Multimedia & Wireless Networking Laboratory

Session 3. High-Level Tracing 및 ns-3 기본 제공 어플리케이션

최준영

Multimedia & Wireless Networking Laboratory, SNU

jychoi@mwnl.snu.ac.kr

Contents

- Tracing system
  - PCAP tracing
- Applications in ns-3
  - UdpEcho
  - OnOffApplication
  - UdpClientServer
  - PacketSink
- Simulation example
  - UdpEcho example
  - OnOffApplication example
- Exercise

Multimedia & Wireless Networking Laboratory

Tracing System

Tracing System

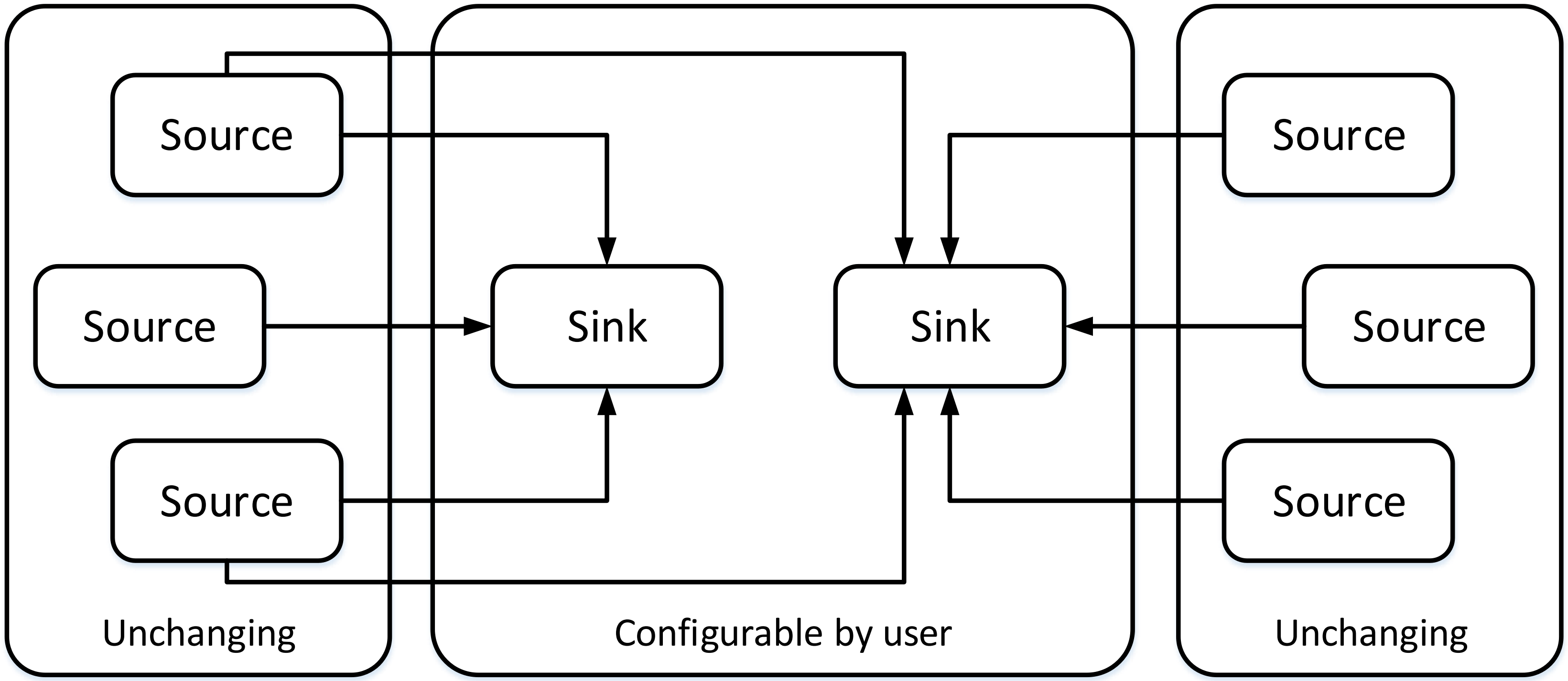
- Generating output after running a simulation
  - Checking the operation
  - Measuring the system performance
- Advantage of tracing
  - Reaching into the core system and getting required information without changing or recompiling
- Independent tracing sources and tracing sinks along with a uniform mechanism for connecting sources to sinks

