch
./waf --run scratch/first

# install NetAnim
cd ..
cd netanim-3.107
make clean
qmake NetAnim.pro
make ./NetAnim
```

- Reference
  - ns-3 Install (link1、link2)
  - ns-3 Tutorial (link)

How to use NetAnim to animate our project?

```
#include “ns3/netanim-module.h”

.
.
.

AnimationInterface anim (“ex.xml”);
anim.SetStartTime (Seconds(0));
Anim.SetStopTime(Seconds(simTime));
```

Then, you can open NetAnim to import “ex.xml” file.