MODULES & SCRIPTS

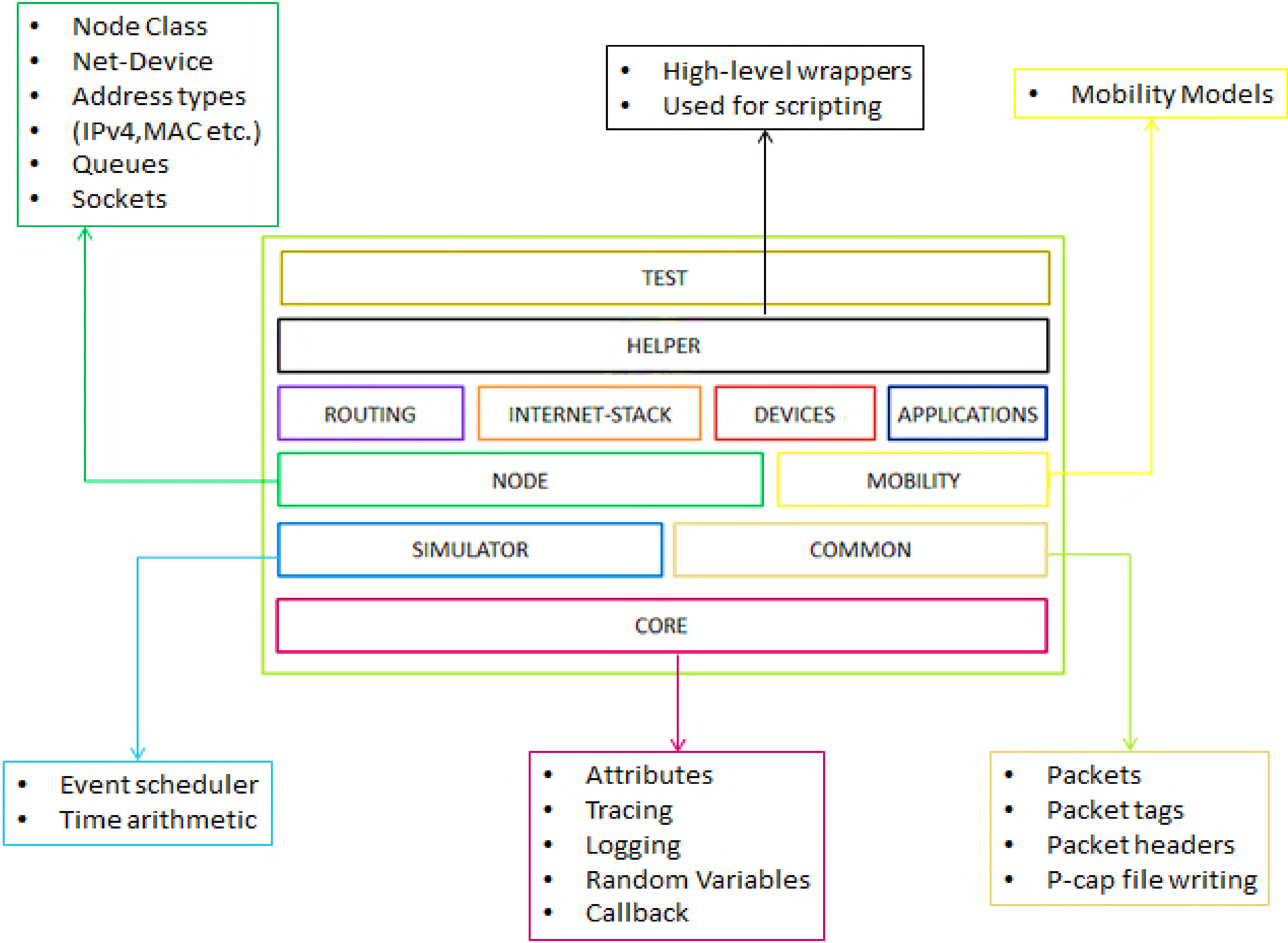
NS-3 MODULES

- NS-3 software organized into modules, each built as a separate software library.

