Introduction to ns-3

Cesar D. Guerrero, Ph.D.
cguerrer@unab.edu.co

When preparing these slides, I used slides provided by others. Specially, I would like to acknowledge Professor James Sterbenz and Professors Gustavo Carneiro for the slides posted.

Outline

- Network simulation
- Ns-3 simulator
- Tutorial
- Lab

Network simulation

Network analysis techniques

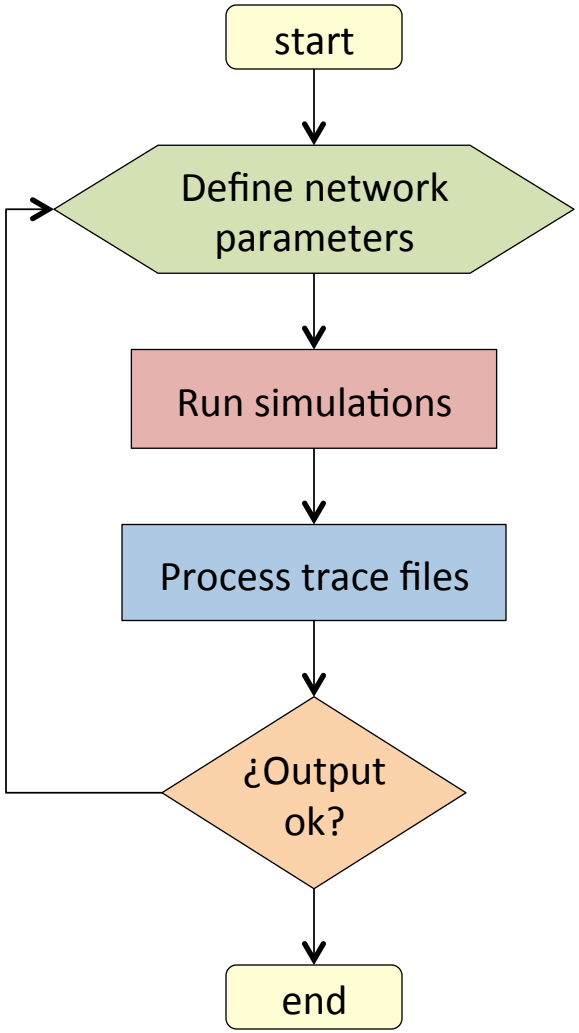
- Analytical analysis
  - Mathematical analysis (eg. Jackson Networks)
- Simulation
  - Model the network using software (eg. ns-3, QualNet, OpNet)
- Emulation
  - Hardware behaving as a real network (eg. AvBw testbed @ UNAB)
- Real network
  - Active or passive (via traces) analysis of a real network

Outline

- Network simulation
- Ns-3 simulator
- Tutorial
- Lab

Ns-3 simulator

Simplified flowchart



- Define topology, nodes, channel, NICs
- Define protocols
- Define initial values

- ./waf --run hello-simulator

- Process output/trace files (eg. wireshark)
- Plot results (eg. gnuplot)

- ¿Are results as expected?
- ¿Confidence level attained?

Ns-3 simulator
Key abstractions (1)

- Node
  - Basic computing device abstraction (host, computer).
  - Represented in C++ by the class **`Node`**
- Application
  - User program running on a **`Node`** that generates some activity to be simulated
  - e.g. UdpEchoClientApplication, UdpEchoServerApplication
  - Represented in C++ by the class **`Application`**
- Channel
  - The medium over which the data flows
  - e.g. CsmaChannel, PointToPointChannel and WifiChannel.
  - Represented in C++ by the class **`Channel`**

