CCE 3301
Communication Systems

Dr Ing. C.J. Debono

Introduction to *NS-3*

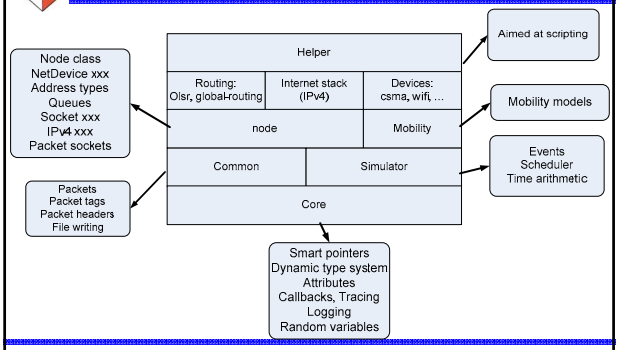
Outline

- Introduction
- NS-3 module stack
- A simple example
- Lower level
  - Scheduler
  - Random variables
  - Memory management
  - Packets
  - Nodes
  - Callbacks
  - Tracing
- NS-3 features

Introduction

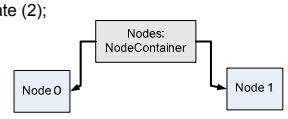
- NS-3 is a new simulator
  - Programming languages: C++, Python
  - Unlike NS-2, everything designed for C++
  - Optional Python scripting
- Project started around mid 2006
- ns-3 is written in C++
- Bindings in Python
- ns-3 uses the waf build system
- API documentation using doxygen

NS-3 Modules



Example (first.cc)

```
int main (int argc, char *argv[])
{
    LogComponentEnable ("UdpEchoClientApplication",
                                        LOG_LEVEL_INFO);
   LogComponentEnable ("UdpEchoServerApplication",
                                        LOG_LEVEL_INFO);
    RandomVariable: :UseGlobalSeed (1, 1, 2, 3, 5, 8);
    NodeContainer nodes;
    nodes.Create (2);
```



Example (first.cc)

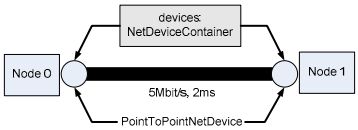
PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute (“DataRate”, StringValue (5Mbps”));

pointToPoint.SetChannelAttribute (“Delay”, StringValue (“2ms”));

NetDeviceContainer devices;

devices = pointToPoint.Install (nodes);

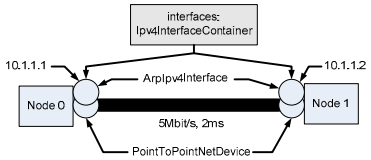


Example (first.cc)

InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase (“10.1.1.0”, “255.255.255.0”);

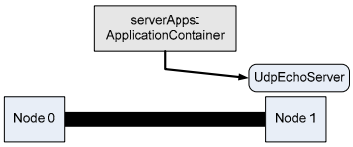
Ipv4InterfaceContainer interfaces = address.Assign (devices);



Example (first.cc)

UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps =
echoServer.Install (nodes.Get (1));

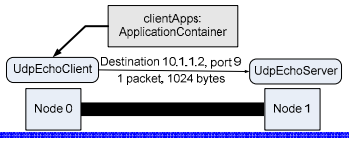
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));



Example (first.cc)

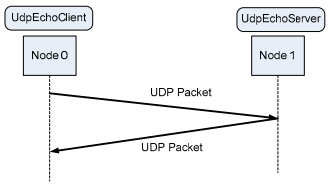
UdpEchoClientHelper *echoClient* (*interfaces*.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

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



Example (first.cc)

[…]
Simulator::Run ();
Simulator::Destroy ();
return 0;
}



$ ./waf -- run
[…]
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
…

Same example in Phyton

```
import ns3
ns3.LogComponentEnable("UdpEchoClientApplication", ns3.LOG_LEVEL_INFO)
ns3.LogComponentEnable("UdpEchoServerApplication", ns3.LOG_LEVEL_INFO)
ns3.RandomVariable.UseGlobalSeed(1, 1, 2, 3, 5, 8)
nodes = ns3.NodeContainer()
nodes.Create(2)
pointToPoint = ns3.PointToPointHelper()
pointToPoint.SetDeviceAttribute("DataRate", ns3.StringValue("5Mbps"))
pointToPoint.SetChannelAttribute("Delay", ns3.StringValue("2ms"))
devices = pointToPoint.Install(nodes)
stack = ns3.InternetStackHelper()
stack.Install(nodes)
address = ns3.Ipv4AddressHelper()
address.SetBase(ns3.Ipv4Address("10.1.1.0"), ns3.Ipv4Mask("255.255.255.0"))
interfaces = address.Assign(devices)
echoServer = ns3.UdpEchoServerHelper(9)
serverApps = echoServer.Install(nodes.Get(1))
serverApps.Start(ns3.Seconds(1.0))
serverApps.Stop(ns3.Seconds(10.0))
echoClient = ns3.UdpEchoClientHelper(interfaces.GetAddress(1), 9)
echoClient.SetAttribute("MaxPackets", ns3.UintegerValue(1))
echoClient.SetAttribute("Interval", ns3.TimeValue(ns3.Seconds (1.0)))
echoClient.SetAttribute("PacketSize", ns3.UintegerValue(1024))
clientApps = echoClient.Install(nodes.Get(0))
clientApps.Start(ns3.Seconds(2.0))
clientApps.Stop(ns3.Seconds(10.0))
ns3.Simulator.Run()
ns3.Simulator.Destroy()
```