Tracing System

- Trace source
  - Signaling events and providing access to interesting data
- Trace sink
  - Consuming trace information
- ns-3 provides a set of pre-configured trace sources
- Users provide trace sinks and attach to the trace source

Tracing System

- Multiple trace sources can be connected to a trace sink



Tracing System

- High-level tracing
  - Using a *trace helper* to hook a predefined trace source to an existing trace sink
  - No special step to create trace sink and connect trace source and sink manually
  - PCAP (Packet capture)
- Low-level tracing
  - Connecting trace source(s) to a custom trace sink manually

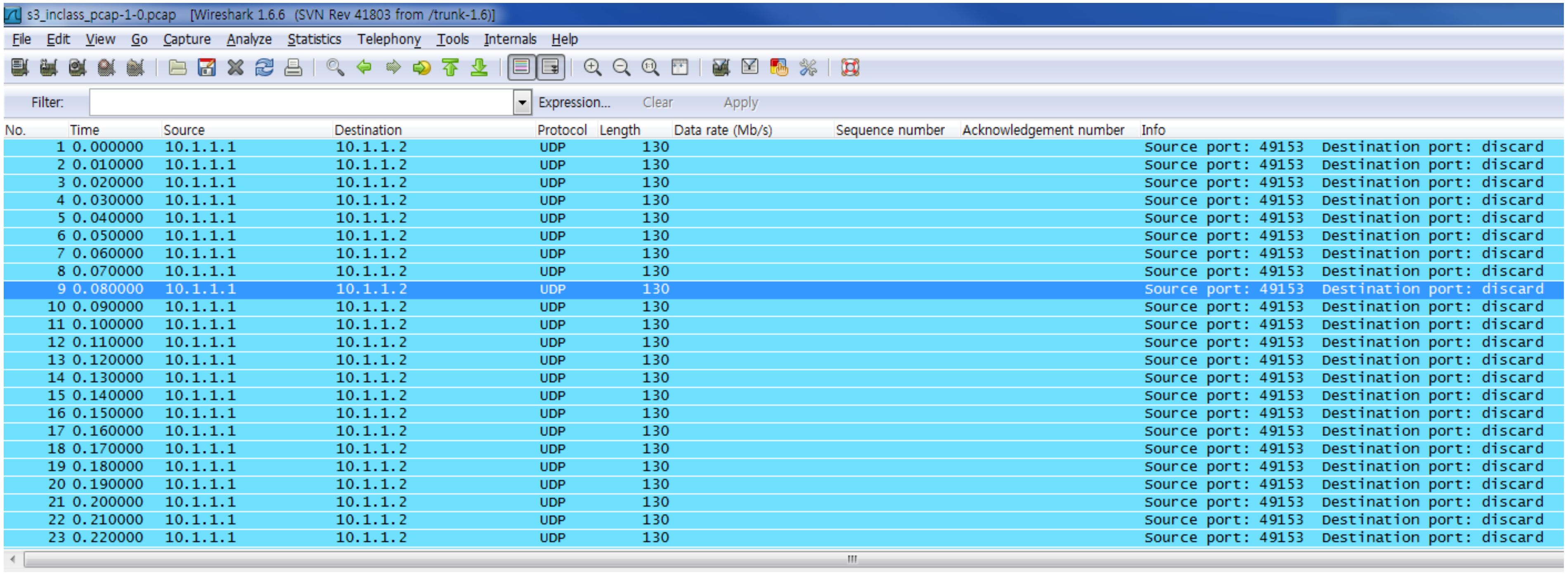
PCAP Tracing

- PCAP tracing
  - .pcap file format
  - Traffic trace analyze
  - pointToPoint.EnablePcapAll (“example”);
- Reading output using tcpdump

```
$ tcpdump -nn -tt -r s3_inclass_pcap-0-0.pcap
reading from file s3_inclass_pcap-0-0.pcap, link-type PPP (PPP)
1.000000 IP 10.1.1.1.49153 > 10.1.1.2.9: UDP, length 100
1.010000 IP 10.1.1.1.49153 > 10.1.1.2.9: UDP, length 100
```

