LAB4: ns-3 Tracing System
CS169: Mobile Wireless Networks - Winter 2017

Kittipat Apicharttrisorn (Patrick)
Department of Computer Science and Engineering
University of California, Riverside

February 6-7, 2017
Dale’s Cone of Experience

People Generally Remember:

- 10% of what they Read
- 20% of what they Hear
- 30% of what they See
- 50% of what they hear and see
- 70% of what they say and write
- 90% of what they do

People Are Able To: (Learning Outcomes)

- Define List
- Describe Explain
- Demonstrate Apply Practice
- Analyze Design Create Evaluate
- Participate in Hands-On Workshop
- Design Collaborative Lessons
- Simulate or Model a Real Experience
- Design/Perform a Presentation - Do The Real Thing
NS_LOG and std::cout are quick and dirty, so they may not be effective for serious work.

What if we want to look at specific data or state changes?

Trace source (generators of trace data) → trace sink (consumer)

Ex. we are interested in Congestion Window size or Mobility Tracking locations.
ns-3 Tracing Model

- Decouple trace sources from trace sinks:
Callbacks

- pointer-to-function-returning-integer (PFI)
  
  \[ int (*pfi)(int arg) = 0; \]

- Creating MyFunction returning int
  
  \[ int MyFunction (int arg) \{
  \}
  
- Initialize pfi to point to MyFunction
  
  \[ pfi = MyFunction; \]

- Then we can call MyFunction indirectly by
  
  \[ int result = (*pfi) (1234) \]

  or

  \[ int result = pfi (1234) \]

- The system maintains a list of callbacks triggered by events of interest, whose data are passed along from trace source to the target function (sink)
```cpp
class MyObject : public Object
{
    public:
        /**
         * Register this type.
         * \return The TypeId.
         */
        static TypeId GetTypeId (void)
        {
            static TypeId tid = TypeId("MyObject")
                .SetParent<Object>()
                .SetGroupName("Tutorial")
                .AddConstructor<MyObject>()
                // connect trace source
                .AddTraceSource("MyInteger", // trace source name
                    "An integer value to trace.", // helping string
                    MakeTraceSourceAccessor (&MyObject::m_myInt), // TracedV
                    value added to the class
                    "ns3::TracedValueCallback::Int32") // for documentation
                ;
            return tid;
        }

    MyObject () {}  
    TracedValue<int32_t> m_myInt;
};
```
Trace sink function

```cpp
//trace sink function
void
IntTrace (int32_t oldValue, int32_t newValue) //matched callback signature
{
    std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}
```
main

```cpp
int main (int argc, char *argv[])
{
    // create a MyObject instance named myObject
    Ptr<MyObject> myObject = CreateObject<MyObject> ();
    // connect trace source MyInteger with trace sink function
    // through MakeCallBack
    myObject->TraceConnectWithoutContext ("MyInteger", MakeCallback (&IntTrace)) ;
    // set member variable m_myInt to value "1234" which triggers a callback
    myObject->m_myInt = 1234;
}
```

Let’s make some change to `myObject → m_myInt` and see what will happen.
Connect with Config Subsystem

- Let’s re-visit mythird
- $ vim scratch/mythird.cc
- and insert this code before `int main` and do you think this is trace source or trace sink?

```cpp
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

void CourseChange (std::string context, Ptr<const MobilityModel> model){
    Vector position = model->GetPosition ();
    NS_LOG_UNCOND (context <<
        " x = " << position.x << ", y = " << position.y);
}

int main (int argc, char *argv[])
{
```
Connect with Config Subsystem

- Then we use a config path as a trace source by inserting this before `Simulator::Run();`
- Then try running `mythird`

```cpp
//track locations
std::ostringstream oss;
oss << 
"/NodeList/" << wifiStaNodes.Get (nWifi - 1)->GetId () << 
"/$ns3::MobilityModel/CourseChange";

Config::Connect (oss.str (), MakeCallback (&CourseChange));

Simulator::Run ();
Simulator::Destroy ();
return 0;
```
Actually, the config path 
`/NodeList/7/$ns3::MobilityModel/CourseChange` is broken down to

- `/NodeList/7/` is a node object and `$ns3::MobilityModel` is another object aggregated with the node object and finally `CourseChange` is the attribute we want to take a look
$ cp examples/tutorial/fifth.cc scratch/myfifth.cc
$ vim scratch/myfifth.cc
$ ./waf --run scratch/myfifth.cc > cwnd.dat 2>&1

Take a look at the result. Do you see congestion window changes? Do you see packet drop? How many times per second?
Exercise

- From mythird, show the CourseChange of all the WiFi nodes. Plot data into x-y axes to show where they go.
- From myfifth, plot congestion window with time. Set receive error rate to $10^{-4}$, $10^{-3}$, $10^{-2}$ and compare congestion window and Rx drop rate (packets/second).
Questions?