Running Programs

- Install all needed tools
  - sudo apt-get install build-essential g++ python mercurial (ubuntu)
- Programs are built as
  - Build/<variant>/path/program_name
    - <variant> = debug or optimized
  - Using ./waf
    - ./waf –shell
    - ./build/debug/examples/simple-point-to-point
  - Using waf – run
    - ./waf –run simple-point-to-point
  - Waf homepage code.google.com/p/waf

Simulator Core

- Time is not manipulated directly: the Time class
  - Time class supports high precision 128 bit time values(ns precision)
    - Time t1 = Seconds (10);
    - Time t2 = t1 + MilliSeconds (100);
    - std::cout << t2.GetSeconds () << std::endl; // t2 = 10.1
- Get current time:
  - Time now = Simulator::Now ();
- Schedule an event to happen in 3 seconds:
  - Void MyCallback (T1 param1, T2 param2) {...}
  - […]
  - Simulator::Schedule (Seconds (3), MyCallback, param1, param2);
  - Values *param1* and *param2* passed as callback parameters
- Also works with instance methods:
  - Simulator::Schedule (Seconds (3), &MyClass::Method, instancePtr, param1, param2);

Random Variables

- Implemented distributions
  - Uniform
  - Constant
  - Sequential
  - Exponential
  - Normal
  - Log-normal
  - …

Memory Management

- Many NS-3 objects use automatic garbage collection
- Reference counting
  - Packet *p = new Packet; # refcount starts at 1
  - p ->Ref (); #refcount becomes 2
  - p ->Unref (); #refcount becomes 1
  - p ->Unref (); #refcount becomes 0, packet is freed
- Smart pointers
  - Take care of all reference counting work
  - Ex:

```
void MyFunction ()
{ Ptr<Packet> p = Create<Packet> (10);
std::cerr << ”Packet size: ” << p->GetSize () << std::endl;
} # Packet is released
```


Packets

- Packet objects used *vertically* in NS-3 to represent:
  - Units of information sent and received by applications
  - Information chunks of what will become a real packet
  - Simulated packets and L2/L1 frames being transmitted
- Basic Usage
  - Create an empty packet
    - Ptr<Packet> packet = Create<Packet> ();
- Create a packet with 10 ”dummy” bytes
  - Ptr<Packet> packet = Create<Packet> (10);
  - “Dummy” bytes are simulated, but do not actually occupy any memory
- Create a packet with user data
  - Ptr<Packet> packet = Create<Packet> (”hello”, 5);
- Copy a packet
  - Ptr<Packet> packet2 = packet1->Copy ();

Nodes

- Node class
  - Represents a network element
  - May have an ***IPv4 stack*** object
  - May have a ***mobility model***
    - ex CsmaNetDevice needs no mobility model
  - Contains a list of ***NetDevice***s
  - Contains a list of ***Applications***
- NodeList class (singleton)
  - Tracks all nodes ever created
  - Node index <=> Ptr conversions

Callbacks

- NS-3 Callback class implements ***function objects***
  - Type safe callbacks, manipulated by value
  - Used in sockets and tracing
- Example

```
double MyFunc (int x, float y) {
return double (x + y) / 2;
}
[...]
Callback<double, int, float> cb1;
cb1 = MakeCallback (MyFunc);
double result = cb1 (2, 3);
```


Tracing

```
uint64_t g_packetDrops = 0;
uint64_t g_packetDropBytes = 0;

void TraceDevQueueDrop (std::string context,
Ptr<const Packet> droppedPacket)
{
g_packetDrops += 1;
g_packetDropBytes += droppedPacket->GetSize ();
}

int main (int argc, char *argv[])
{
[…]
Config::Connect ("/NodeList/*/DeviceList/*/TxQueue/Drop",
MakeCallback (&TraceDevQueueDrop));
[…]
}
```


Features

- Scalable
  - Packets can have virtual zero bytes
  - Optional features
    - Mobility model may not be needed
    - No memory waste in IPv4 stack (if nodes do not need it)
    - Easy addition of new features
  - Cross-layer features
    - Packet tags
      - Attached to packets
    - Tracing
      - Event reporting across layers

Conclusion

- Decide what you want to simulate
  - Define the topology of the network
  - Create nodes, channel, and network interfaces
  - Configure IPv4 stack and applications
  - Set attributes
- Build the simulation script using a text editor
- Execute the .cc program via waf
- Analyze the output
- Document the work (easier to debug!!)

Further Information

- Tutorial available here:
  http://www.nsnam.org/docs/release/tutorial.pdf

Today's Assignment

- 2 nodes with 1 interface each
- Point-to-point link having a data rate of 2Mbps and transmission delay of 2ms
- IP address assigned 192.168.15.0/24
- Application with a packet size of 256 bytes on server port 65
- Enable ASCII and pcap tracing
- Other attributes use default in example first.cc

Acknowledgements

- Material for this tutorial comes from the presentation by

  Gustavo J. A. M. Carneiro