PCAP Tracing

- Reading output using Wireshark
- http://www.wireshark.org/download.html


s3_inclass_pcap-1-0.pcap [Wireshark 1.6.6 (SVN Rev 41803 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

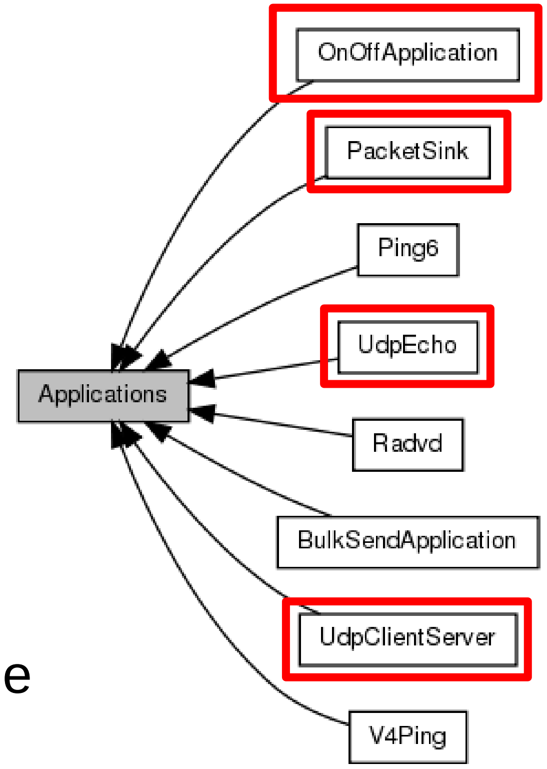
No.	Time	Source	Destination	Protocol	Length	Data rate (Mb/s)	Sequence number	Acknowledgement number	Info
1	0.000000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
2	0.010000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
3	0.020000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
4	0.030000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
5	0.040000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
6	0.050000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
7	0.060000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
8	0.070000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
9	0.080000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
10	0.090000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
11	0.100000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
12	0.110000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
13	0.120000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
14	0.130000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
15	0.140000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
16	0.150000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
17	0.160000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
18	0.170000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
19	0.180000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
20	0.190000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
21	0.200000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
22	0.210000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard
23	0.220000	10.1.1.1	10.1.1.2	UDP	130				Source port: 49153 Destination port: discard

Multimedia & Wireless Networking Laboratory

Applications in ns-3

Applications in ns-3

- An Application generates packets
- Applications are associated with individual nodes
- Class **ns3::Application** can be used as a base class for **ns3** applications
- Various application modules are provided by **ns3**
- Application **Helper**
  - Can make it easy to install application to node
  - Set attributes



Applications in ns-3

- **OnOffApplication (OnOffHelper)**
  - During the “On” state, constant bit rate (CBR) traffic generated
- **PacketSink (PacketSinkHelper)**
  - Receive and consume packets
- **UdpEcho**
  - UdpEchoClient / UdpEchoServer / UdpEchoClient(Server)Helper
  - A client sends packets to a server and the server returns packets
- **UdpClientServer**
  - A UDP client and server

OnOffApplication

- Class ns::OnOffApplication
  - Generate traffic to a single destination according to an OnOff pattern
  - After Application::StartApplication is called, “On” and “Off” states alternate
  - onTime and offTime variables represent duration of each state
  - “Off” state: no traffic, “On” state: CBR (constant bit rate) traffic
  - The attributes, **data rate** and **packet size**, characterize CBR traffic
- Class ns::OnOffHelper
- Examples
  - onoff.SetAttribute ("MaxPackets", UintegerValue (**100**));
  - onoff.SetAttribute (“DataRate”, StringValue (“**14kb/s**”));