- NS-3 models are abstract representations of real-world objects, protocols, devices etc.

NS-3 MODULES

- Simulation core & models implemented in C++
- NS-3 built as a library, can be statically/ dynamically linked to a C++ program.
- NS-3 ‘exports’ almost all of its API to python which allows python programs to ‘import’ ns3 module the same way as for C++ programs.
- Source code for ns3 organized in /src directory.


Node Class
Net-Device
Address types
(IPv4,MAC etc.)
Queues
Sockets
High-level wrappers
Used for scripting
Mobility Models
TEST
HELPER
ROUTING
INTERNET-STACK
DEVICES
APPLICATIONS
NODE
MOBILITY
SIMULATOR
COMMON
CORE
Event scheduler
Time arithmetic
Attributes
Tracing
Logging
Random Variables
Callback
Packets
Packet tags
Packet headers
P-cap file writing

CORE

- Components common across all protocol, hardware & environmental models.
- Simulation core is implemented in src/core.
- Attributes, callbacks, tracing, logging, random variables, smart pointers etc.

CORE

- Packets are fundamental objects and implemented in src/network (common)

- These 2 modules core and network comprises a generic simulation core that can be used by different type of networks.

CORE

- Random variables: NS3 contains built-in pseudo random number generator (PRNG) to obtain randomness in simulations.
- Attributes: To configure network element models with a set of default values.
  - Configuration: In topology (code) or inside module where the value is instantiated.
- Tracing: To trace packet info, in a structured format. In ns3, pcap (packet capture) files.

CORE

- Logging: To monitor/debug progress of simulation programs.
- Callbacks: To allow a piece of code to call another method without module inter-dependency.
- Smart Pointers: To manage dynamically allocated memory.

Helper

- Provides set of classes and methods that makes common operations easy to code.
- NS-3 programs may access all API's directly / make use of helper API that provides wrappers/encapsulation of low-level API calls.
- Anything that can be done here can be done at low level APIs also.

ns-3 NETWORK SIMULATOR

Helper

- Contains container & helper classes
- Containers: Often simulations will need to do a number of identical actions to groups of objects (similar operations can be performed)
- Container classes: NodeContainer, NetDeviceContainer, Ipv4AddressContainer.
- Helper classes: InternetStackHelper, WifiHelper, MobilityHelper, OlsrHelper.

Simulator

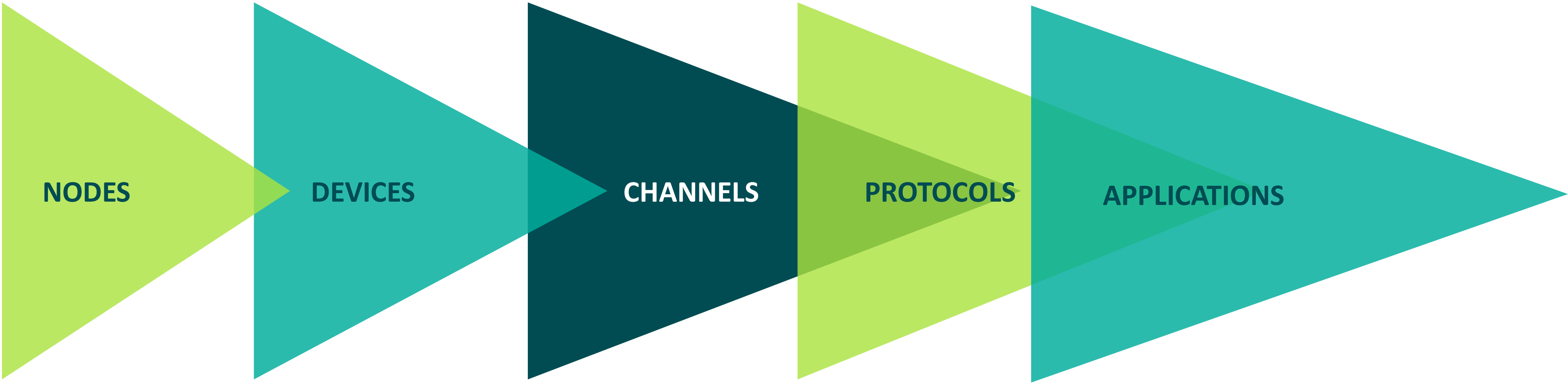
- Simulator keeps tracks of events scheduled to execute at specific simulation time in sequential time order.
- To implement this, following things are needed:
  - Simulator object that can access an event queue (stores and manages execution of events).
  - Scheduler to insert/remove events from queue.
    - Represent simulation time
  - Events

