ns-3 Training

Session 8: Monday 3:30pm

ns-3 Annual Meeting
May 2014
Outline

• Emulation modes
  – Tap Bridge
  – FdNetDevice

• Direct Code Execution (DCE)
  – Applications
  – Linux Kernel
  – DCE Cradle
Emulation support

- Support moving between simulation and testbeds or live systems
- A real-time scheduler, and support for two modes of emulation
- Linux is only operating system supported
- Must run simulator in real time
  - GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealTimeSimulatorImpl"));
- Must enable checksum calculations across models
  - GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));
- Must run as root, or with the --enable-sudo
ns-3 emulation modes

1) ns-3 interconnects real or virtual machines

2) testbeds interconnect ns-3 stacks

Various hybrids of the above are possible
Example use case: testbeds

- Support for use of Rutgers WINLAB ORBIT radio grid
Example use case: PlanetLab

- The PlanetLabFdNetDeviceHelper creates TAP devices on PlanetLab nodes using specific PlanetLab mechanisms (i.e. the vsys system), and associates the TAP device to a FdNetDevice in ns-3.
Example use case: mininet

- Mininet is popular in the Software-Defined Networking (SDN) community
- Mininet uses "TapBridge" integration
- https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3

![GitHub repository for mininet](image-url)
Emulation Devices
Device models

• **File Descriptor Net Device (FdNetDevice)**
  – read and write traffic using a file descriptor provided by the user
  – this file descriptor can be associated to a TAP device, to a raw socket, to a user space process generating/consuming traffic, etc.

• **Tap Bridge**
  – Integrate Tun/Tap devices with ns-3 devices

• **EmuNetDevice**
  – Deprecated (ns-3.17) in favor of FdNetDevice
“TapBridge": netns and ns-3 integration

Tap device pushed into namespaces; no bridging needed
TapBridge modes

• ConfigureLocal (default mode)
  – ns-3 configures the tap device
  – useful for host to ns-3 interaction

• UseLocal
  – user has responsibility for device creation
  – ns-3 informed of device using “DeviceName” attribute

• UseBridge
  – TapDevice connected to existing Linux bridge
ConfigureLocal

ns-3 ensures that Mac addresses are consistent.
UseLocal

Mac X spoofed to Mac Y
ns-3 devices must support SendFrom() (i.e. bridging)
TapCsma example

• Demo the TapCsma example
FdNetDevice

• Unified handling of reading/writing from file descriptor

• Three supported helper configurations:
  – EmuFdNetDeviceHelper (to associate the ns-3 device with a physical device in the host machine)
  – TapFdNetDeviceHelper (to associate the ns-3 device with the file descriptor from a tap device in the host machine) (not the same as TapBridge)
  – PlanetLabFdNetDeviceHelper (to automate the creation of tap devices in PlanetLab nodes, enabling ns-3 simulations that can send and receive traffic though the Internet using PlanetLab resource.)
EmuFdNetDeviceHelper

- Device performs MAC spoofing to separate emulation from host traffic
PlanetLabFdNetDeviceHelper

- Special case of TapFdNetDeviceHelper where Tap devices configured according to PlanetLab conventions
ns-3 over host sockets

- Two publications about how to run ns-3 applications over real hosts and sockets
  - "Simulator-agnostic ns-3 Applications", Abraham and Riley, WNS3 2012
Generic Emulation Issues

• Ease of use
  – Configuration management and coherence
  – Information coordination (two sets of state)
    • e.g. IP/MAC address coordination
  – Output data exists in two domains
  – Debugging can be more challenging

• Error-free operation (avoidance of misuse)
  – Synchronization, information sharing, exception handling
    • Checkpoints for execution bring-up
    • Inoperative commands within an execution domain
    • Deal with run-time errors
  – Soft performance degradation (CPU) and time discontinuities
Direct Code Execution
**Goals**

- Lightweight virtualization of kernel and application processes, interconnected by simulated networks

**Benefits:**
- Implementation realism in controlled topologies or wireless environments
- Model availability
- Debugging a whole network within a single process

**Limitations:**
- Not as scalable as pure simulation
- Tracing more limited
- Configuration different
Direct Code Execution

• DCE/ns-3 framework requires the virtualization of a series of services
  – Multiple isolated instances of the same protocol on the same machine
• System calls are captured and treated by DCE
• Network stack protocols calls are captured and redirected
• To perform its work DCE re-implement the Linux program loader and parts of libc and libpthread
Direct Code Execution

• Developed by Mathieu Lacage and Frederic Urbani, INRIA, Hajime Tazaki (University of Tokyo)

Figure 1: Architecture of Direct Code Execution. Kernel network devices and timers are synchronized with simulated NetDevice and clock.