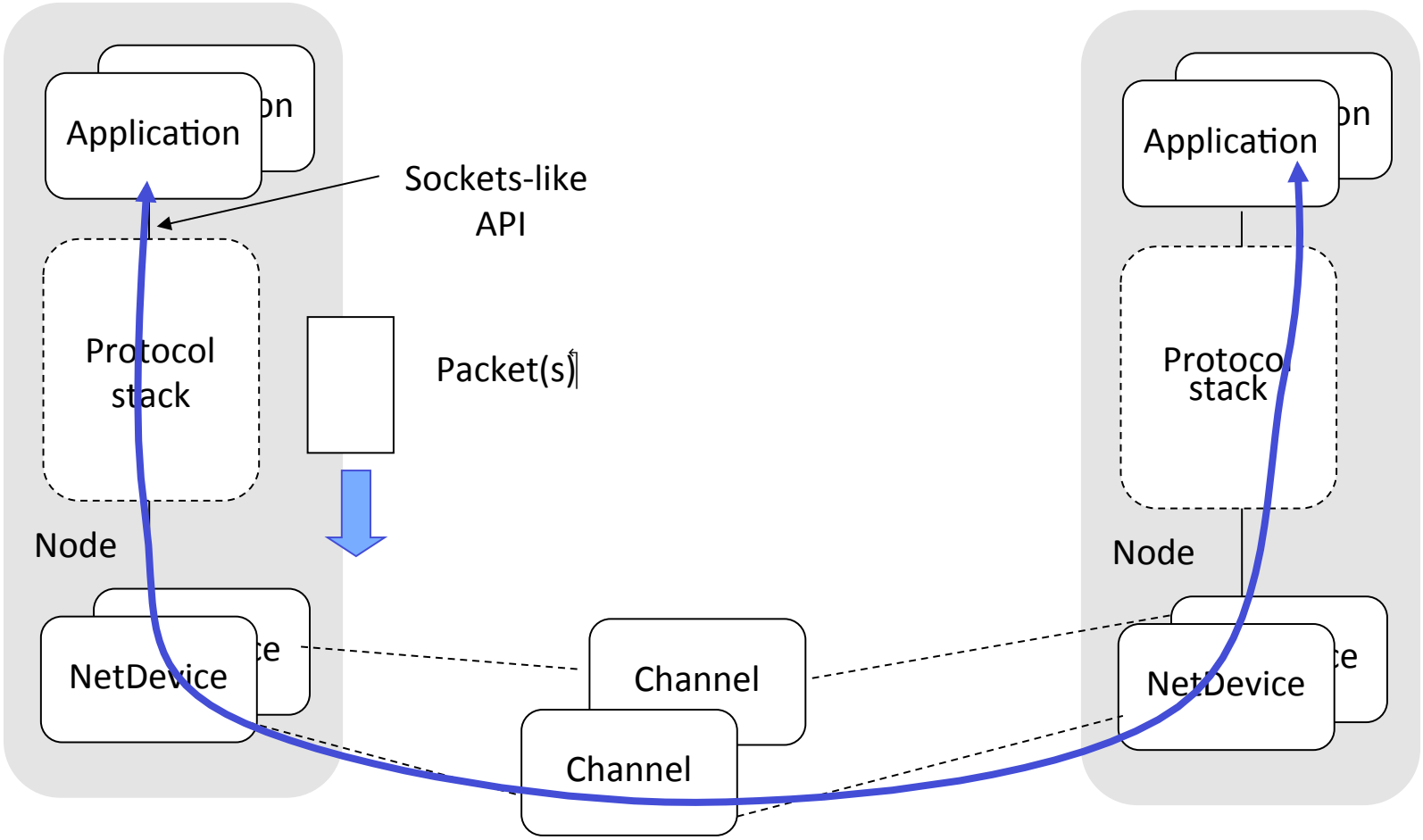
Ns-3 simulator

Key abstractions (2)

- Network Device
  - Like a specific kind of network cable, hardware device (Network Interface Cards), and network device driver.
  - A net device is "installed" in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels.
  - e.g. CsmaNetDevice, PointToPointNetDevice, and WifiNetDevice.
  - Represented in C++ by the class **`NetDevice`**
- Topology Helper
  - NodeContainer: Way to create, manage and access any Node object
  - PointToPointHelper: Configure and connect PointToPointNetDevice and PointToPointChannel objects. Set DataRate on NetDevices and Delay on Channel
  - NetDeviceContainer: To Install NetDevice at the nodes and create Channel between the nodes
  - InternetStackHelper: To Install an Internet Stack (TCP, UDP, IP, etc.) on each of the nodes in the node container
  - Ipv4AddressHelper: Manage the allocation of IP address and assign addresss to the devices using Ipv4InterfaceContainer

Ns-3 simulator

Basic simulation model



Ns-3 simulator

c++ script main structure

- main function: declare main function
- topology helpers: objects to combine distinct operations
- applications: on/off application, UdpEchoClient/Server
- tracing: .tr and/or .pcap files
- simulator: start/end simulator, cleanup