ns-3 NETWORK SIMULATOR

NS-3 Modules

- Network: address, channel, net-device, queue, socket, packet
- Internet-stacks: arp, ipv4, icmpv4,udp,tcp.
- Devices: point-to-point, csma, wiFi, bridge.
- Applications: udp echo, on/off, sink etc.
- Mobility models: random walk, random direction 2D, constant acceleration etc.
- Routing: olsr, static global etc.
- Error models

KEY ABSTRACTIONS "NS-3"

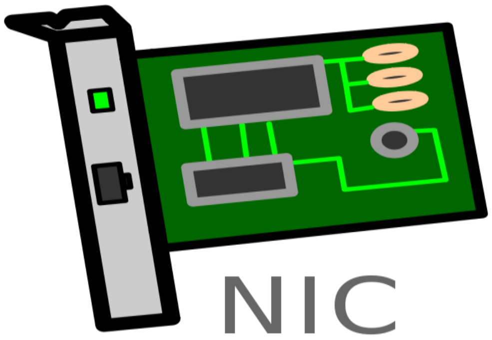


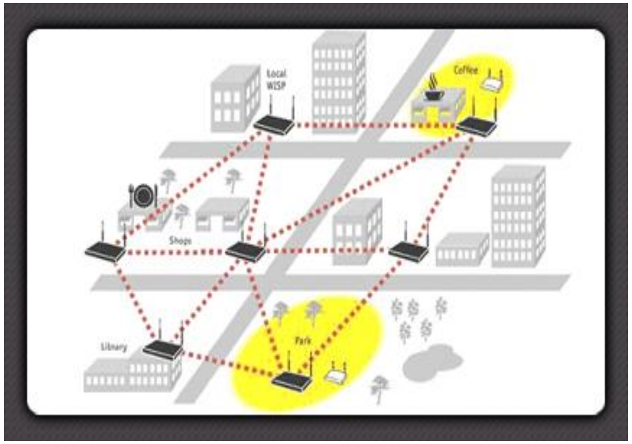
NODES

- End systems (computers/laptops), routers, hubs, switches.
- Represented by class “Node”
- Programmer can add functionality to it e.g., applications, peripheral devices, protocol stacks etc.

Network Devices

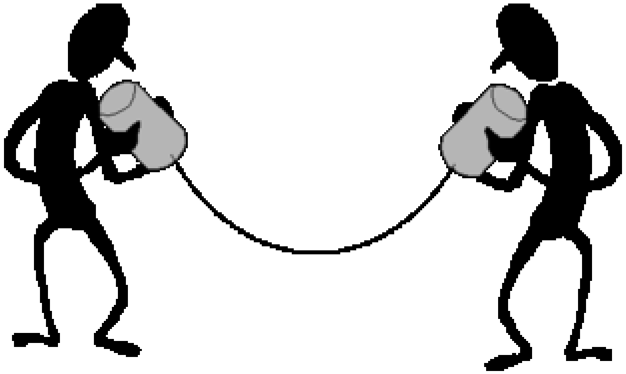
- Physical devices that connects a node to communications channel (NIC, complex wireless IEEE 802.11 device).
- Incorporates physical and MAC layer.





Communication Channel

- Medium to send info between network devices(fiber point-to-point links, shared broadcast media or wireless channel etc.).



