Multimedia & Wireless Networking Laboratory

Session 2. Logging과 Command Line Argument ^사용법

장민석

Multimedia & Wireless Networking Laboratory, SNU

msjang@mwnl.snu.ac.kr

Contents

- First ns-3 Script
- Logging Modules
- Command Line Arguments

First ns-3 Script

Module Includes

- Code starts with a number of include statements

```
#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”
```

- Rather than having to look up exactly what header you need, giving the ability to load a group of files
  - Ex)
    - ns3/core-module.h contains header files in src/core
    - build/ns3/core-module.h

Module Includes

- ns-allinone-3.29/ns-3/build/ns3/core-module.h

```
// Module headers:
#include "abort.h"
#include "assert.h"
#include "attribute-accessor-helper.h"
#include "attribute-construction-list.h"
#include "attribute-helper.h"
#include "attribute.h"
#include "boolean.h"
#include "breakpoint.h"
#include "build-profile.h"
#include "calendar-scheduler.h"
#include "callback.h"
#include "command-line.h"
#include "config.h"
#include "default-deleter.h"
#include "default-simulator-impl.h"
#include "deprecated.h"
#include "des-metrics.h"
#include "double.h"
#include "empty.h"
#include "enum.h"
#include "event-garbage-collector.h"
#include "event-id.h"
```

ns3 Namespace

- Namespace declaration

  using namespace ns3;

- ns-3 project is implemented in a C++ namespace called `ns3`
  - groups all ns-3 related declarations in a scope outside the global namespace
  - C++ `using` statement introduces the *ns-3* namespace into the current (global) declarative region
  - helps the integration with other codes

Logging

- Logging component definition

  NS_LOG_COMPONENT_DEFINE (“FirstScriptExample”);

- Declares a logging component called FirstScriptExample
  - Using the component name, we can enable and disable logging messages in the file

Main Function

- Declaration of the main function

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

- Time resolution setting
  - Change the resolution exactly once

```
Time::SetResolution (Time::NS);
```

- Logging components enabling

```
LogComponentEnable("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

Enabling UdpEchoClient and Server Application with info level

Creating Network Topology

- Using topology helpers

- NodeContainer
  - List of Nodes
  - Representative member functions
    - void **Create** (uint32_t n): creates *n* nodes, n: # of Nodes
    - void **Add** (Ptr<Node> node): append a single node
    - Ptr<Node> **Get** (uint32_t i): returns i th node, i: node index
    - uint32_t **GetN** (void): returns total number of nodes
  - Ex)

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

Creating Network Topology

- PointToPointHelper
  - Helps construct a point to point link
  - With high-level functions, NetDevice and Channel are generated and connected
  - Representative member functions
    - void **SetDeviceAttribute** (name, value)
    - void **SetChannelAttribute** (name, value)
    - NetDeviceContainer **Install** (NodeContainer c)
      - Creates p2p net devices and common channel internally, and then install the device and channel on each node in node container c
  - Ex)

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

Creating Network Topology

- NetDeviceContainer
  - List of NetDevice
  - Representative member functions
    - void **Add** (Ptr<Netdevice> device): append a single device
    - Ptr<NetDevice> **Get** (uint32_t i): returns i th net device
    - uint32_t **GetN** (void): returns total number of net devices
  - Ex)

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

After installation, we have two nodes, each with an installed point-to-point net device and a single point-to-point channel between them

Creating Network Topology

- InternetStackHelper
  - Helps construct the Internet stack
  - Representative member functions
    - void **Install** (NodeContainer c)
    - When the install call is executed, it will install Internet stack (TCP, UDP, IP, etc.) on each node in the node container
  - Ex)

```
InternetStackHelper stack;

stack.Install (nodes);
```

Creating Network Topology

- Ipv4AddressHelper
  - Assigning IP address to net devices
  - Representative member functions
    - void **SetBase** (Ipv4Address network, Ipv4Mask mask, Ipv4Address base = “0.0.0.1”)
    - Ipv4InterfaceContainer **Assign** (const NetDeviceContainer &c)
  - Ex)
    - `Ipv4AddressHelper address;`
    - `address.SetBase ("10.1.1.0", "255.255.255.0");`
    - `Ipv4InterfaceContainer interfaces = address.Assign (devices);`
    - After assigning, the IP addresses of the first node and the second node are 10.1.1.1 and 10.1.1.2, respectively

Creating Applications

- ApplicationContainer
  - List of Applications
  - Representative member functions
    - Ptr<Application> **Get** (uint32_t i): returns i th application
    - uint32_t **GetN** (void): returns total number of net devices
    - void Start (Time start): all applications start at start time
    - void Stop (Time stop): all applications stop at stop time
- Ex)

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

Creating Applications

- UdpEchoClientHelper
  - Send UDP packets
  - Representative member functions
    - **UdpEchoClientHelper** (Address ip, uint16_t port)
    - void **Setattribute** (name, value)
      - RemoteAddress, RemotePort, MaxPackets, Interval, PacketSize
    - ApplicationContainer **Install** (NodeContainer c)
- Ex)