Attributes for OnOffApplication

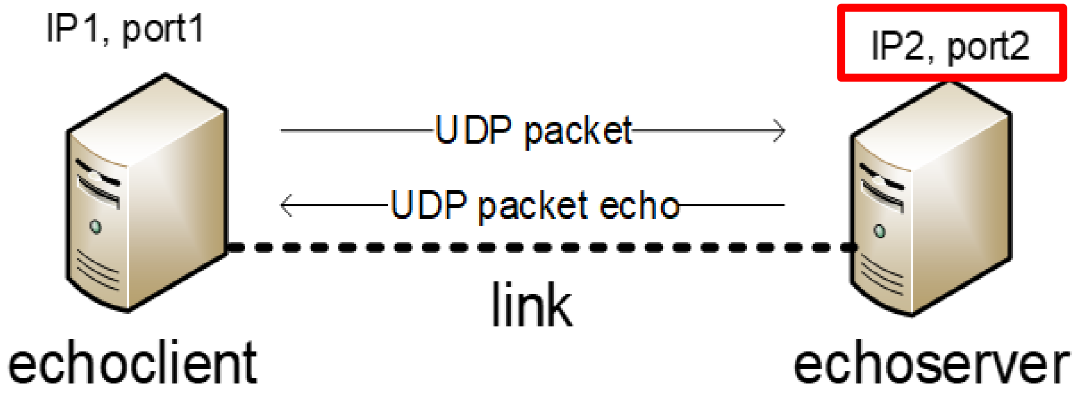
- **DataRate**: The data rate in on state.
  - Set with class: **DataRateValue**
  - Underlying type: **DataRate**
  - Initial value: 500000 bps
- **PacketSize**: The size of packets sent in on state
  - Set with class: **ns3::UintegerValue**
  - Underlying type: **uint32_t** (1 to 4294967295)
  - Initial value: 512
- **OnTime (OffTime)**: The duration of the 'on (off)' state.
  - Set with class: **RandomVariableValue**
  - Underlying type: **RandomVariable**
  - Initial value: Constant:1
- **Link**
  - https://www.nsnam.org/doxygen/classns3_1_1_on_off_application.html

PacketSink

- **PacketSink**
  - Receive and consume traffic
  - Attributes
    - **Protocol**: The type id of the protocol to use for the rx socket
    - **Local**: The Address on which to bind the rx socket
- **PacketSinkHelper**
  - ns3::PacketSinkHelper::PacketSinkHelper (**std::string protocol**, **Address address**)
- Example
  - PacketSinkHelper sink (“**ns3::UdpSocketFactory**”, **Address (InetSocketAddress (Ipv4Address::GetAny (), port))**);

UdpEcho

- **UdpEchoServerHelper**
  - Installed on server
  - UdpEchoServerHelper::UdpEchoServerHelper (uint16_t **port**)
  - **Port**: The port the server will wait on for incoming packets
- **UdpEchoClientHelper**
  - Installed on client
  - UdpEchoClientHelper::UdpEchoClientHelper (Address **IP**, uint16_t **port**)
  - **IP and Port**: IP address and port number of the remote UpdEchoServer



Attributes for UdpEcho

- **MaxPackets**: The maximum number of packets the application will send
  - Set with class: **ns3::UintegerValue**
  - Underlying type: **uint32_t**
- **Interval**: The time to wait between packets
  - Set with class: **ns3::TimeValue**
  - Underlying type: **Time**
- **PacketSize**: Size of echo data in outbound packets
  - Set with class: **ns3::UintegerValue**
  - Underlying type: **uint32_t**
- Example
  - echoClient.SetAttribute ("MaxPackets", **UintegerValue (1)**);
  - echoClient.SetAttribute ("Interval", **TimeValue (Seconds (1.))**);
  - echoClient.SetAttribute ("PacketSize", **UintegerValue (1024)**);

UdpClientServer

- **Applications**
  - UdpClient
  - UdpServer
- **Application Helpers**
  - UdpClientHelper
  - UdpServerHelper

