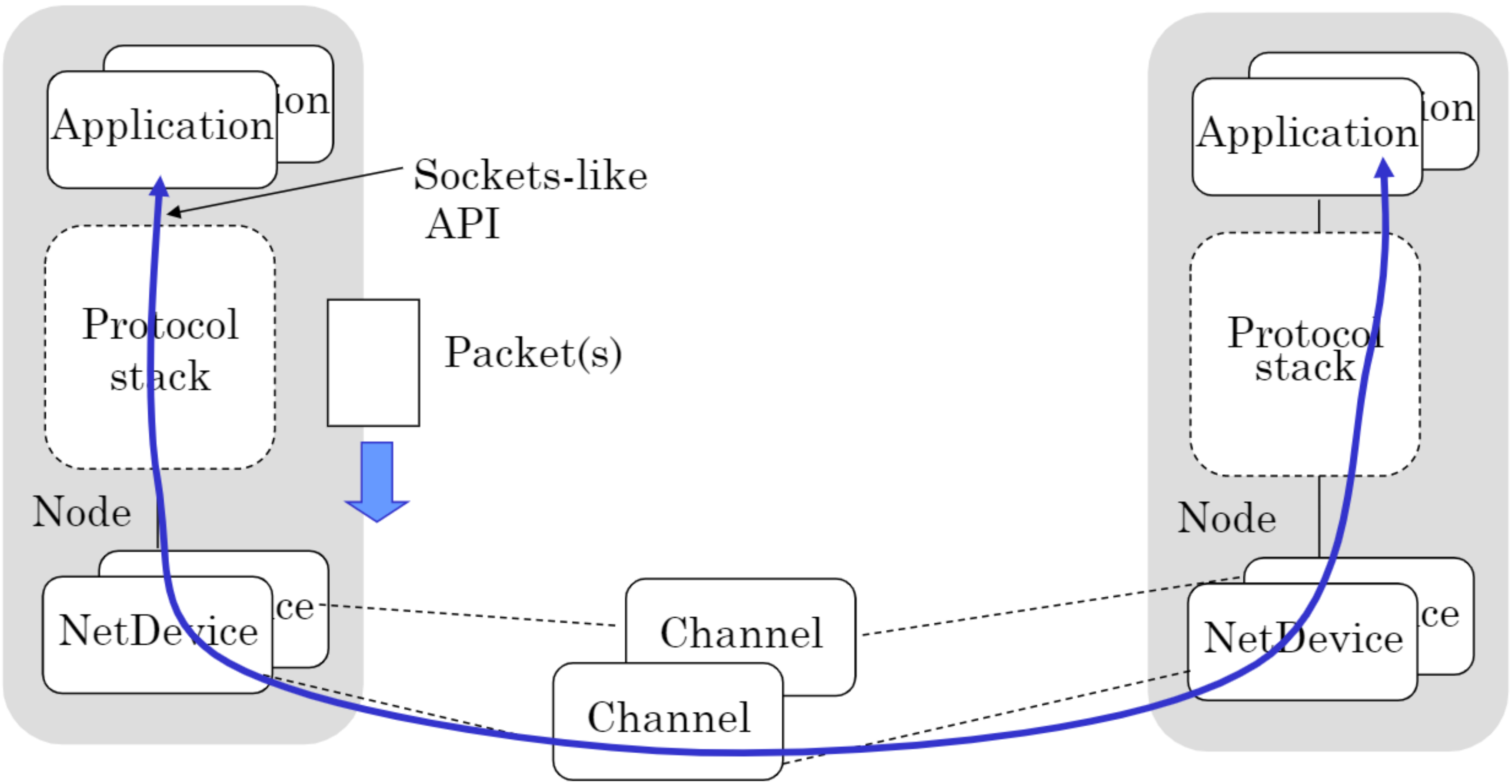
MOBILE NETWORK PERVASIVE COMPUTING (POINT TO POINT PROTOCOL)

