Communication Principles and Wireless Networks
Lab 1: NS3 Propagation Models

WN Lab, Institute of Network Engineering
MBWC Lab, Institute of Communications Engineering
National Yang Ming Chiao Tung University, Taiwan

Outline

A. Installation
   - Prerequisites
   - Build & Validation

B. Introduction
   - Overview
   - Directory structure
   - Hello Simulator
   - Simulation structure
   - Walkthrough of third.cc

C. Exercise

D. Turn in

E. Appendix
   - Analyzing in Wireshark

Installation - Prerequisites

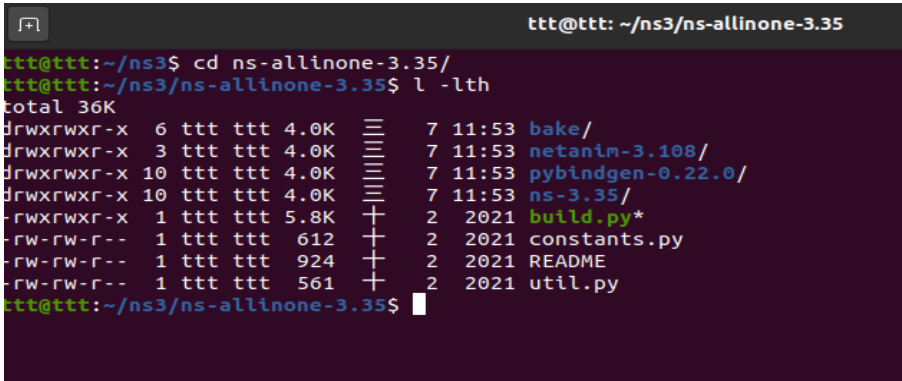
- **Setup OS environment**
  - **Minimum requirement: 20GB of memory allocation and 2~4GB RAM allocation(less means slower)**
  - **Linux Ubuntu 20.04 LTS(https://www.ubuntu-tw.org/modules/tinyd0/)**
    Note: recommend to use virtual machine(eg. virtualbox) if your os is windows.
  - **Download vscode for writing codes(https://code.visualstudio.com/#alt-downloads)**

Installation - Prerequisites

- Install ns-3 specific dependencies
  - $ sudo apt update && sudo apt upgrade -y
  - $ sudo apt install build-essential git python3-setuptools castxml -y
  - $ sudo apt install g++ pkg-config sqlite3 qt5-default -y

- Download ns-3 package and unzip it
  - $ cd ~
  - $ mkdir ns3 && cd ns3
  - $ wget https://www.nsnam.org/release/ns-allinone-3.35.tar.bz2
  - $ tar xjf ns-allinone-3.35.tar.bz2

- Following above steps, if your change into the dir. ns-allinone-3.35, you should see a number of files and dirs
  - $ cd ns-allinone-3.35/



Installation - Build & Validation

- Build ns-3 simulator with waf command

  $ cd ns3/ns-allinone-3.35/ns-3.35/

  $ ./waf configure --enable-tests --enable-examples

  $ ./waf build

- Validate the build

  $ ./waf check

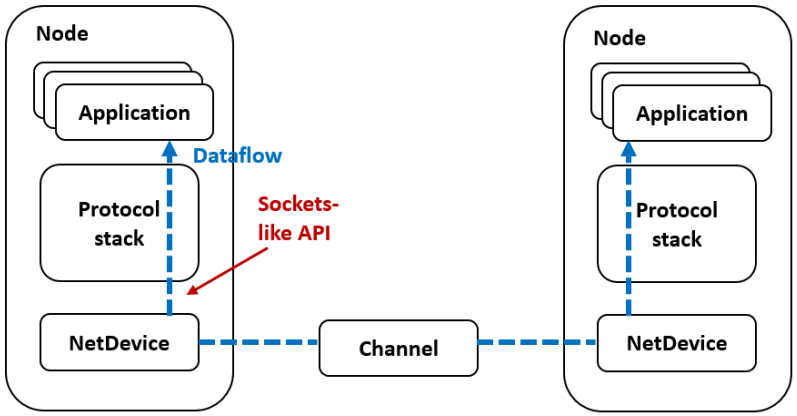
- Test the simulator, the terminal should output ”Hello Simulator”

