

# ns3 -Tracing System

IWING Team, Kasetsart University

# Tracing Overview

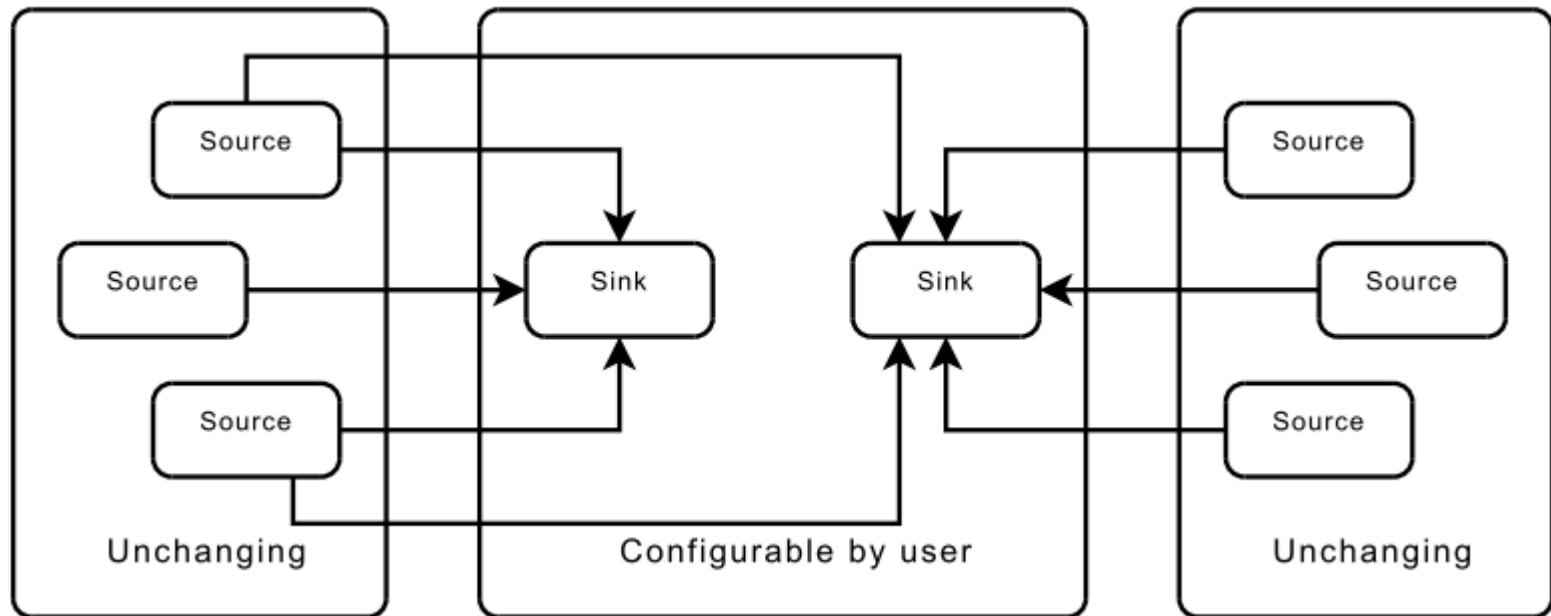
---

- ▶ ns-3 provides a set of pre-configured trace sources
- ▶ Users provide trace sinks and attach to the trace source
- ▶ Multiple trace sources can connect to a trace sink

# ns-3 Tracing Model

---

- ▶ Decouple trace sources from trace sinks:



# ns-3 Trace Sources

NS-3 - Mozilla Firefox

file:///home/cpj/install/ns-allinone-3.10/ns-3.10/doc/html/index.html

Most Visited Hits Deals Dicts Netstat Refs Theses Java API BashPitfalls - Greg's... CS Profs Debate Rol... Khan Academy

NS-3

**NS-3**

- ns-3 Documentation
- Modules
  - Simulator
  - Core
    - The list of all trace sources.**
    - The list of all global values.
  - Debugging
  - Attribute Helper
  - Attribute
  - MakeCallback
  - Object
  - Smart Pointer
  - Random Variable Distributions
  - Tracing
- Common
- Node
- Devices
- InternetStack

Main Page Modules Namespaces Classes Files

## The list of all trace sources. [Core]

Collaboration diagram for The list of all trace sources.:

```
graph LR; A[Core] --> B[The list of all trace sources.]
```

**ns3::LteMacQueue**

- Enqueue: Enqueue trace
- Dequeue: Dequeue trace
- Drop: Drop trace

**ns3::LteSpectrumPhy**

- TxStart: Trace fired when a new transmission is started
- TxEnd: Trace fired when a previously started transmission is finished
- RxStart: Trace fired when the start of a signal is detected
- RxAbort: Trace fired when a previously started RX is aborted before time
- RxEndOk: Trace fired when a previously started RX terminates successfully
- RxEndError: Trace fired when a previously started RX terminates with an error (packet is corrupted)