Mochammad Zen Samsono Hadi, ST. MSc. Ph.D

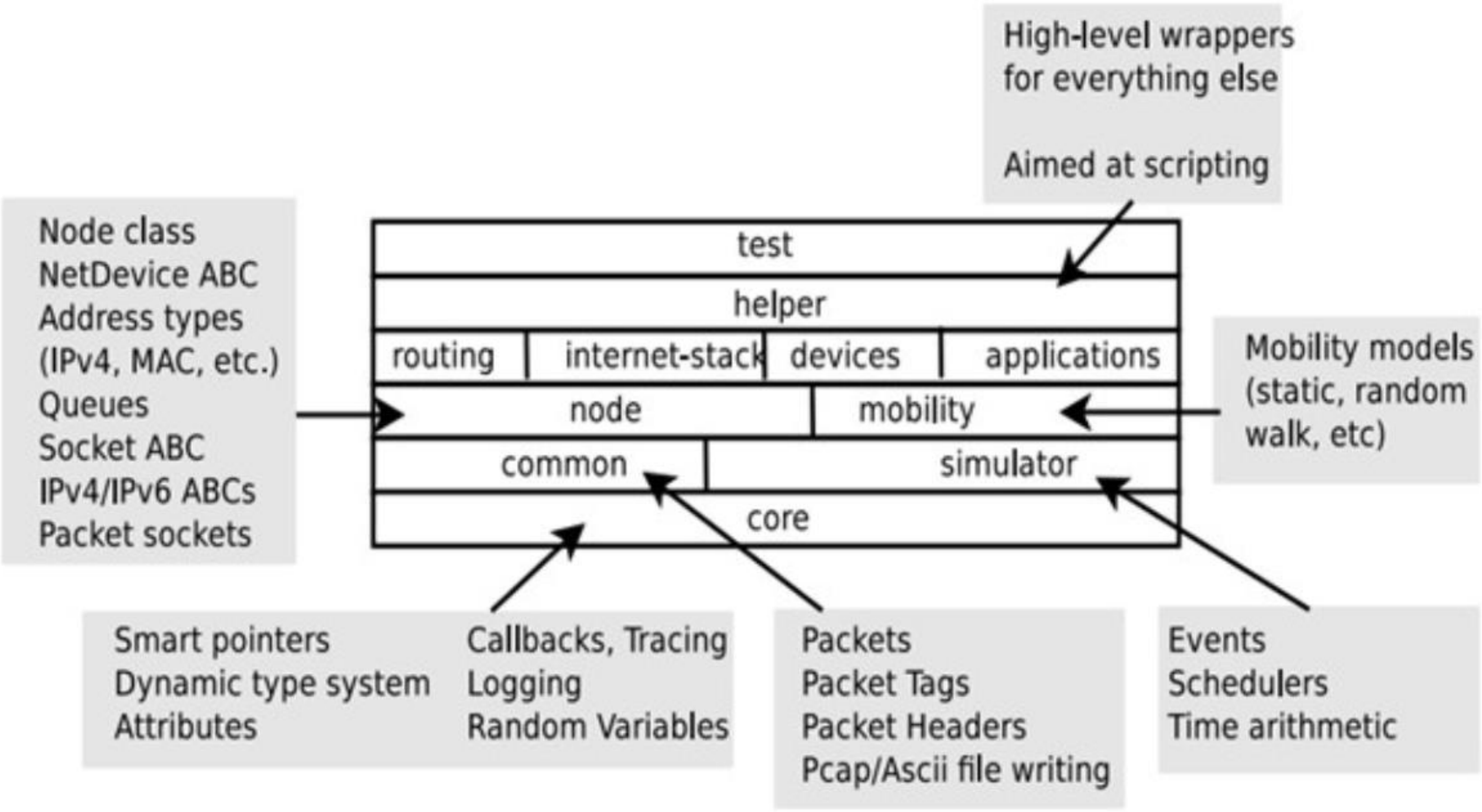
TOPIK PEMBAHASAN

- Fundamental of ns-3
- Point-to-point Protocol
- Illustrations of simulations
- Tracing output

Basic Model


Application
Sockets-like
API
Protocol
stack
Packet(s)
Node
NetDevice
Channel
Channel
Application
Protocol
stack
Node
NetDevice