  $ ./waf --run hello-simulator

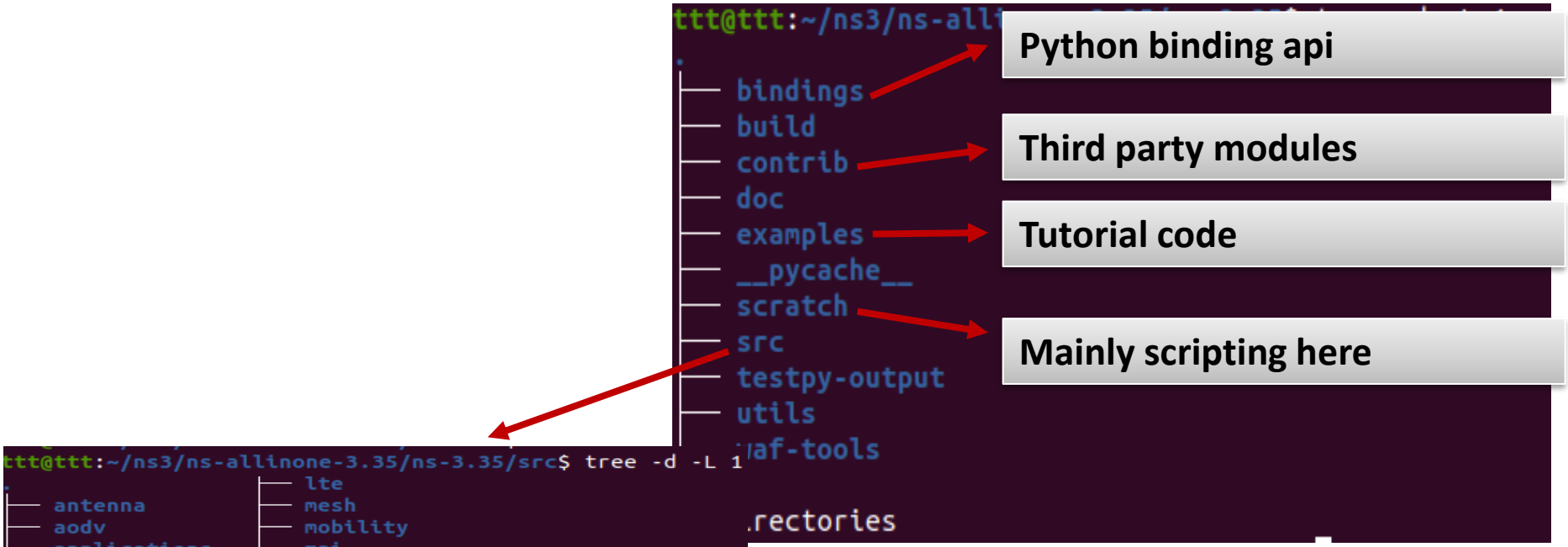
```
ttt@ttt:~/ns3/ns-allinone-3.35/ns-3.35$ ./waf --run hello-simulator
Waf: Entering directory `/home/ttt/ns3/ns-allinone-3.35/ns-3.35/build'
Waf: Leaving directory `/home/ttt/ns3/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.662s)
Hello Simulator
ttt@ttt:~/ns3/ns-allinone-3.35/ns-3.35$
```