Communication Protocols

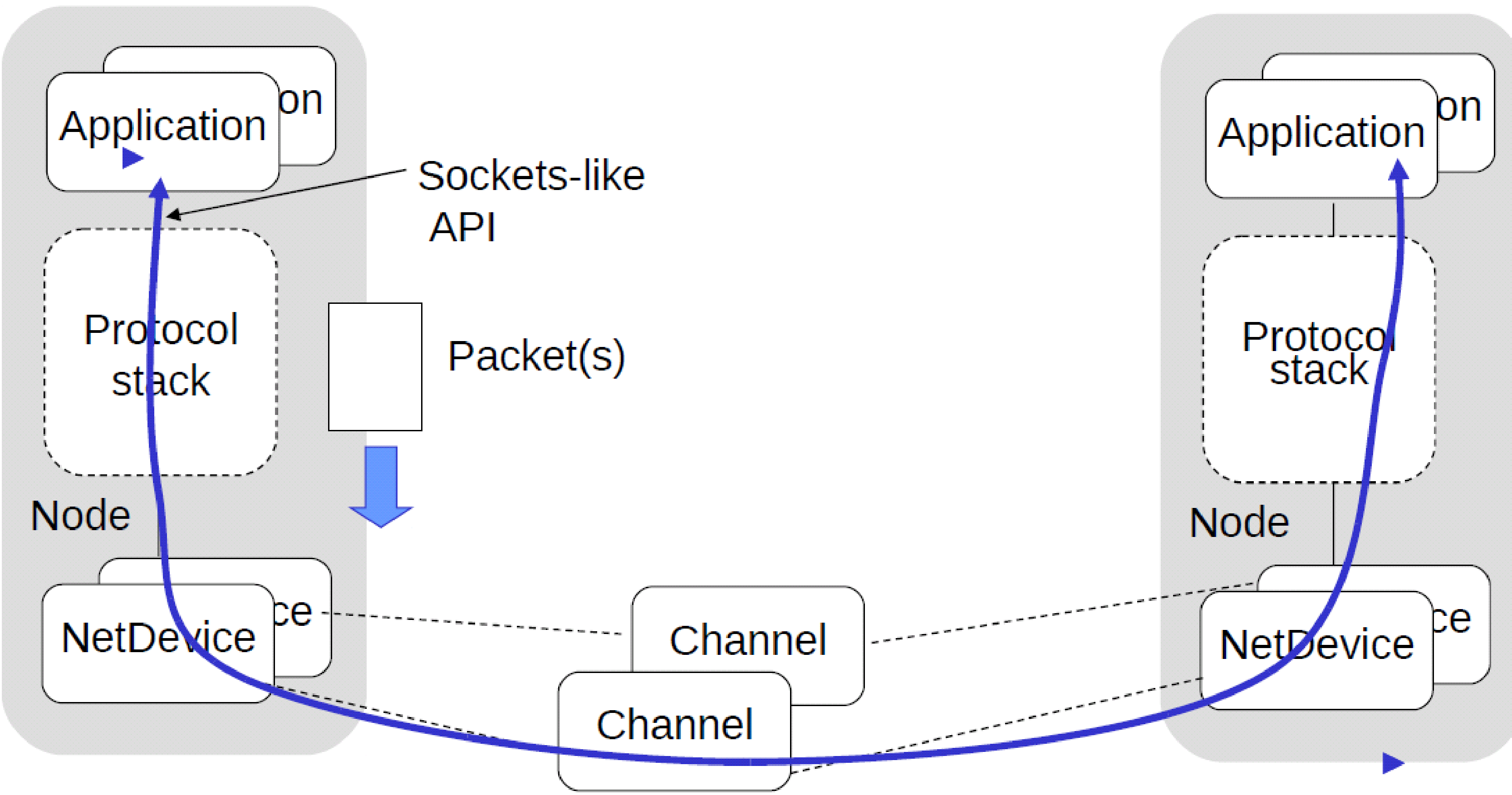
- Models the implementations of protocol descriptions found in Internet RFC documents
- Also newer experimental protocols not yet standardized