NS-3 Modules

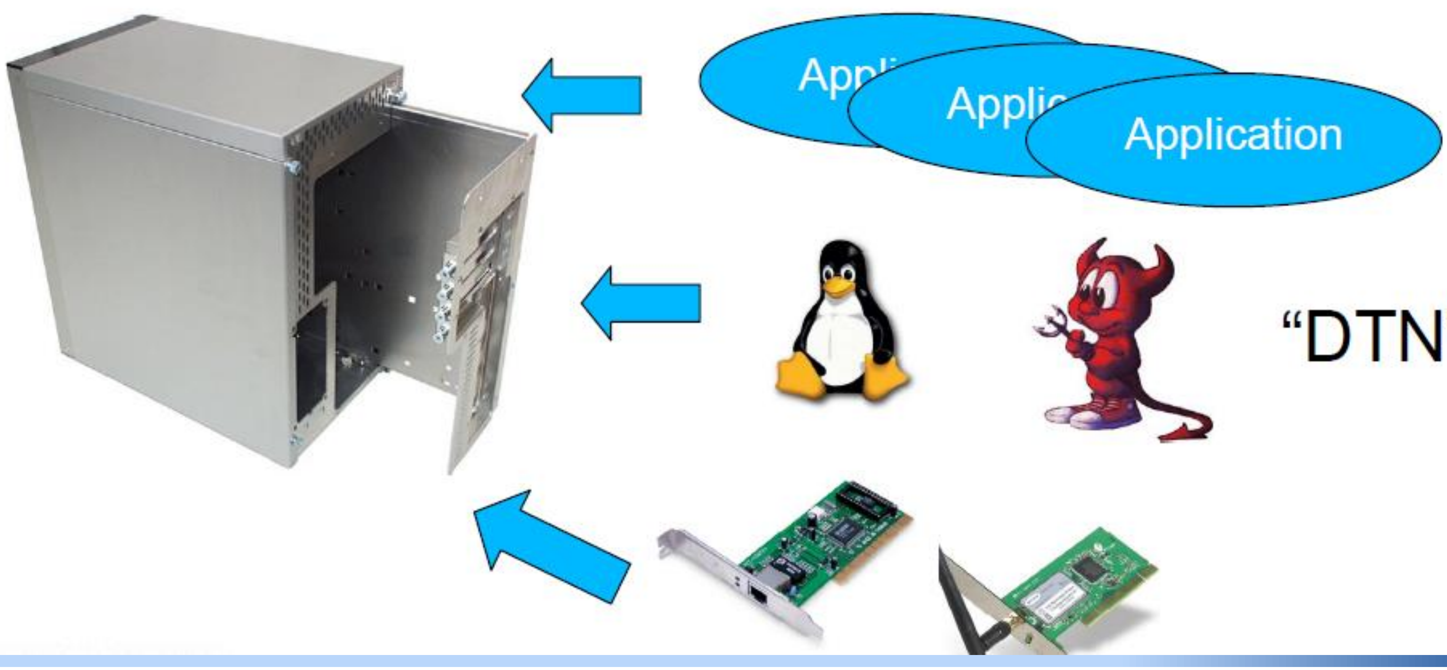

High-level wrappers
for everything else
Aimed at scripting
test
helper
routing
internet-stack
devices
applications
node
mobility
common
simulator
core
Node class
NetDevice ABC
Address types
(IPv4, MAC, etc.)
Queues
Socket ABC
IPv4/IPv6 ABCs
Packet sockets
Mobility models
(static, random
walk, etc)
Smart pointers
Dynamic type system
Attributes
Callbacks, Tracing
Logging
Random Variables
Packets
Packet Tags
Packet Headers
Pcap/Ascii file writing
Events
Schedulers
Time arithmetic

Fundamentals

- Key object in the simulator are:
  - Nodes
  - Packets
  - Channels: CsmaChannel, PointToPointChannel, WifiChannel
- Nodes contain
  - Applications: user-level applications
  - “stacks”: 7-layers OSI
  - NetDevices (NIC): CsmaNetDevice, PointToPointNetDevice, WifiNetDevice
- Topology helpers
  - In a large simulated networks, it will need to arrange many connections between Nodes, NetDevices and Channels
  - We can connect NetDevices to Nodes, NetDevices to Channels, and assign IP addresses to NetDevices