Figure source: Direct Code Execution: Revisiting Library OS Architecture for Reproducible Network Experiments (CONEXT 13)
DCE modes

• DCE modes in context of possible approaches

Figure 1: Current possible combinations of network stacks and applications.

Figure source: DCE Cradle: Simulate Network Protocols with Real Stacks for Better Realism, Tazaki et al, WNS3 2013.
Paper references

• Direct Code Execution: Revisiting Library OS Architecture for Reproducible Network Experiments
  – Tazaki et al, CONEXT 2013
  – http://hal.archives-ouvertes.fr/docs/00/88/08/70/PDF/con013-hal.pdf

• DCE Cradle: Simulate Network Protocols with Real Stacks for Better Realism
  – Tazaki et al, WNS3 2013
  – http://hal.archives-ouvertes.fr/docs/00/78/15/91/PDF/wns3-2013.pdf
Hands on, what do we need

- **What do you need to start using the framework!**
  - ns-3 – The network simulator
    - [http://www.nsnam.org/](http://www.nsnam.org/)
  - DCE
  - Applications
    - iperf, wget, thttpd
  * All software must be re-compiled with –fpic and linked with –pie to generate the code with Position Independent Code (PIC) and permit context switch

- **To make things easier**
  - Bake – Installation tool
    - [http://planete.inria.fr/software/bake/index.html](http://planete.inria.fr/software/bake/index.html)
  - Mercurial – source control management tool
  - Python – for running bake
    - [www.python.org](http://www.python.org)
The plan is

- The plan is to present
  - Installation
  - Examples of:
    - iperf with ns-3 stack
    - www server and wget with ns-3 stack
    - iperf with Linux stack
The shared scenario

- The shared scenario is a simple three nodes network

![Diagram of a three nodes network with IP addresses and bandwidth specifications]
Step by step example
- Installing the required software

* Into a Linux machine

1) > mkdir dce_tutorial; cd dce_tutorial
2) > hg clone http://code.nsnam.org/bake bake
3) > export BAKE_HOME=`pwd`/bake
4) > export PATH=$PATH:$BAKE_HOME
5) > export PYTHONPATH=$PYTHONPATH:$BAKE_HOME
6) > mkdir DCE; cd DCE
7) > bake.py configure -e dce-ns3
8) > bake.py install
9) > . bakeSetEnv.sh
Step by step example
- What we need to do!

1. Create the nodes
2. Create stack
3. Create devices
4. Set addresses
5. Connect devices
6. Create DCE
7. Configuration the applications to run
8. Set start time for server and client
9. Set simulation time
10. Start simulation
Step by step example
- What we need to do!

1) Create the nodes
2) Create stack
3) Create devices
4) Set addresses
5) Connect devices
6) Create DCE
7) Configuration the applications to run
8) Set start time for server and client
9) Set simulation time
10) Start simulation

Standard ns-3 procedures
DCE specific
Step by step example
- iperf with ns-3 stack (I)

```cpp
int main (int argc, char *argv[]) {
    // Node Container creation
    NodeContainer nodes;
    nodes.Create (3);

    // Linux stack creation
    InternetStackHelper stack;
    stack.Install (nodes);

    // For real time
    // GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealtimeSimulatorImpl"));
    // GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));

    // Device and channel creation
    PointToPointHelper p2p;
    p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    p2p.SetChannelAttribute ("Delay", StringValue ("1ms"));
}
```
Step by step example
- iperf with ns-3 stack (II)

// Node0-Node1 setup
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.252"); // Node0-Node1 addresses

NetDeviceContainer devices;
devices = p2p.Install(nodes.Get(0), nodes.Get(1)); // connecting nodes
Ipv4InterfaceContainer interfaces = address.Assign(devices); // assign addresses

// Node1-Node2 setup
devices = p2p.Install(nodes.Get(1), nodes.Get(2)); // connecting nodes
address.SetBase("10.1.2.0", "255.255.255.252"); // Node1-Node2 addresses
interfaces = address.Assign(devices); // assign addresses

// setup ip routes
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
Step by step example
- iperf with ns-3 stack (III)

DceManagerHelper dceManager;
dceManager.Install (nodes);

DceApplicationHelper dce;
ApplicationContainer apps;
dce.SetStackSize (1 << 20);  // 1MB stack
dce.SetBinary ("iperf");  // Launch iperf client on node 0
dce.ResetArguments ();  // clean arguments
dce.ResetEnvironment ();  // clean environment
dce.AddArgument ("-c");  // client
dce.AddArgument ("10.1.2.2");  // target machine address
dce.AddArgument ("-i");  // interval
dce.AddArgument ("1");
dce.AddArgument ("--time");  // how long
dce.AddArgument ("10");
apps = dce.Install (nodes.Get (0));  // install application
apps.Start (Seconds (0.7));  // start at 0.7 simulation time
apps.Stop (Seconds (20));  // stop at 20s simulation time