Ns-3 simulator

Running simulations using *waf*

- waf is a general purpose build system to:
  - Configure
  - Compile
  - Install
- Similar to ./configure; make; make install
  - Just type ./waf
- Python based
- More information
  - http://code.google.com/p/waf/
  - http://freehackers.org/~tnagy/wafbook154/index.html

Ns-3 simulator

Post processing

- Once the simulations are over process trace files
- Trace files can be filtered via a script
  - e.g. Python, Perl, Awk
- Filtered results can be processed via a plotting tool
  - e.g. gnuplot
- Output files in .pcap format is possible
  - Wireshark or tcpdump can be used to view .pcap files
- Logs can be enabled to analyze output
- ns-3 package built-in tools for post-processing
  - flow monitor
    - http://telecom.inescn.pt/~gjc/flowmon-presentation.pdf

Outline

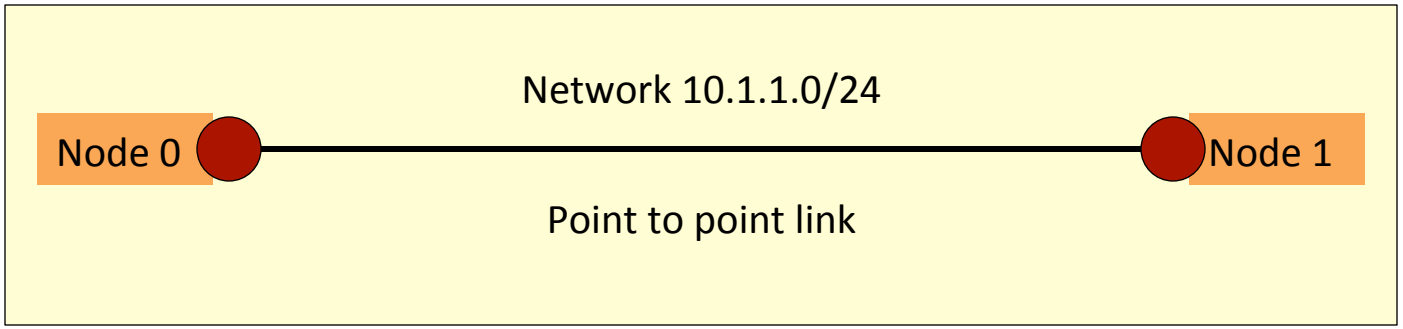
- Network simulation
- Ns-3 simulator
- Tutorial
- Lab

Tutorial
Workspace

- ~/tarballs/ns-allinone-3.12.1/ns-3.12.1
  - Run programs only from the scratch folder
- Run program by the commands below
  - ./waf --run scratch/example
  - (or) ./waf --run example

Tutorial

examples/tutorial/first.cc (1)



- Two nodes, one NIC per node
- Point to point link
  - Transmission delay: 2ms, Link capacity: 5Mbps
- Application
  - UdpEchoClient on Node 0, UdpEchoServer on Node 1
  - Payload size of 1024-byte packet
  - Time interval between packets is 1 s

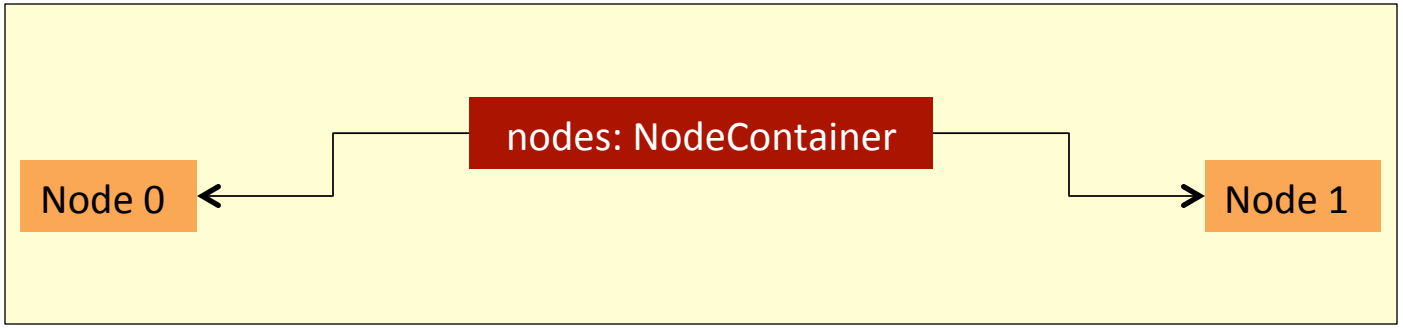
Tutorial

examples/tutorial/first.cc (2)

```
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);

  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;
}
```