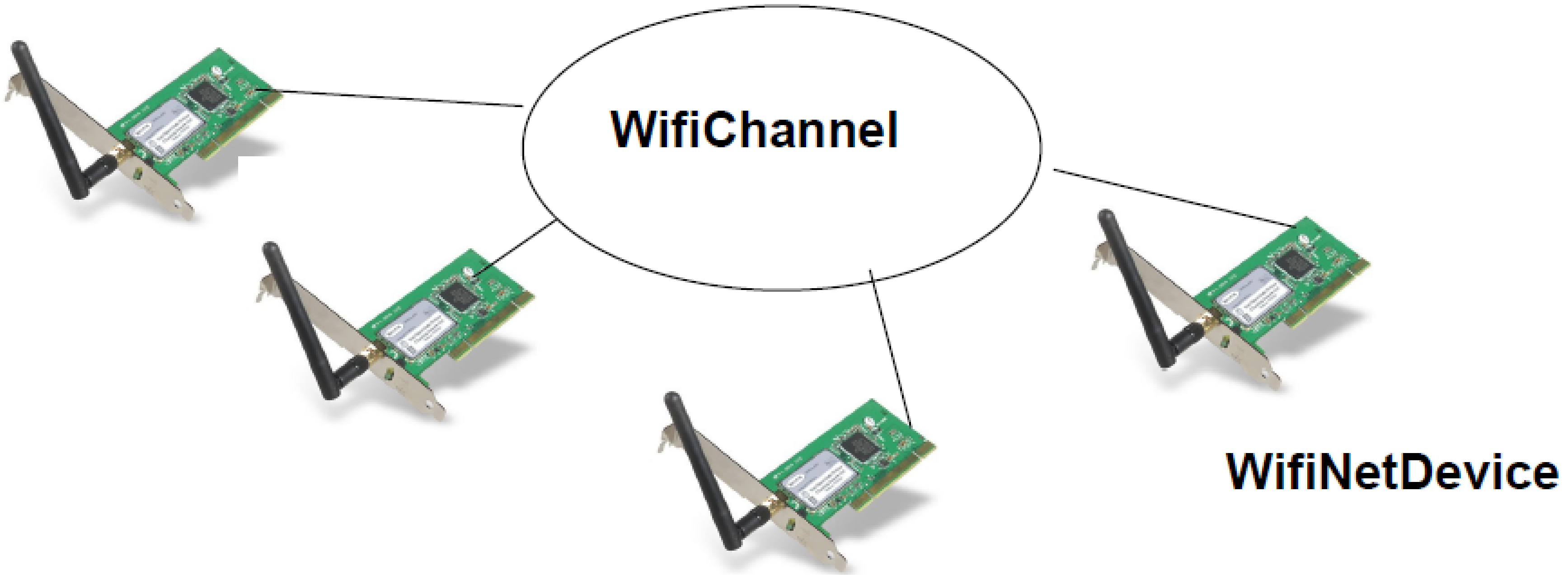
Node Basics

- Basic computing device abstraction
- Node class provides methods for managing computing devices
- A Node is a husk of a computer to which applications, stacks, and NICs are added



Net Devices and Channels

- NetDevices are strongly bound to Channel of a matching type
- Nodes are architected for multiple interface

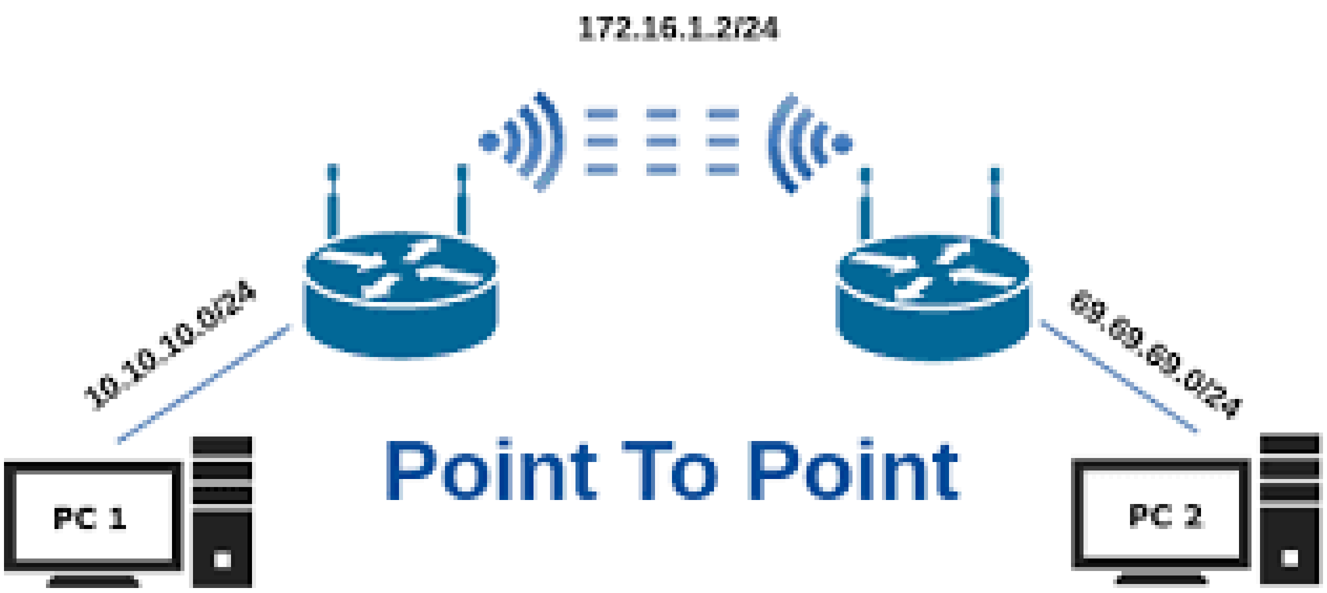


ns-3 Packets

- Each network packet contains a byte buffer, a list of tags, and metadata
  - buffer: bit-by-bit (serialized) representation of headers and trailers
  - tags: set of arbitrary, user-provided data structures (e.g., per-packet cross-layer messages, or flow identifiers)
  - metadata: describes types of headers and trailers that have been serialized
    - optional – disabled by default
- To add a new header, subclass from Header, and write Serialize() and Deserialize() methods
  - How bits get written to/from the Buffer

Point to point protocol

- **Point-to-Point Protocol** (**PPP**) is a data link layer (layer 2) communications **protocol** between two routers directly without any host or any other **networking** in between. It can provide connection authentication, transmission encryption, and compression.