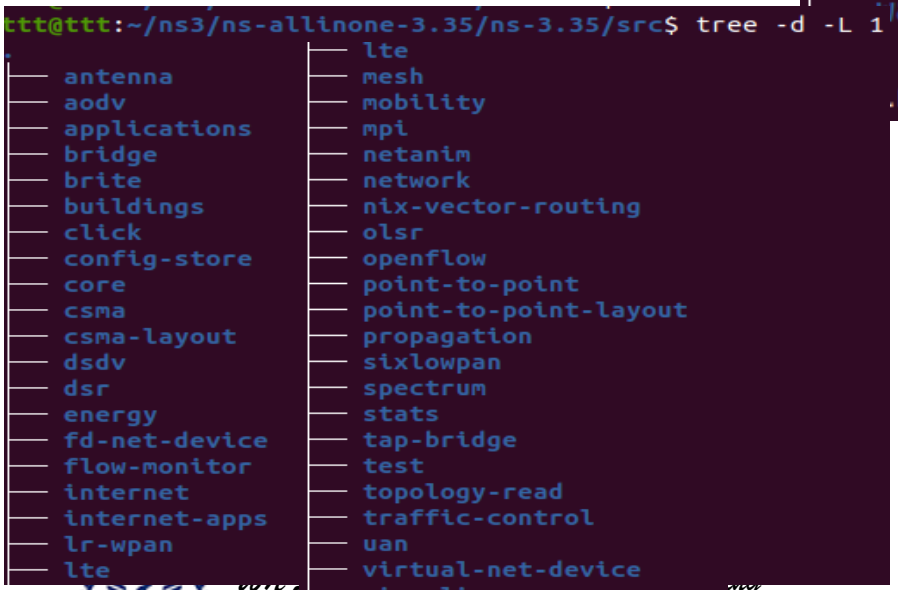
Introduction - Overview

- NS-3 is free and open source discrete event network simulator
- Users can write in c++, with optional python interface for visualization and scripting
- Support different network layers
  - Applications: On/Off, Bulk transfer, HTTP, etc.
  - Transport: TCP, UDP
  - Network: IPv4, IPv6, routing
  - Physical: Ethernet, wifi, LTE, etc.
- Key abstractions
  - NetDevice: tx/rx over the channel
  - Channel: tx/rx medium b/w nodes
  - Application: create data flow
  - Helper: use to quickly configure above function



Introduction - Directory structure





- Lots of modules could be used in the script depending on your simulation scenarios

Introduction – Hello Simulator

- Basic structure

```
#include “ns3/core-module.h”        → Include necessary module depending on simulation scenarios
using namespace ns3;                → Group all ns3 related declarations in ns3 namespace
NS_LOG_COMPONENT_DEFINE(“HelloSimulator”);   → Logging module section
int main(int argc, char *argv[]){
        CommandLine cmd(__File__);
        cmd.Parse(argc, argc);      } cmd arguments parser
        LogComponentEnable(“HelloSimulator”, LOG_LEVEL_ALL)   → Enable different logging levels
        NS_LOG_INFO(“Hello Simulator”);
            ⋮                       → Write your simulation logic here!!!
        Simulator::Run();
        Simulator::Destroy();       } Start the simulation and cleanup afterwards
        return 0;
}
```

Introduction - Simulation structure

- Create nodes

```
NodeContainer nodes;
nodes.Create(uint32_t n); // create n nodes and append pointers to them
nodes.Get(uint32_t n); // get the Ptr<Node> at a given index
```

- Install NetDevice

```
NetDeviceContainer dev;
PointToPointHelper ptp;
ptp.SetDeviceAttribute(std::string name, const AttributeValue &value);
ptp.SetChannelAttribute(std::string name, const AttributeValue &value);
ptp.Install(NodeContainer c);
```

- Install protocol stack

```
InternetStackHelper stk;
stk.install(NodeContainer c);
```

Introduction - Simulation structure

- Assign IP address

```
Ipv4AddressHelper ipv4;
ipv4.SetBase(Ipv4Address network,
             Ipv4Mask mask,
             Ipv4Address base = "0.0.0.1")
ipv4.Assign(const NetDeviceContainer &c);
```

- Install application

```
UdpEchoServerHelper(uint16_t port);
UdpEchoClientHelper(Address ip, uint16_t port);
BulkSendHelper(std::string protocol, Address address);
        ⋮
```

- Tracing

```
AsciiTraceHelper();
ptp.EnablePcapAll(std::string prefix);  // use wireshark to open
Config::Connect(std::string context, MakeCallback(&TraceSink));
```

Introduction - Walkthrough of third.cc

```
NodeContainer wifiStaNodes;
wifiStaNodes.Create(1);
NodeContainer wifiApNode;
wifiApNode.Create(1);
```

- Create nodes and NodeContainer
- Different containers for AP and stations for purpose of application/device installation later