Done 1921 zotero

# Multiple Levels of Tracing

---

- ▶ High-level
  - ▶ Use a helper to hook a predefined trace source to an existing trace sink (e.g., ascii, pcap)
- ▶ Mid-level
  - ▶ Hook an existing trace source to a custom trace sink
- ▶ Low-level
  - ▶ Add a new trace source and connect it to a special trace sink

# Trace Helpers

---

- ▶ Ascii Trace Helper
- ▶ Pcap Trace Helper

# Custom Trace Sink

---

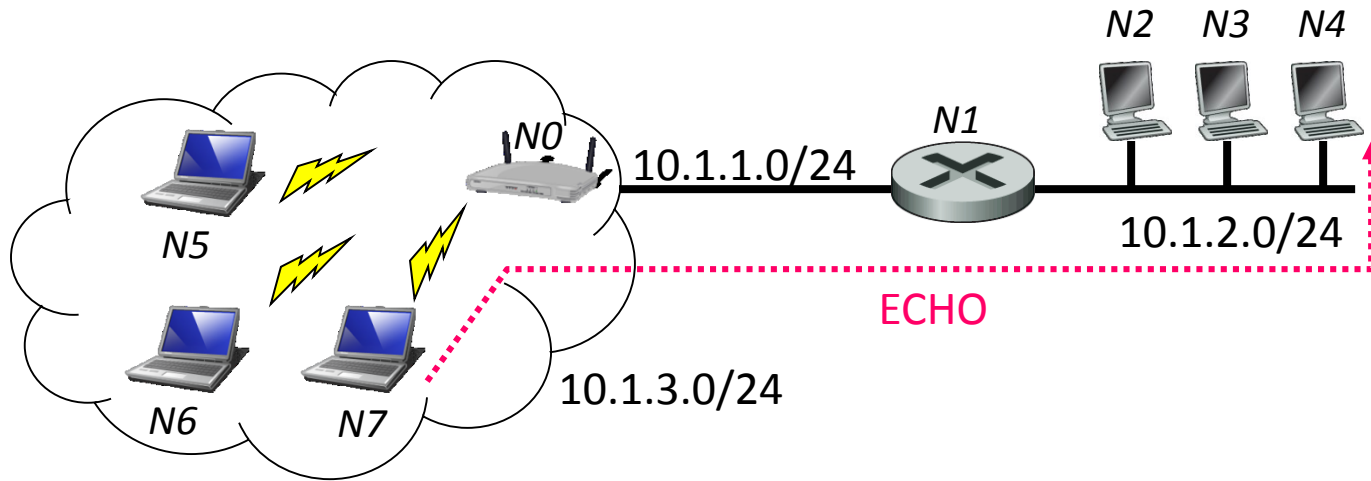
```
void
DevTxTrace(
    std::string context,
    Ptr<const Packet> p, Mac48Address address)
{
    std::cout << " TX to=" << address << " p: " << *p << std::endl;
}

:

Config::Connect(
    "/NodeList/*/DeviceList/*/Mac/MacTx",
    MakeCallback(&DevTxTrace));
```

# Walk-Through Example I (1)

- ▶ Task: model the network topology below in ns-3 and capture ECHO packets transmitted from N7 to N4



- ▶ We can start from [tutorials/third.cc](http://tutorials.third.cc)



# Walk-Through Example I (2)

---

- ▶ Create a copy of third.cc into the scratch dir and rename it to pm-ex1.cc

```
$ cp examples/tutorials/third.cc scratch/pm-ex1.cc
```

- ▶ Edit the source

- ▶ Change port number from 9 to 7

```
UdpEchoServerHelper echoServer (7);  
:  
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 7);
```

- ▶ Change PCAP file prefix to pm-ex1

```
pointToPoint.EnablePcapAll ("pm-ex1");  
phy.EnablePcap ("pm-ex1", apDevices.Get (0));  
csma.EnablePcap ("pm-ex1", csmaDevices.Get (0), true);
```