Ptop Protocol in ns3

- *ns-allinone-3.25/ns-3.25/scratch/ptop.cc*

```
#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"
#include "ns3/netanim-module.h" // entered for animation
configuration and output file
```

- Location of the files in ns3: */ns-allinone-3.29/ns-3.29/build/ns3/*

core-module.h: determine object, event, simulator, timer

network-module.h: node, node-container, net-device, net-device-container, application, channel, data-rate, flow control, socket, interface, header, ipv4, ipv6, mac address, packet data, node, queue

point-to-point-module.h: channel, helper, net-device, remote-channel, header

application-module.h: http client-server, udp client-server, ftp apps (bulk-send)

Functions in point-to-point-helper.h

- void SetQueue: set the type of queue to the NetDevice
- void SetDeviceAttribute: set attributes on each PointToPointNetDevice
- void SetChannelAttribute: set attributes on each PointToPointChannel
- NetDeviceContainer Install: make NodeContainer
- virtual void EnablePcapInternal: enable pcap output of net device
- virtual void EnableAsciiInternal: enable ascii trace output of net device

https://www.nsnam.org/doxygen/classns3_1_1_internet_stack_helper.html#details

Helper Objects

- NodeContainer: vector of Ptr<Node>
- NetDeviceContainer: vector of Ptr<NetDevice>
- InternetStackHelper
- WifiHelper
- MobilityHelper
- OlsrHelper
- Each model provides a helper class

Listing Program (ptop.cc) 1/2

set application

```
Time::SetResolution (Time::NS);
LogComponentEnable ("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication",LOG_LEVEL_INFO);
```

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

Create 2 nodes

set ptop with attr.

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

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

Create net device on each node

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

Set IP/TCP/UDP and routing protocol on each node and enable pcap and tracing of events in the internet stack associated with a node

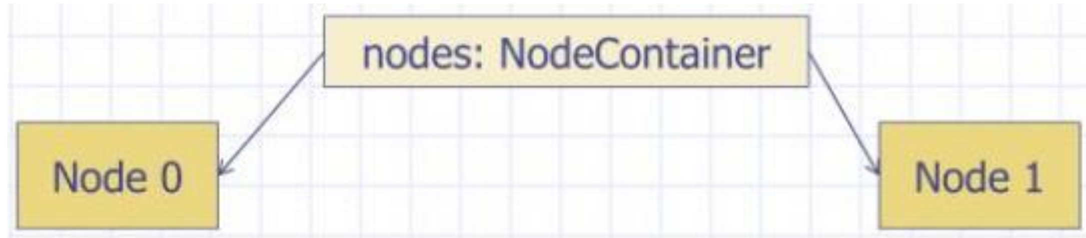
```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0","255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

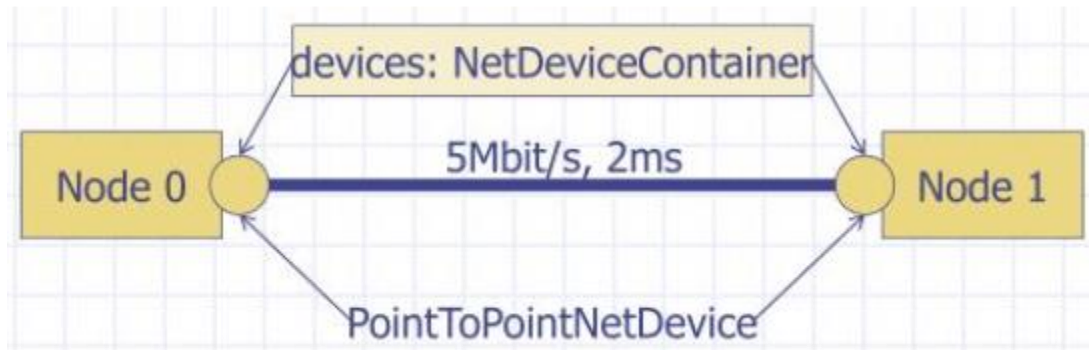
Set ipv4 address (net id) on each node

illustrations of ptop

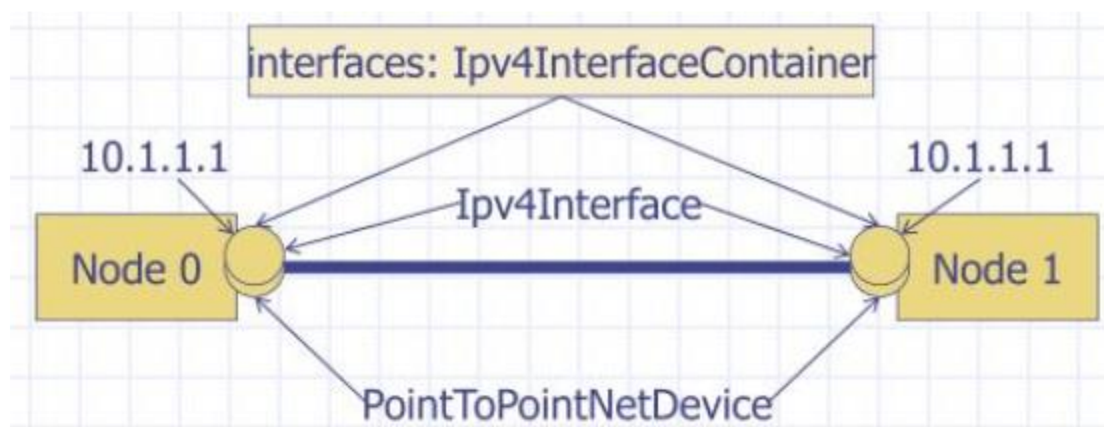
Create 2 nodes



PointToPoint protocol



Internet Stack
(ipv4 address)



Listing Program (ptop.cc) 2/2