NS-3 NETWORK SIMULATOR

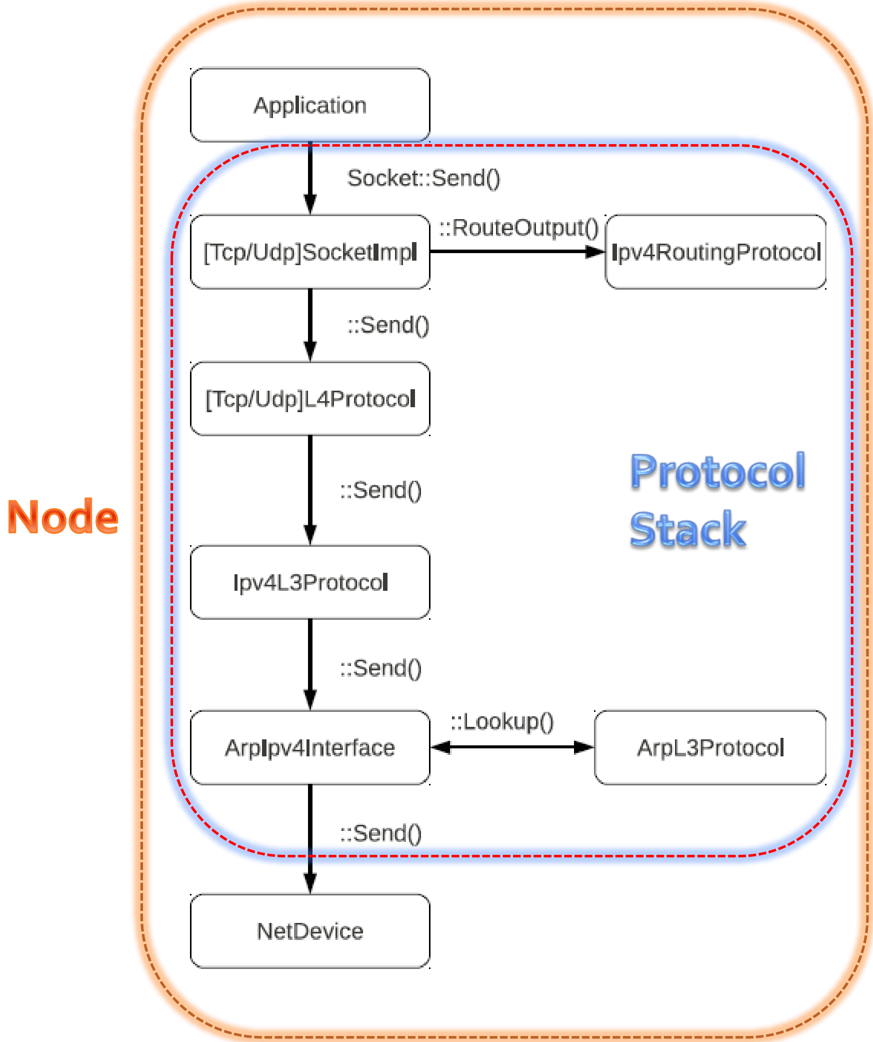
Applications

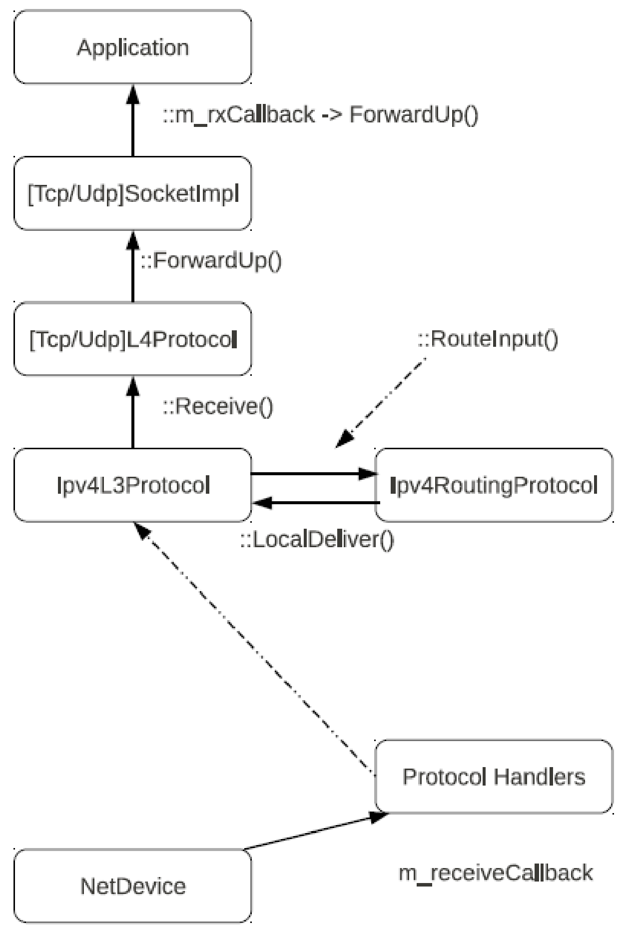
- *System Software* manages system resources, doesn't use them to complete tasks that directly benefit a user.
- A user runs an *application* that acquires & uses resources controlled by system software to accomplish some goal
- NS3 applications run on nodes to drive simulations in simulated world.