Tutorial

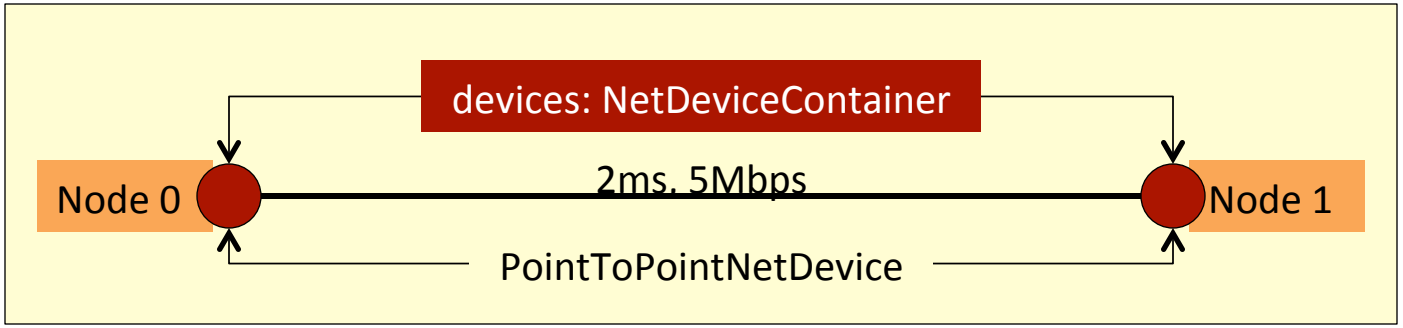
examples/tutorial/first.cc (3)




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

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

Tutorial

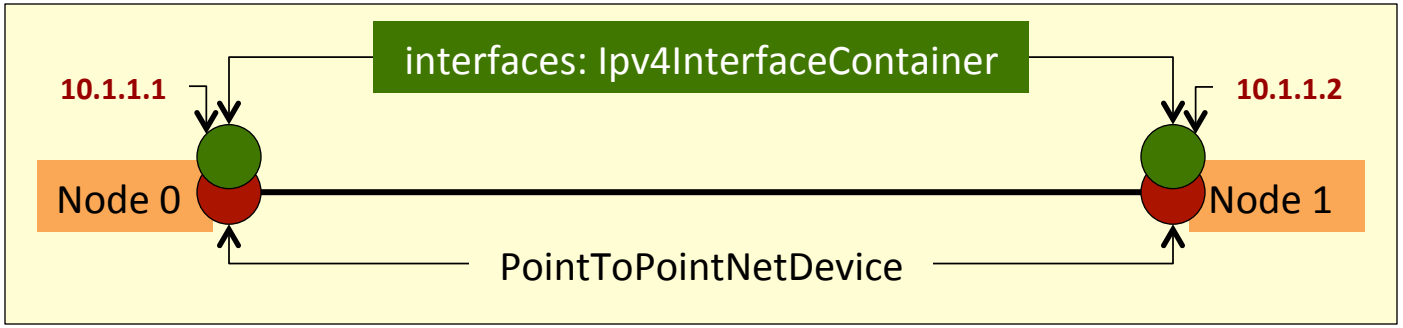
examples/tutorial/first.cc (4)




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

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

Tutorial

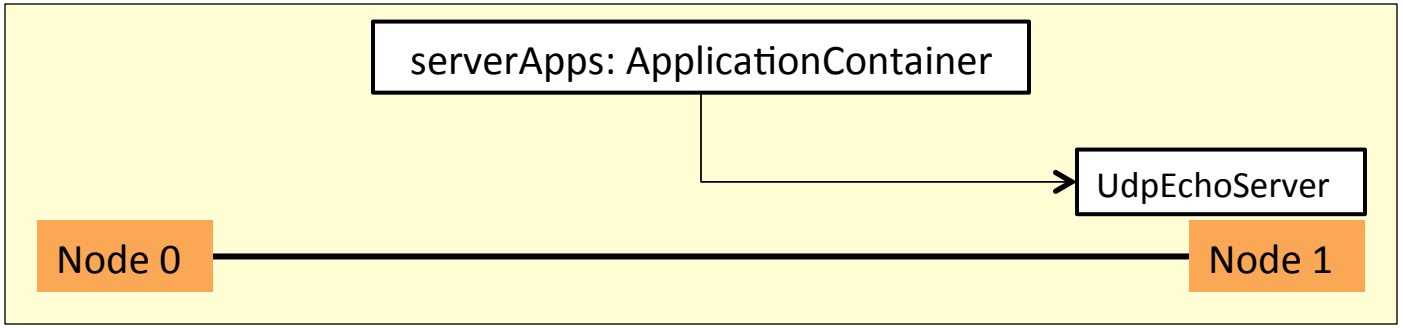
examples/tutorial/first.cc (5)