```
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));
```

Running Simulator

- Stop function
  - Schedule the simulator stop event

```
Simulator::Stop (Seconds (11.0))
```

- Run function
  - Running the simulator and executing scheduled events in chronological order

```
Simulator::Run ()
```

- When Start and Stop method are called, actually the corresponding events are scheduled

```
serverApps.Start (Seconds (1.0));
```

→ server app start method is scheduled at 1s

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

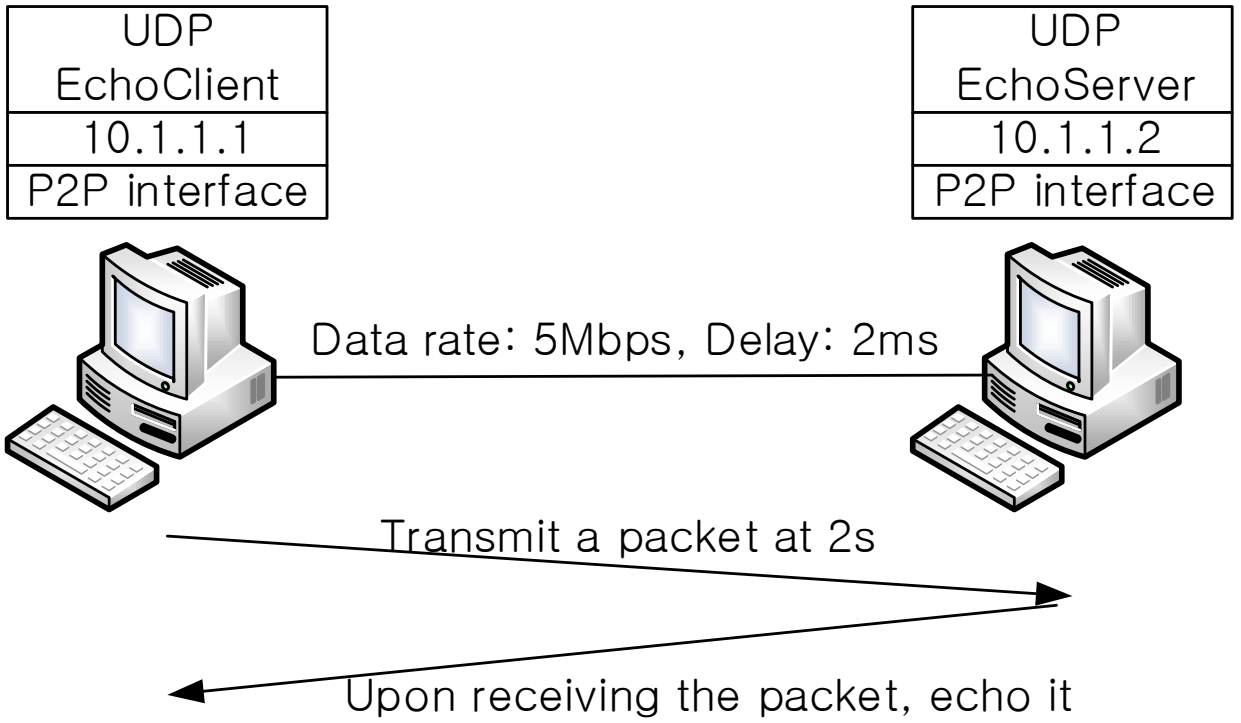
→ server app stop method is scheduled at 10s

Running Simulator

- Destroy function
  - Cleaning up the objects
  - After it is executed, objects are destroyed and the memory is released

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

Simulation Example



Simulation Example (1/3)

- Simple point-to-point link between two nodes and echo a single packet

```
#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[])
{
   Time::SetResolution (Time::NS);
   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"));
```

Simulation Example (2/3)

```
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));
```

Simulation Example (3/3)

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

- ../ns-3.29]$ ./waf --run scratch/s2_ex1

- Output

```
'build' finished successfully (2.238s)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
msjang@gsim2-Z370-UD3H:~/ns-allinone-3.29/ns-3.29$
```

Logging Modules

Logging Module

- Output messages from modules
- Useful when debugging
- NS_LOG environment variable
- NS_LOG_FUNCTION level information
- Verbosity level
  - NS LOG ERROR — Log error messages;
  - NS LOG WARN — Log warning messages;
  - NS LOG DEBUG — Log relatively rare debugging messages;
  - NS LOG INFO — Log informational messages about program progress;
  - NS LOG FUNCTION — Log a message describing each function called;
  - NS LOG LOGIC – Log messages describing logical flow within a function;
  - NS LOG ALL — Log everything.
  - NS LOG UNCOND – Log the associated message unconditionally.

Logging Module

- Example 1
  - $ export NS_LOG=UdpEchoClientApplication=level_all
  - $ ./waf --run scratch/s2_ex1
- Output