```
UdpEchoServerHelper echoServer(9);

ApplicationContainer serverApps = echoServer.Install (nodes.Get(1));

serverApps.Start (Seconds (3.0));
serverApps.Stop (Seconds (10.0));
```

set UDP server

```
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets",UintegerValue(1024));
echoClient.SetAttribute ("Interval", TimeValue (Seconds(1.0)));
echoClient.SetAttribute ("PacketSize",UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (nodes.Get(0));

clientApps.Start (Seconds (4.0));
clientApps.Stop (Seconds (10.0));
```

set UDP client

```
//Animation configuration lines
AnimationInterface anim ("ptop.xml");
anim.SetConstantPosition (nodes.Get(0), 3.0, 3.0);
anim.SetConstantPosition (nodes.Get(1), 3.0, 3.0);
anim.UpdateNodeSize (0, 0.2, 0.2);
anim.UpdateNodeSize (1, 0.2, 0.2);
//End of animatin configuration
```

Animation with static movement

```
//Ascii Format Tracing
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll (ascii.CreateFileStream("ptop.tr"));
```

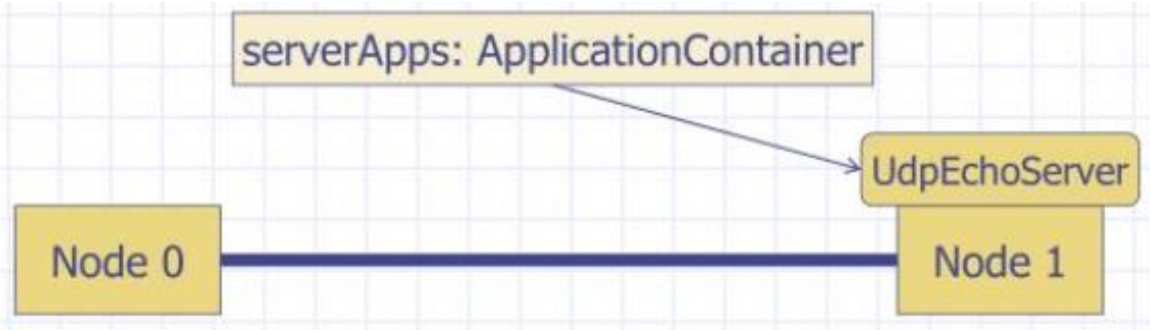
Trace file