Attributes for UdpClient

- **Interval**: The time to wait between packets
  - Set with class: **ns3::TimeValue**
  - Underlying type: **Time**
- **RemoteAddress**: The destination **Address** of the outbound packets
  - Set with class: **AddressValue**
  - Underlying type: **Address**
- **RemotePort**: The destination port of the outbound packets
  - Set with class: **ns3::UintegerValue**
  - Underlying type: **uint16_t**

Attributes for UdpServer

- **Port**
  - Port on which we listen for incoming packets.
  - Set with class: **ns3::UintegerValue**
  - Underlying type: **uint16_t**
- **PacketWindowSize**
  - The size of the window used to compute the packet loss. This value should be a multiple of 8.
  - Set with class: **ns3::UintegerValue**
  - Underlying type: **uint16_t** (8 to 256)

Functions for Application

- void Application::SetStartTime(Time time);
  - Specifies when the application should be started
  - The application subclasses should override the private StartApplication method, which is called at the time specified, to cause the application to begin
- void ApplicationContainer::Start(Time start);
  - Call Application::SetStartTime method for all of applications in the application container

Functions for Application

- void Application::SetStopTime(Time time);
  - Specifies when an application is to stop
  - The application subclasses should override the private StopApplication method, to be notified when that time has come
- void ApplicationContainer::Stop(Time stop);
  - Call Application::SetStopTime method for all of applications in the application container

Functions for Application

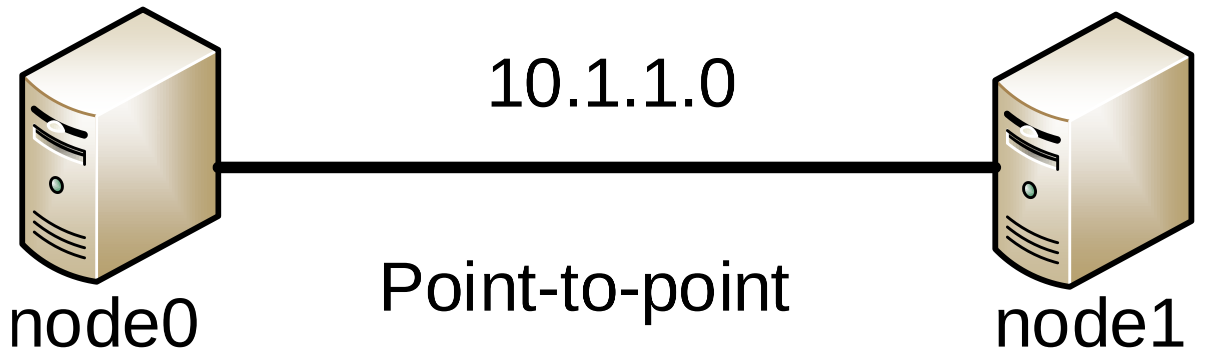
- virtual void Application::StartApplication(void);
  - The StartApplication method is called at the start time specified by SetStartTime
  - This method should be overridden by all or most application subclasses
- virtual void Application::StopApplication(void);
  - The StopApplication method is called at the stop time specified by SetStopTime
  - This method should be overridden by all or most application subclasses

Multimedia & Wireless Networking Laboratory

Simulation Example

Simulation Example

- P2P link
- UDP packet transmission
  - UdpEcho
  - OnOffApplication



UdpEcho Example (1)

```
// s3_ex1.cc
#include "ns3/core-module.h“
…

using namespace ns3;

NS_LOG_COMPONENT_DEFINE (“UdpEchoExample");

int main (int argc, char *argv[])
{
```

To determine whether the logging components are enabled or not

```
  bool verbose = true;
```

command line argument for variable “verbose”

```
  commandLine cmd;
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.Parse (argc,argv);
```

UdpEcho Example (2)

If verbose "true", log components are enabled

```
if (verbose)
  {
    LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
  }
```

Create a node container and 2 nodes

```
NodeContainer p2pNodes;
p2pNodes.Create (2);
```

UdpEcho Example (3)

p2p link generation and install on the p2p nodes

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);
```

Install Internet stack on the nodes

```
InternetStackHelper stack;
stack.Install (p2pNodes.Get (0));
stack.Install (p2pNodes.Get (1));
```

UdpEcho Example (4)

Allocate IP address