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

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

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

Tutorial

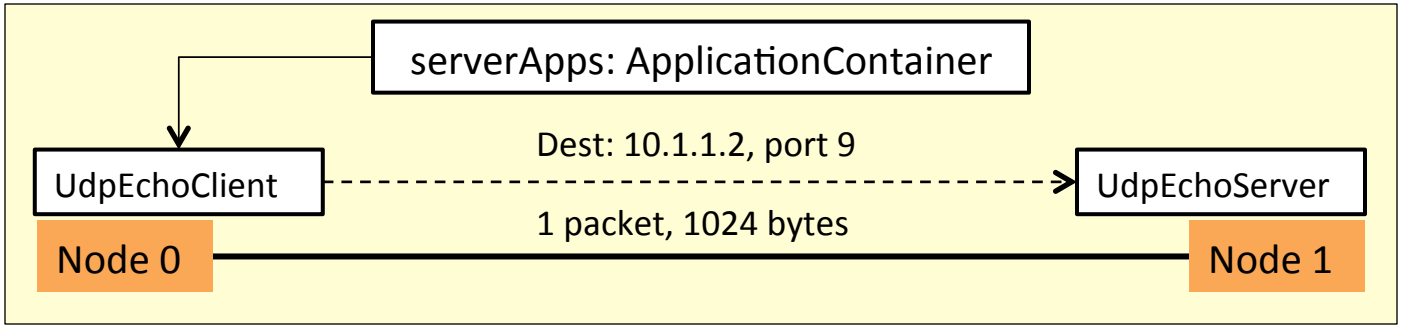
examples/tutorial/first.cc (6)




```
UdpEchoServerHelper echoServer (9);

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

Tutorial

examples/tutorial/first.cc (7)

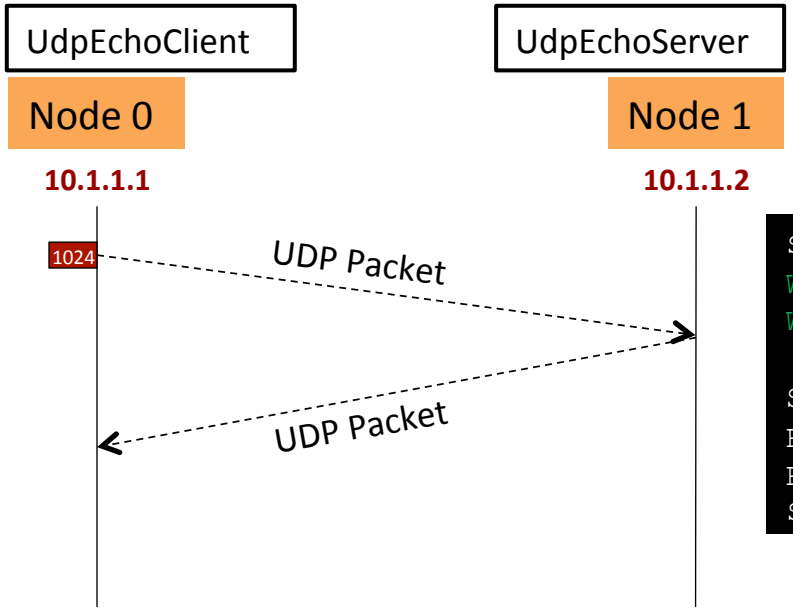



```
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

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

Tutorial

examples/tutorial/first.cc (8)