Introduction - Walkthrough of third.cc

```
YansWifiChannelHelper channel;
channel.AddPropagationLoss("ns3::LogDistancePropagationLossModel",
                           "Exponent", DoubleValue(3.0),
                           "ReferenceDistance", DobuleValue(1.0),
                           "ReferenceLoss", DoubleValue(46.677));
channel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
wifiPhy.SetChannel(channel.Create());
```

- Use helper function to setup the wifi channel
- Specify physical attributes for different propagation model

Note: https://www.nsnam.org/docs/models/html/propagation.html

Introduction - Walkthrough of third.cc

```
wifiHelper wifi;
WifiMacHelper wifiMac;
YansWifiPhyHelper wifiPhy;
Ssid ssid = Ssid("wifi-default");
wifi.SetRemoteStationManager("ns3::ArfWifiManager")
wifiMac.SetType("ns3::StaWifiMac",
                "ActiveProbing", BooleanValue(true),
                "Ssid", SsidValue(ssid));
wifi.Install(wifiPhy, wifiMac, wifiStaNodes);
wifiMac.SetType("ns3::ApWifiMac",
                "Ssid", SsidValue(ssid));
wifi.Install(wifiPhy, wifiMac, wifiApNode);
```

- Install NetDevice for wifi nodes
- Use wifiHelper function for setting station manger or setting standards

Note: https://www.nsnam.org/docs/models/html/wifi-user.html

Introduction - Walkthrough of third.cc

```
MobilityHelper mobility;
mobility.SetPositionAllocator(“ns3::RandomRectanglePositionAllocator ”,
                              “X”, StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=50.0]"),
                              “Y”, StringValue("ns3::UniformRandomVariable[Min=0.0|Max=50.0]"));
mobility.SetMobilityModel(“ns3::RandomWalk2dMobilityModel”,
                          “Bounds”, RectangleValue(Rectangle(-50, 50, -50, 50)));
mobility.Install(wifiStaNodes);
```

- **Initialize position and specify movement of nodes**

  Note: https://www.nsnam.org/docs/models/html/mobility.html

Introduction - Walkthrough of third.cc

```
InternetStackHelper stack;
stack.Install (wifiStaNodes);
stack.Install (wifiApNodes);
Ipv4AddressHelper address;
address.SetBase ("192.168.0.0", "255.255.255.0");
Ipv4InterfaceContainer stainterface = address.Assign (staDevs);

UdpEchoServerHelper serv(9);
ApplicationContainer servApps = serv.Install(ap.Get(0));

UdpEchoClientHelper client(stainterface.GetAddress(1), 9);
client.SetAttribute("MaxPackets", UintegerValue(1000));
client.SetAttribute("Interval", TimeValue(Seconds(1)));
client.SetAttribute("PacketSize", UintegerValue(1024));
ApplicationContainer clientApps = client.Install(wifiStaNodes.Get(0));
```

- Install Application to send or receive packets

Introduction - Walkthrough of third.cc

```
void Position(std::string context, Ptr<const MobilityModel> model)
{
        Vector pos = model->GetPosition();
        std::cout << " x: " << pos.x
                  << " y: " << pos.y
                  << " z: " << pos.z << std::endl;
}
Config::Connect("/NodeList/*/$ns3::MobilityModel/CourseChange",
                MakeCallback(&Position));
```

- Trace sources are entities that can signal events that happen during simulation and provide access to interesting underlying data
- Callback function: void Trace(std::string context, T1 value, T2 value, …)

Note: https://www.nsnam.org/docs/release/3.35/doxygen/_trace_source_list.html

Introduction - Walkthrough of third.cc

```
Simulator::Stop(Seconds(30.0));
Simulator::Run();
Simulator::Destroy();
```

- Run the simulation, clean up and then exit the program

- In order to run script, you have to copy it into scratch directory and use Waf to build and run it