# Walk-Through Example I (3)

- ▶ Run the script

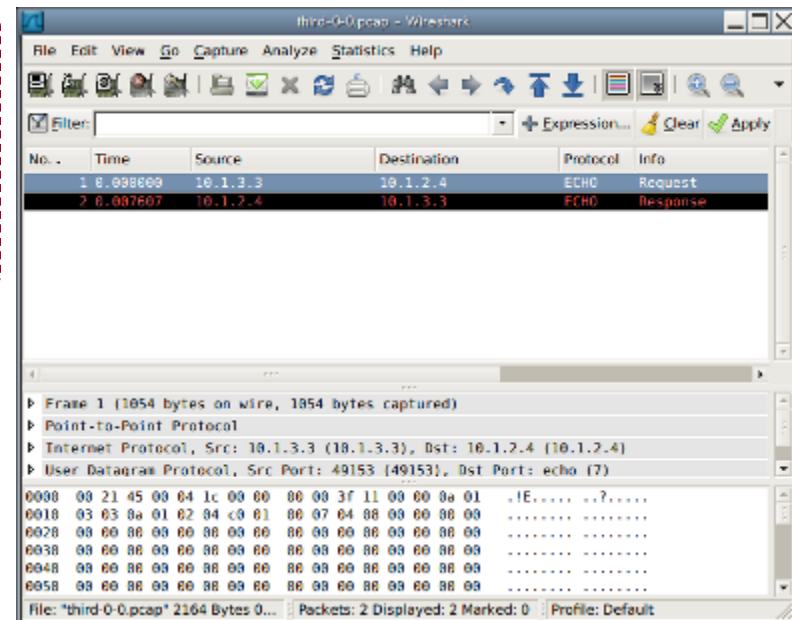
```
$ ./waf --run pm-ex1
```

- ▶ Open the pcap files with Wireshark

```
$ ls *.pcap
```

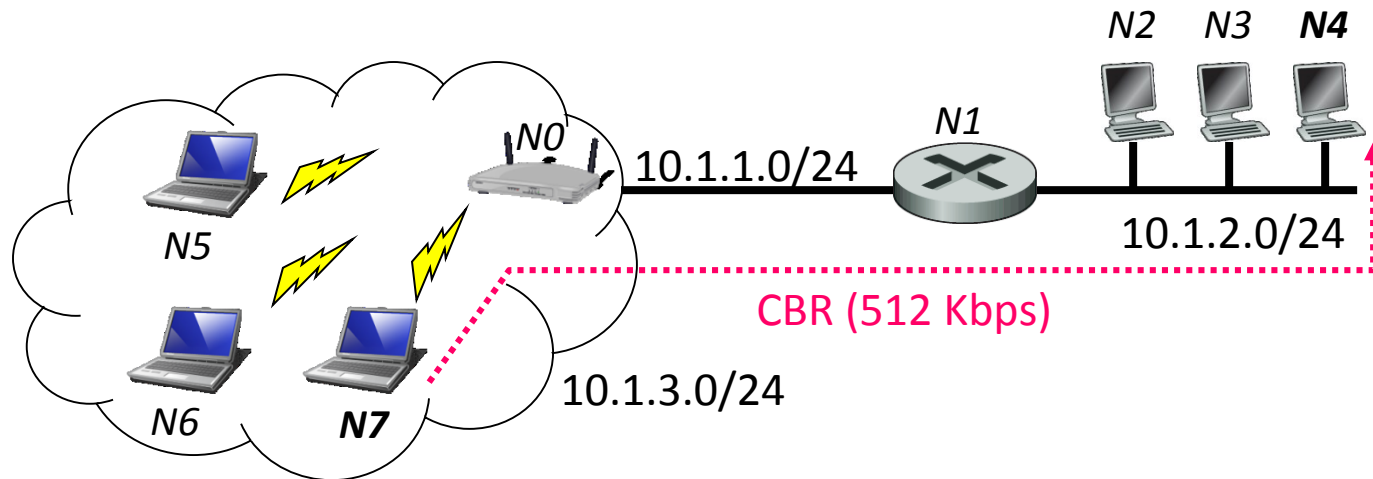
```
pm-ex1-0-0.pcap  pm-ex1-0-1.pcap  
pm-ex1-1-0.pcap  pm-ex1-1-1.pcap
```

```
$ wireshark pm-ex1-0-0.pcap
```



# Walk-Through Example II (1)

- ▶ Task: Using the previous topology, replace ECHO traffic with CBR (Constant Bit Rate) traffic



# Walk-Through Example II (2)

---

- ▶ Create pm-ex2.cc from pm-ex1.cc

```
$ cp scratch/pm-ex1.cc scratch/pm-ex2.cc
```

- ▶ Use OnOffHelper to generate CBR traffic from the client
- 

```
uint16_t port = 9;
OnOffHelper onoff(
    "ns3::UdpSocketFactory",
    InetSocketAddress("10.1.2.4", port));
onoff.SetAttribute("OnTime",StringValue("ns3::ConstantRandomVariable[Constant=1]"));
onoff.SetAttribute("OffTime",StringValue("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute("DataRate", StringValue("512Kbps"));
onoff.SetAttribute("PacketSize", StringValue("512"));
onoff.Install(wifiStaNodes.Get(nWifi - 1));
ApplicationContainer apps = onoff.Install(wifiStaNodes.Get(nWifi-1));
apps.Start(Seconds(5.0));
apps.Stop(Seconds(20.0));
```

---

# Walk-Through Example II (3)

---

- ▶ Create a packet sink on the server

```
PacketSinkHelper sink(  
    "ns3::UdpSocketFactory",  
    InetSocketAddress("10.1.2.4", port));  
sink.Install(csmaNodes.Get(nCsma));
```

- ▶ Set simulation time to 30 seconds

```
Simulator::Stop(Seconds(25.0));
```

- ▶ Don't forget to change the PCAP trace prefix

# Walk-Through Example III (1)

---

- ▶ Task: calculate average throughput from previous scenario
- ▶ Idea:
  - ▶ Connect a custom trace sink to PacketSink's Rx trace source
  - ▶ Compute average throughput from
    - ▶ first and last packets' timestamps
    - ▶ total bytes received