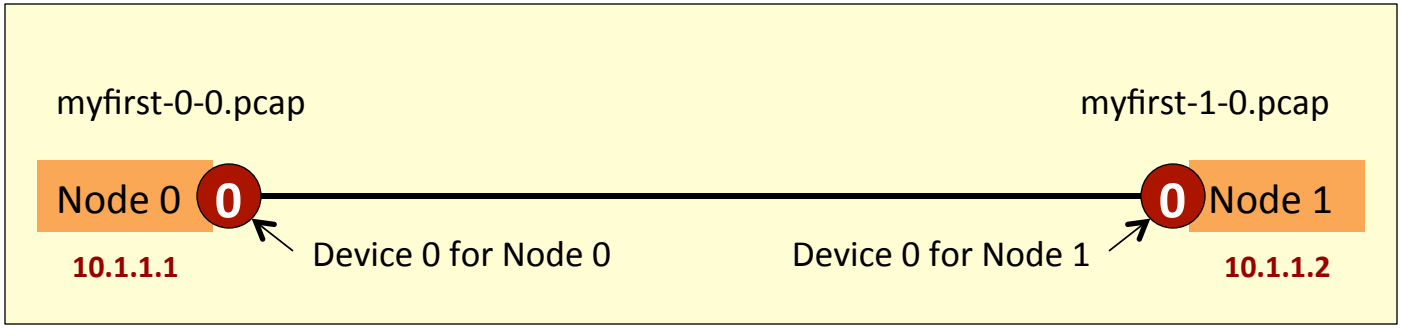
```
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```




```
$ ./waf -run first
Waf: Entering directory `/home/…/build'
Waf: Leaving directory `/home/…/build'
'build' finished successfully (1.833s)
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2
$
```

Tutorial
Enabling ASCII/PCAP Post-processing

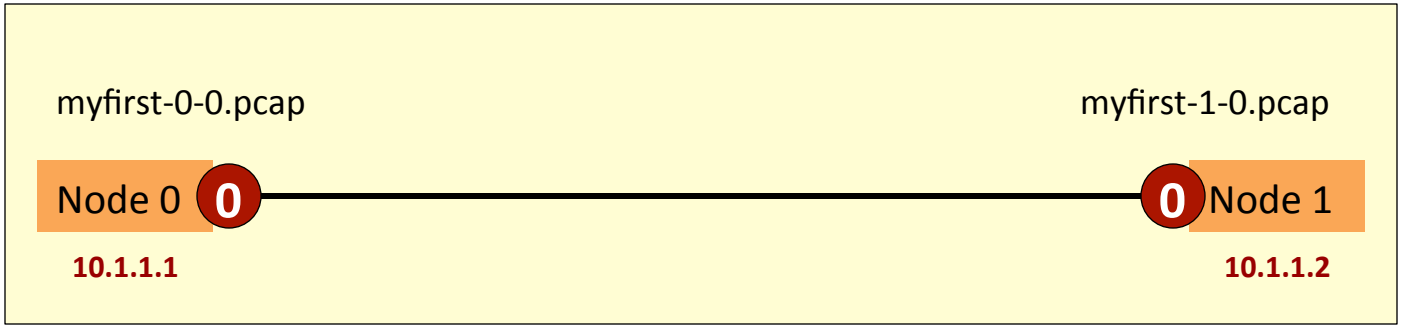
- Copy `examples/tutorial/first.cc` to `./scratch/myfirst.cc`
- To enable ASCII tracing:
  - Include to the top of the script: `#include <fstream>`