```
Simulator::Run ();
Simulator::Destroy ();
```

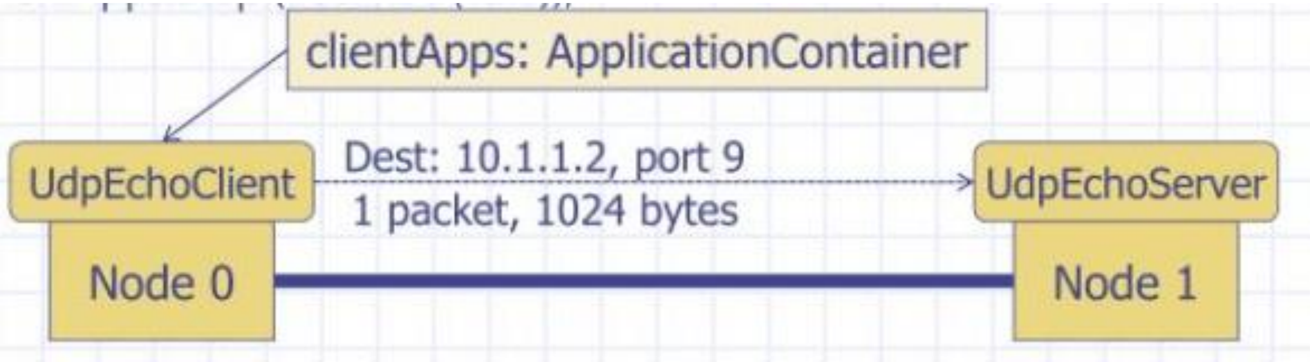
Start simulation

illustrations of ptop

UDP Echo System (Server)

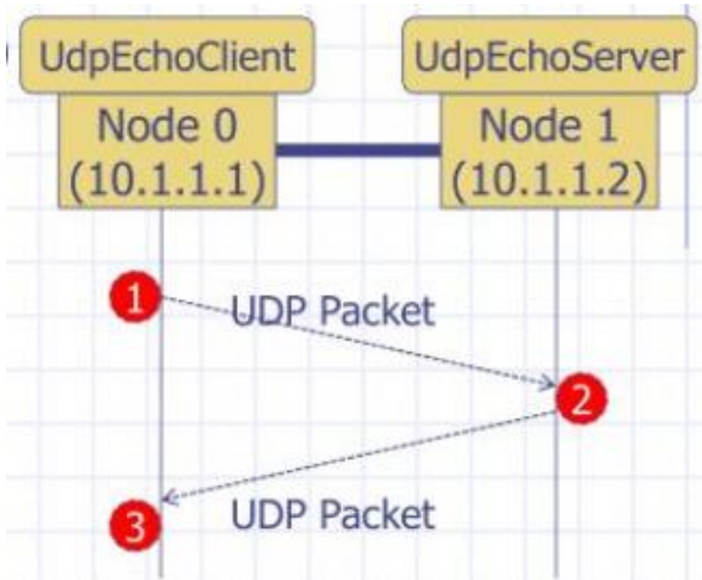


UDP Echo System (Client)



illustrations of ptop

Simulator::Run()

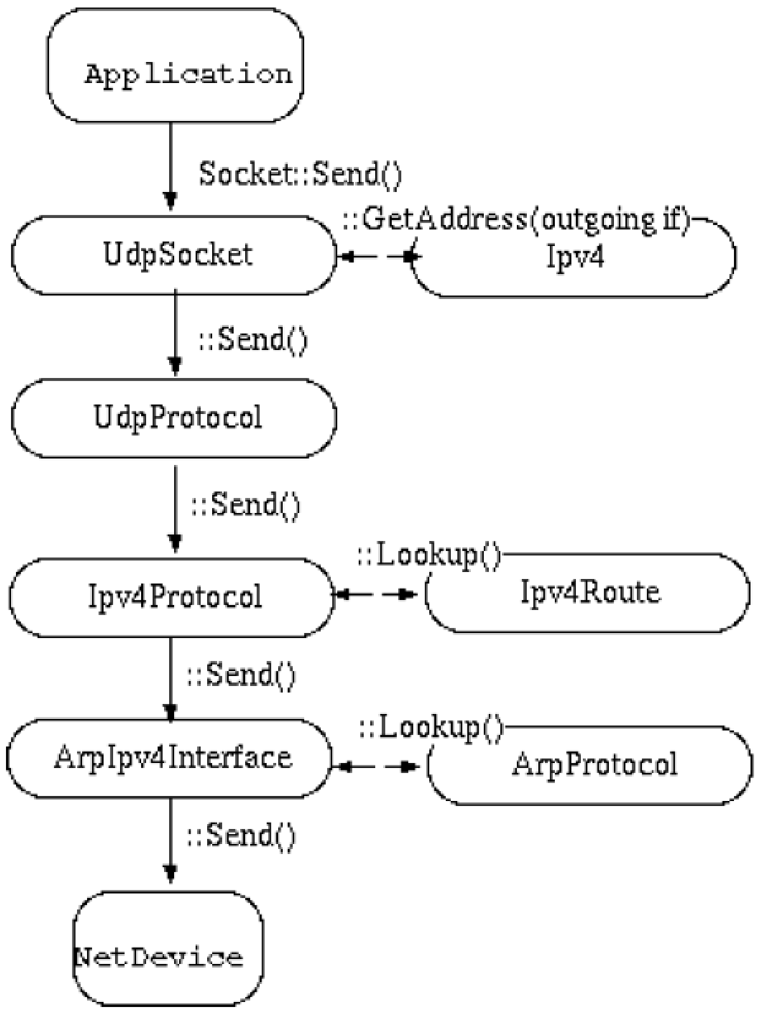