  $ ./waf --run scratch/scriptname.cc

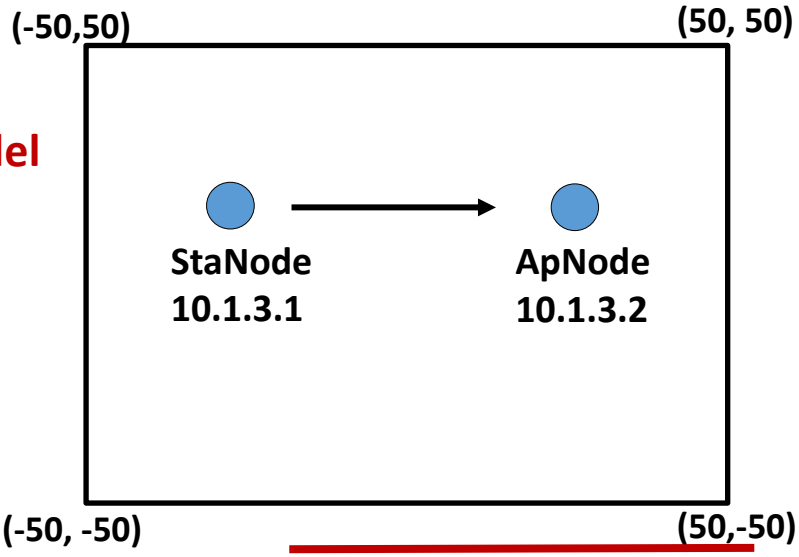
- If command line arguments are required

  $ ./waf --run “scratch/scriptname.cc --argument=value”

Exercise

- Use ppg.cc to see how different propagation model performs. First, you should complete TODO sections. There are four parts in the script, such as create nodes, change propagation model, install mobility and hook trace source

- After completion, plot two figures using log-distance path loss model and Rayleigh fading model
  - Figure 1: combine log-distance path loss model and Rayleigh fading model
    - X-axis: distance(m)
    - Y-axis: received signal strength(dBm)
  - Figure 2: only log-distance path loss model
    - X-axis: PLE(path loss exponent)
    - Y-axis: throughput(Mbps)



Turn in

- **Report:**
  - **Submit the report in pdf format. It should cover**
    a) **Experiment settings and any parameters you use**
    b) **Figures with observation and brief discussion**
- **Source code**

Note: Please zip the source code and report in one file and name it like yourstudentID_name.zip(e.g. 0856XXX_王小明.zip)

Appendix - Analyzing in Wireshark

- Install wireshark first

  $ sudo apt install wireshark

- After simulation done, *.pcap files will be generated

```
ttt@ttt:~/ns3/ns-allinone-3.35/ns-3.35$ ls
AUTHORS         contrib            LICENSE       RELEASE_NOTES
bindings        CONTRIBUTING.md    Makefile      scratch
build           doc                __pycache__   ScratchSimulator-0-0.pcap
CHANGES.html    examples           README.md     ScratchSimulator-1-0.pcap
ttt@ttt:~/ns3/ns-allinone-3.35/ns-3.35$
```

- Examine the files

  $ wireshark ScratchSimulator-0-0.pcap

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.1	10.1.1.2	TCP	58	49153 → 9 [SYN] Seq=0
2	0.011856	10.1.1.2	10.1.1.1	TCP	58	9 → 49153 [SYN, ACK]
3	0.011856	10.1.1.1	10.1.1.2	TCP	54	49153 → 9 [ACK] Seq=1
4	0.012720	10.1.1.1	10.1.1.2	TCP	590	49153 → 9 [ACK] Seq=1
5	0.022160	10.1.1.1	10.1.1.2	TCP	542	49153 → 9 [FIN, ACK]
6	0.033024	10.1.1.2	10.1.1.1	TCP	54	9 → 49153 [ACK] Seq=1
7	0.041696	10.1.1.2	10.1.1.1	TCP	54	9 → 49153 [ACK] Seq=1
8	0.042560	10.1.1.2	10.1.1.1	TCP	54	9 → 49153 [FIN, ACK]
9	0.042560	10.1.1.1	10.1.1.2	TCP	54	49153 → 9 [ACK] Seq=1

```
▸ Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
▸ Point-to-Point Protocol
▸ Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.1.1.2
▸ Transmission Control Protocol, Src Port: 49153, Dst Port: 9, Seq: 0, Len: 0
```