Conceptual Implementation



Actual Implementation



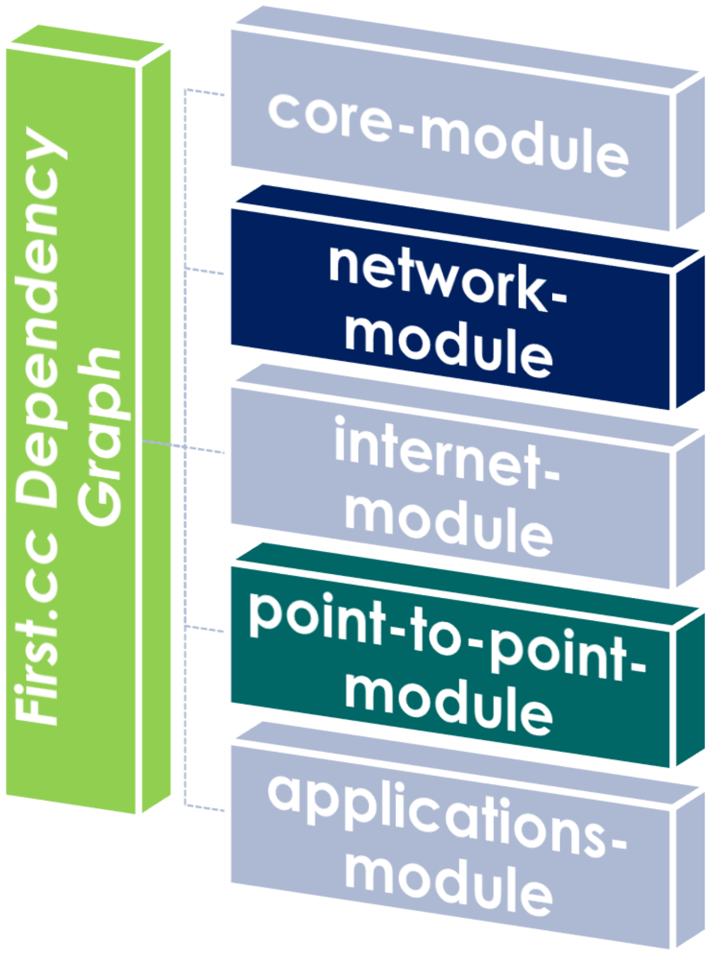


first.cc

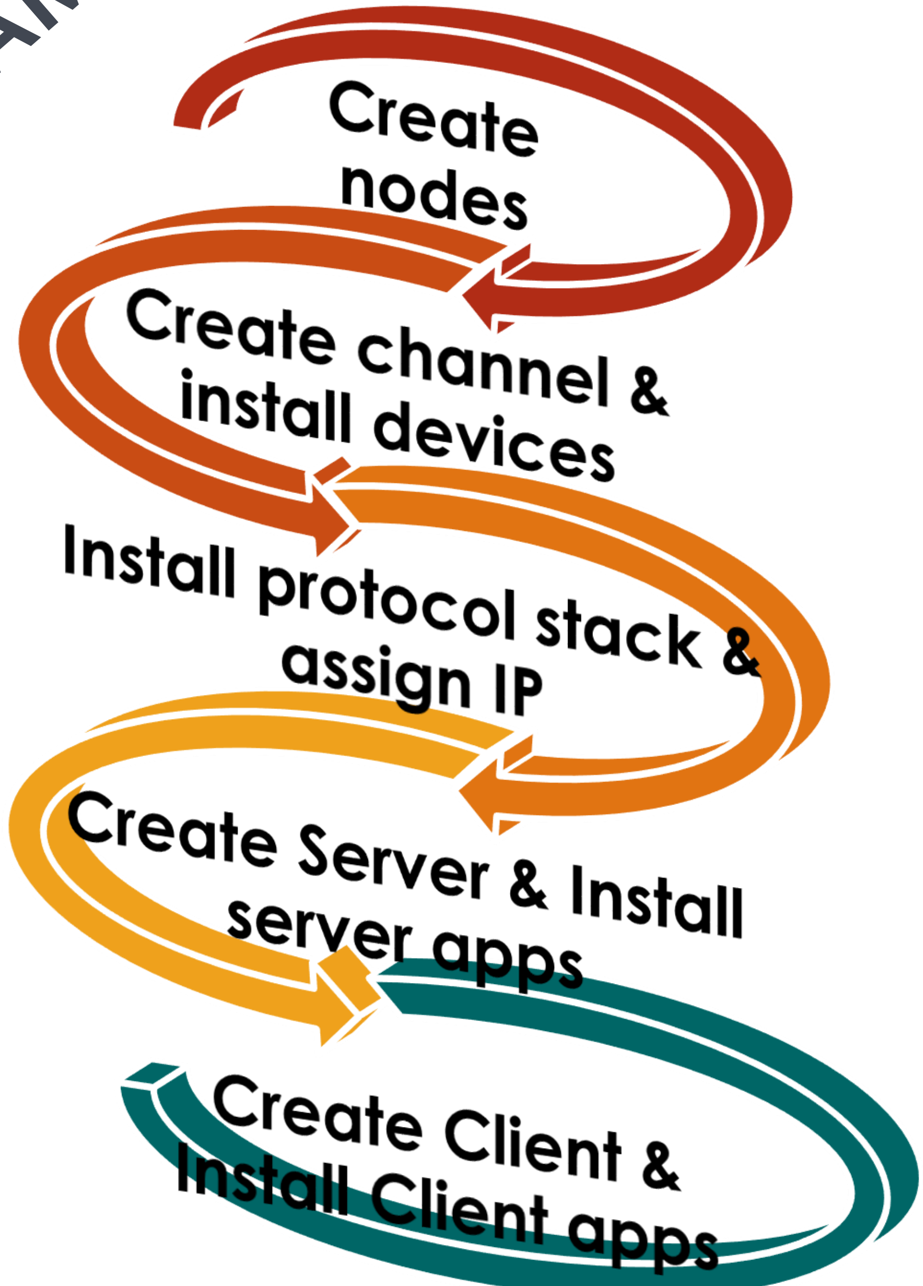
Script file in examples

Dependency Graph

- NS-3 grouped together related files in a module.
- These modules provide the ability to load a group of files at a large granularity.
- Easy to code.



FLOW DIAGRAM



Code Breakdown

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
// GPLv2 Licence …

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int main (int argc, char *argv[])
{
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

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

include modules that will be used

ns-3 project namespace

enable and disable console message logging by reference to the name

Topology Configuration

Code Breakdown

```
InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);

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);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

Set up internet stack

Set up applications

Run the simulation

Script Execution

ALL SCENARIOS SHOULD BE RUN UNDER SCRATCH

```
% cp examples/tutorial/first.cc scratch/myfirst.cc
% ./waf
% ./waf --run /scratch/myfirst
% Waf: Entering directory '/scratch/ns3-
  workshop/ns-allinone-3.13/ns-3.13/build'
Waf: Leaving directory '/scratch/ns3-workshop/ns-
  allinone-3.13/ns-3.13/build'
'build' finished successfully (1.218s)
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2
```