  - Then add right before the call to Simulator::Run ():
    - `AsciiTraceHelper ascii;`
    - `pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("myfirst.tr"));`
- To enable pcap tracing a one-line is needed just right before the call to Simulator::Run ():
  - `pointToPoint.EnablePcapAll ("myfirst");`
- Files named “myfirst-0-0.pcap” and “myfirst.1-0.pcap” are the pcap traces for node 0-device 0 and node 1-device 0, respectively.



Tutorial

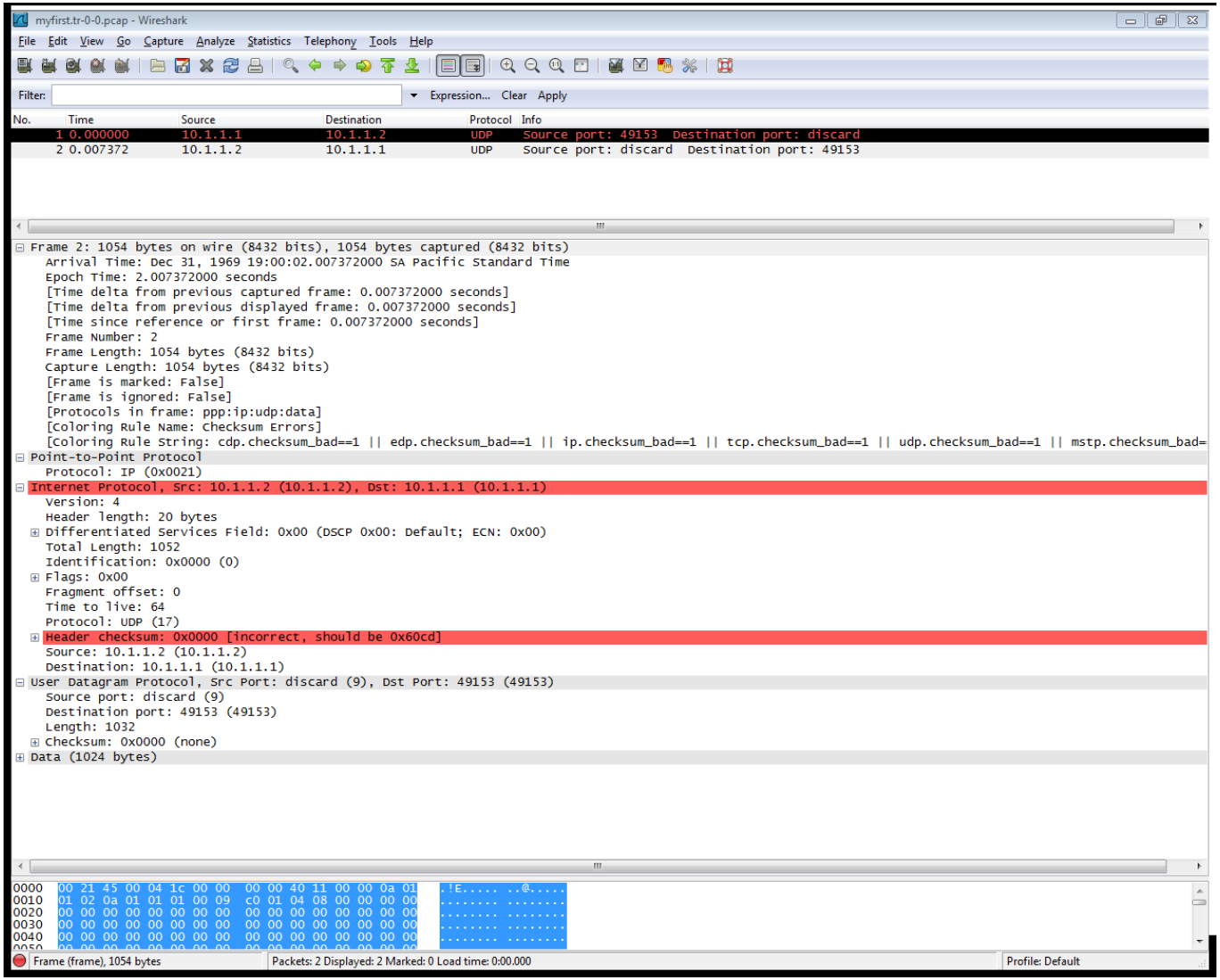
Reading output with *tcpdump*