```
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
At time 4s client sent 1024 bytes to 10.1.1.2 port 9
At time 4.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 4.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 4.00737s client received 1024 bytes from 10.1.1.2 port 9
```

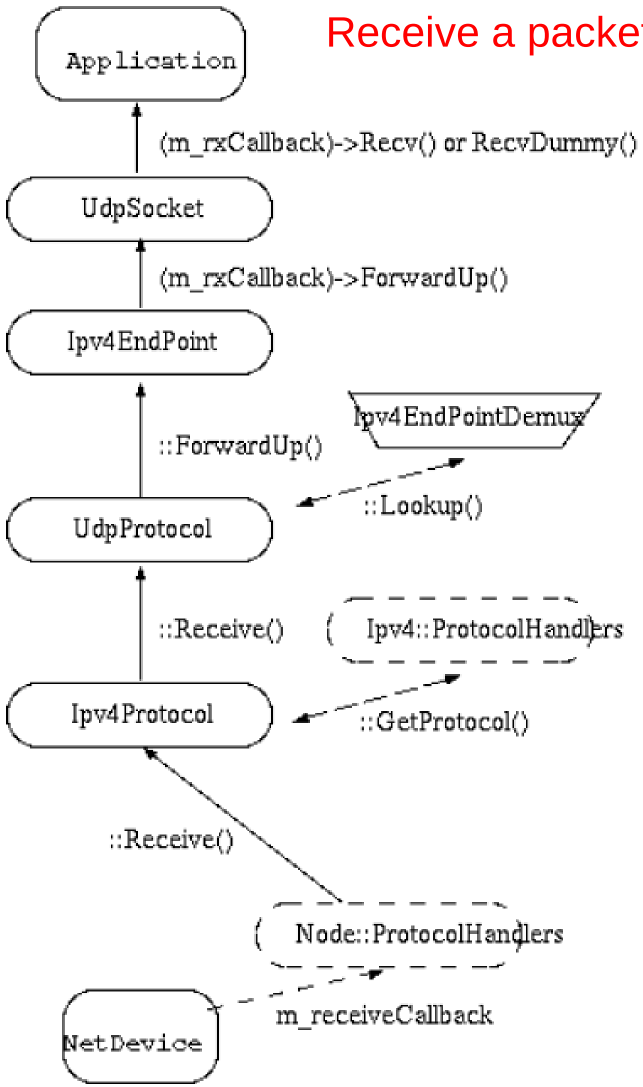
1 2 3

Path of packet (Client - Server)

Send a packet



Receive a packet



Tracing model

- Tracing is a structured form of simulation output
  - Tracing format should be relatively static across simulator releases
- Example (from ns-2):

```
+ 1.84375 0 2 cbr 210 ------- 0 0.0 3.1 225 610
- 1.84375 0 2 cbr 210 ------- 0 0.0 3.1 225 610
r 1.84471 2 1 cbr 210 ------- 1 3.0 1.0 195 600
r 1.84566 2 0 ack 40 ------- 2 3.2 0.1 82 602
+ 1.84566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
```

event	time	from node	to node	pkt type	pkt size	flags	flow_id	src addr	Dst addr	seq num	pkt id

http://www.jgyan.com/ns2/trace%20file.php

Tracing model

- Parsing Ascii traces
  - +: An enqueue operation occurred on the device queue;
  - -: A dequeue operation occurred on the device queue;
  - d: A packet was dropped, typically because the queue was full;
  - r: A packet was received by the net device.

```
+
2
/NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue
ns3::PppHeader (
  Point-to-Point Protocol: IP (0x0021))
  ns3::Ipv4Header (
    tos 0x0 ttl 64 id 0 protocol 17 offset 0 flags [none]
    length: 1052 10.1.1.1 > 10.1.1.2)
    ns3::UdpHeader (
      length: 1032 49153 > 9)
      Payload (size=1024)
```

```
r
2.25732
/NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/MacRx
  ns3::Ipv4Header (
    tos 0x0 ttl 64 id 0 protocol 17 offset 0 flags [none]
    length: 1052 10.1.1.1 > 10.1.1.2)
    ns3::UdpHeader (
      length: 1032 49153 > 9)
      Payload (size=1024)
```

Simulation

- Simulation time moves discretely from event to event
- C++ functions schedule events to occur at specific simulation times
- A simulation scheduler orders the event execution
- Simulation::Run() gets it all started
- Simulation stops at specific time or when events end