```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
```

Setup echoServer and Install it on node1

```
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (p2pNodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

Setup echoClient

```
UdpEchoClientHelper echoClient (p2pInterfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (100));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
```

UdpEcho Example (5)

Install echoClient on node0

```
  ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

enable pcap tracing

```
  pointToPoint.EnablePcapAll ("second");

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

OnOffApplication Example (1)

```
// s3_ex2.cc
#include <iostream>
…

using namespace ns3;

int
main (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.Parse (argc, argv);
```

OnOffApplication Example (2)

Create a node container and 2 nodes for p2p link

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

Create a netdevice container and p2p link

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (terminals);
```

OnOffApplication Example (3)

Install Internet stack on the nodes

```
InternetStackHelper internet;
internet.Install (terminals.Get (0));
internet.Install (terminals.Get (1));
```

Allocate IP address

```
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
ipv4.Assign (p2pDevices);
```

OnOffApplication Example (4)

Setup OnOff application

```
uint16_t port = 9;
OnOffHelper onoff ("ns3::UdpSocketFactory",
                  Address (InetSocketAddress (Ipv4Address ("10.1.1.2"), port)));
onoff.SetAttribute ("OnTime",
          StringValue("ns3::ConstantRandomVariable[Constant=1.0]"));
onoff.SetAttribute ("OffTime",
           StringValue("ns3::ConstantRandomVariable[Constant=1.0]"));
onoff.SetAttribute ("DataRate", DataRateValue(5000000));
```

Install OnOff application sender on a node

```
ApplicationContainer app = onoff.Install (terminals.Get (0));

app.Start (Seconds (1.0));
app.Stop (Seconds (10.0));
```

OnOffApplication Example (5)

Create a packet sink to receive packets and install it on a node

```
PacketSinkHelper sink ("ns3::UdpSocketFactory",
        Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
app = sink.Install (terminals.Get (1));
app.Start (Seconds (1.0));
```

Running simulator

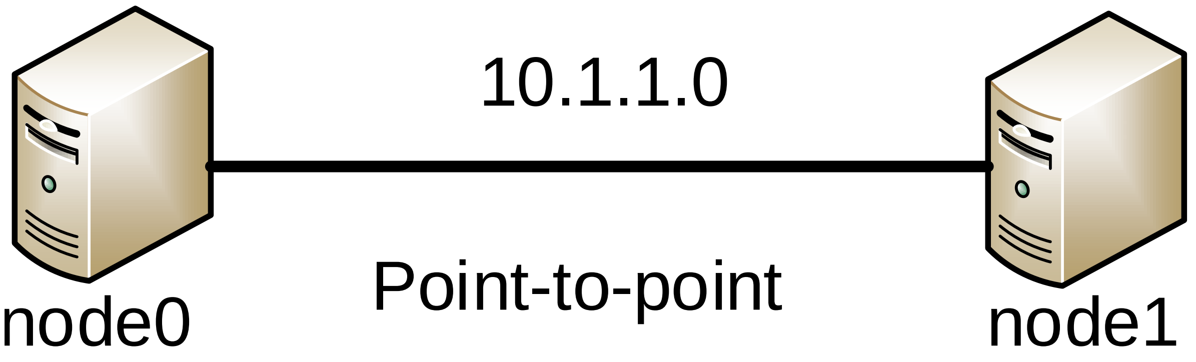
```
  Simulator::Stop(Seconds (15));
  Simulator::Run ();
  Simulator::Destroy ();
}
```

Multimedia & Wireless Networking Laboratory

Exercise

Exercise

- Modify s3_inclass_skel.cc
- UDP Udpclientserver application
  - p2p link: DataRate **5 Mbps**, Delay **10 us**
  - Application & flow
    - Udpclientserver Application
    - UDP flow: node0 → node1, UDP 5 Mbps, 1—10s



Exercise

- Check and verify all the output system of OnOffApplication example
  - Print out all of the logging into .out file
  - Generate .pcap file using PCAP trace
  - Check .pcap file with both **TCPDUMP** and **Wireshark**

Multimedia & Wireless Networking Laboratory

Q & A