```
$ tcpdump -nn -tt -r myfirst-0-0.pcap
reading from file myfirst-0-0.pcap, link-type PPP (PPP)
2.000000 IP 10.1.1.1.49153 > 10.1.1.2.9: UDP, length 1024
2.007372 IP 10.1.1.2.9 > 10.1.1.1.49153: UDP, length 1024

$ tcpdump -nn -tt -r myfirst-1-0.pcap
reading from file myfirst-1-0.pcap, link-type PPP (PPP)
2.003686 IP 10.1.1.1.49153 > 10.1.1.2.9: UDP, length 1024
2.003686 IP 10.1.1.2.9 > 10.1.1.1.49153: UDP, length 1024
```



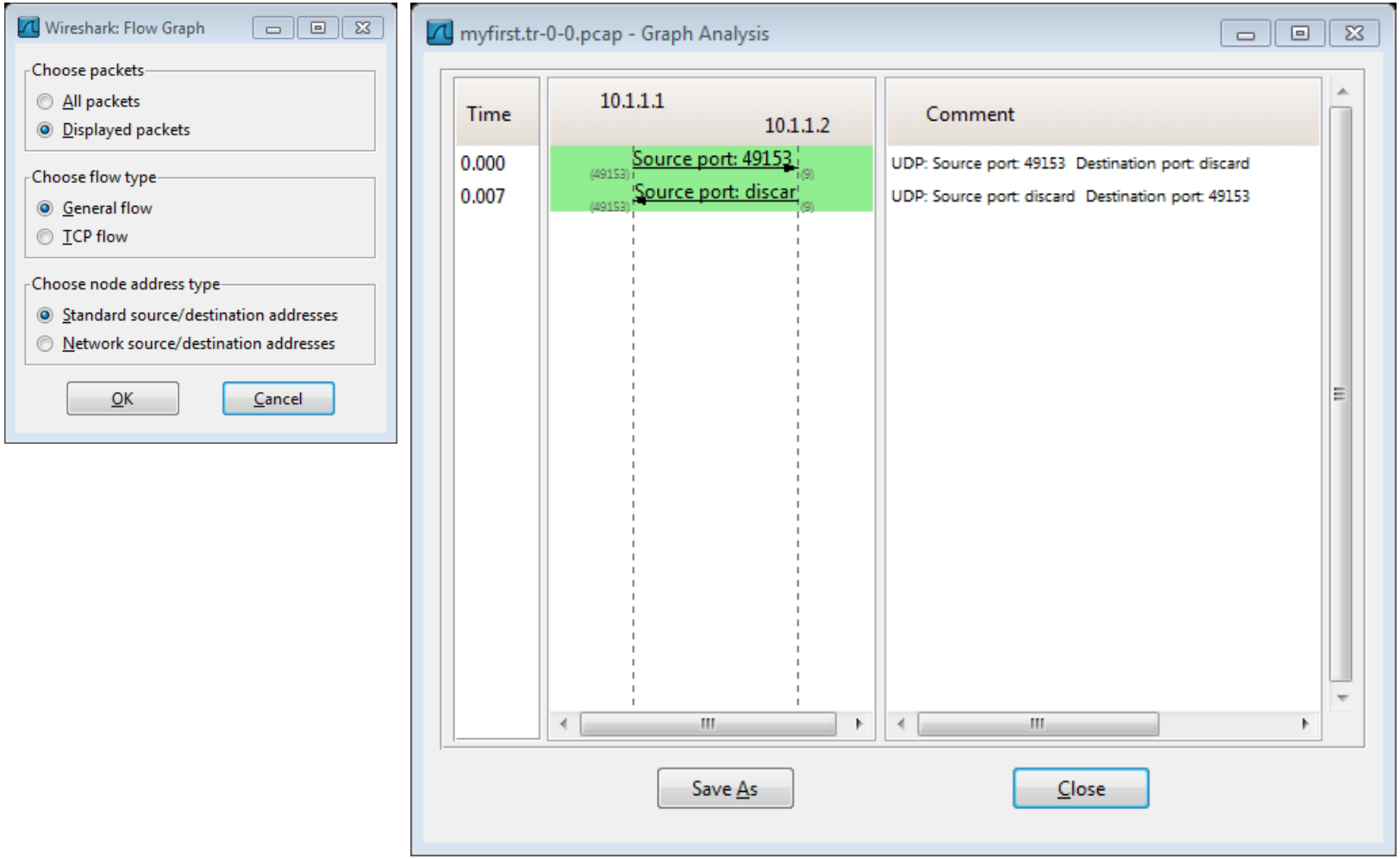
Tutorial

Reading output with *Wireshark* (myfirst-0-0.pcap)



Tutorial

Flow diagram with *Wireshark* (myfirst-0-0.pcap)



Outline

- Network simulation
- Ns-3 simulator
- Tutorial
- Lab

Lab

ns3-intro-lab

Ns-3 Intro Lab[1]

In this lab, you will become familiar with ns-3 simulations based on the example given in class.

1. Log into your ns3 account.

2. Create a new simulation file (<your name>_first.cc) with the following network specification:
   - Two nodes having one interface card each
   - Point to point link
     - 1 ms transmission delay
     - 1 Mbps link capacity
   - IP address assignment
     - 192.168.10.0/24
   - Application
     - UdpEchoServer on port 53
     - Packet size: 512 bytes
   - Rest of the attribute values are same as in myfirst.cc
   - Add a header comment to your file:

```
//GNU release …..
/* File name: <your name>_first.cc
Purpose: This is a ns3 simulation of …
Author: <your full name>
Date: dd-mmm-yy
Version: 1 */
#include <iostream.h>
```