```
'build' finished successfully (2.263s)
UdpEchoClientApplication:UdpEchoClient(0x1d2dbb0)
UdpEchoClientApplication:SetDataSize(0x1d2dbb0, 1024)
UdpEchoClientApplication:StartApplication(0x1d2dbb0)
UdpEchoClientApplication:ScheduleTransmit(0x1d2dbb0, +0.0ns)
UdpEchoClientApplication:Send(0x1d2dbb0)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
UdpEchoClientApplication:HandleRead(0x1d2dbb0, 0x1d2f4f0)
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
UdpEchoClientApplication:StopApplication(0x1d2dbb0)
UdpEchoClientApplication:DoDispose(0x1d2dbb0)
UdpEchoClientApplication:~UdpEchoClient(0x1d2dbb0)
msjang@gsim2-Z370-UD3H:~/ns-allinone-3.29/ns-3.29$
```

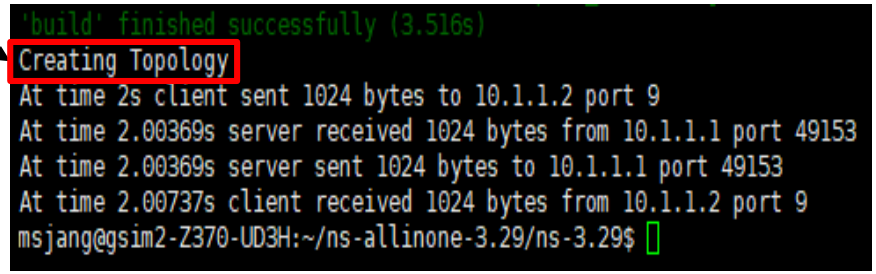
Logging Module

- Example 2
  - $ export ‘NS_LOG=UdpEchoClientApplication=level_all|prefix_time’
  - $ ./waf --run scratch/s2_ex1
- Output

```
'build' finished successfully (2.242s)
+0.000000000s UdpEchoClientApplication:UdpEchoClient(0x2109c40)
+0.000000000s UdpEchoClientApplication:SetDataSize(0x2109c40, 1024)
+2.000000000s UdpEchoClientApplication:StartApplication(0x2109c40)
+2.000000000s UdpEchoClientApplication:ScheduleTransmit(0x2109c40, +0.0ns)
+2.000000000s UdpEchoClientApplication:Send(0x2109c40)
+2.000000000s At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
+2.007372800s UdpEchoClientApplication:HandleRead(0x2109c40, 0x210f0d0)
+2.007372800s At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
+10.000000000s UdpEchoClientApplication:StopApplication(0x2109c40)
UdpEchoClientApplication:DoDispose(0x2109c40)
UdpEchoClientApplication:~UdpEchoClient(0x2109c40)
msjang@gsim2-Z370-UD3H:~/ns-allinone-3.29/ns-3.29$
```

Logging Module

- Example 3
  - $ export ‘NS_LOG=*=level_all|prefix_func|prefix_time’
  - $ ./waf --run scratch/s2_ex1
  - It prints out all of the logging
    - ./waf –-run scratch/s2_ex1 > log.out 2>&1
      - 2>&1 : stderr → stdout
- Example 4
  - Adding logging to your code
    - NS_LOG_**INFO** (**“Creating Topology”**) in your source code
    - $ export NS_LOG=FirstScriptExample=**info**
    - ./waf --run scratch/s2_ex1



- Initializing logging module
  - $ export NS_LOG=

Command Line Arguments

Command Line Arguments

- Change the simulation parameters using command line arguments
- First declare the command line parser in **main function**

```
CommandLine cmd;

cmd.Parse (argc, argv);
```

- Example
  - ./waf --run “scratch/s2_ex1 -PrintHelp”
  - ./waf --run “scratch/s2_ex1 -PrintAttributes=ns3::PointToPointNetDevice”
  - ./waf --run “scratch/s2_ex1
    -ns3::PointToPointNetDevice::DataRate=5Mbps
    -ns3::PointToPointChannel::Delay=2ms”

Command Line Arguments

- When add your own hooks.
  - Use “cmd.AddValue” function in main
    - cmd.AddValue(name, help, value)
- Example

```
int
main (int argc, char *argv[])
{
        uint32_t nPackets =1;
        CommandLine cmd;
        cmd.AddValue(“howmany", "Number of packets to echo", nPackets);
        cmd.Parse (argc, argv);
        …
        echoClient.SetAttribute (“MaxPackets”, UintegerValue (nPackets));
```

- Run simulation with command line argument
  - ./waf --run “scratch/s2_ex1 –howmany=2”

References

- **Network Simulator**
  - http://www.nsnam.org
- **NS-3 tutorial**
  - http://www.nsnam.org/docs/release/3.29/tutorial/singlehtml/index.html
- **NS-3 manual**
  - http://www.nsnam.org/docs/release/3.29/manual/singlehtml/index.html

Q & A