DCE Setup
Step by step example  
- iperf with ns-3 stack (IV)

```cpp
// Simulation stop time
Simulator::Stop (Seconds (40.0));

// Run
Simulator::Run ();

// Stop
Simulator::Destroy ();

return 0;
}
```
Step by step example – iperf, ns-3

- **Generated**
  - elf-cache – program files
  - exitprocs – execution process information
  - files-0 files-2 – execution filesystem

- **files-x**
  - var – “/root” of the machine
  - files-x/var/log/<pid>/
    - cmdline – command executed
    - status – execution information
    - stderr – standard error output
    - stdout – standard output
    - syslog – syslog output
Step by step example  
- HTTP with ns-3 stack (I)

int main (int argc, char *argv[]) 
{ 
  // Node Container creation 
  NodeContainer nodes; 
  nodes.Create (3);

  // Linux stack creation 
  InternetStackHelper stack; 
  stack.Install (nodes);

  // For real time 
  // GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealtimeSimulatorImpl")); 
  // GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));

  // Device and channel creation 
  PointToPointHelper p2p; 
  p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps")); 
  p2p.SetChannelAttribute ("Delay", StringValue ("1ms"));
Step by step example
- HTTP with ns-3 stack (II)

// Node0-Node1 setup
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.252"); // Node0-Node1 addresses

NetDeviceContainer devices;
devices = p2p.Install (nodes.Get (0), nodes.Get (1)); // connecting nodes
Ipv4InterfaceContainer interfaces = address.Assign (devices); // assign addresses

// Node1-Node2 setup
devices = p2p.Install (nodes.Get (1), nodes.Get (2)); // connecting nodes
address.SetBase("10.1.2.0", "255.255.255.252"); // Node1-Node2 addresses
interfaces = address.Assign (devices); // assign addresses

// setup ip routes
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
Step by step example  
- HTTP with ns-3 stack (III)

```c++
// Launch the server HTTP
dce.SetBinary("httpd");
dce.ResetArguments();  // clean arguments
dce.ResetEnvironment();  // clean environment

dce.SetUid(1);  // Set httpd for super user execution

dce.SetEuid(1);
apps = dce.Install(nodes.Get(0));  // install http daemon
apps.Start(Seconds(1));  // start time

// Launch the client WGET

dce.SetBinary("wget");
dce.ResetArguments();  // clean arguments

dce.ResetEnvironment();  // clean environment

dce.AddArgument("-r");  // recursive wget
dce.AddArgument("http://10.1.1.1/index.html");
apps = dce.Install(nodes.Get(2));
apps.Start(Seconds(2));  // start time
```
Step by step example
- HTTP with ns-3 stack (IV)

```cpp
// Simulation stop time
Simulator::Stop (Seconds (40.0));

// Run
Simulator::Run ();

// Stop
Simulator::Destroy ();

return 0;
}
```
Step by step example
- iperf with linux stack (I)

int main (int argc, char *argv[]) {
    // Node Container creation
    NodeContainer nodes;
    nodes.Create (3);

    // Linux stack creation
    dceManager.SetNetworkStack ("ns3::LinuxSocketFdFactory", "Library", StringValue ("liblinux.so");
    LinuxStackHelper stack;
    stack.Install (nodes);

    // For real time
    // GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealtimeSimulatorImpl");
    // GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));

    // Device and channel creation
    PointToPointHelper p2p;
    p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps");
    p2p.SetChannelAttribute ("Delay", StringValue ("1ms"));
// Node0-Node1 setup
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.252"); // Node0-Node1 addresses

NetDeviceContainer devices;
devices = p2p.Install(nodes.Get(0), nodes.Get(1)); // connecting nodes
Ipv4InterfaceContainer interfaces = address.Assign(devices); // assign addresses

// Node1-Node2 setup
devices = p2p.Install(nodes.Get(1), nodes.Get(2)); // connecting nodes
address.SetBase("10.1.2.0", "255.255.255.252"); // Node1-Node2 addresses
interfaces = address.Assign(devices); // assign addresses

// setup ip routes
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
Step by step example
- iperf with linux stack (III)

DceManagerHelper dceManager;
dceManager.Install (nodes);

DceApplicationHelper dce;
ApplicationContainer apps;
dce.SetStackSize (1 << 20); // 1MB stack
dce.SetBinary ("iperf"); // Launch iperf client on node 0
dce.ResetArguments (); // clean arguments
dce.ResetEnvironment (); // clean environment
dce.AddArgument ("-c"); // client
dce.AddArgument ("10.1.2.2"); // target machine address
dce.AddArgument ("-i"); // interval
dce.AddArgument ("1");
dce.AddArgument ("--time"); // how long
dce.AddArgument ("10");
apps = dce.Install (nodes.Get (0)); // install application
apps.Start (Seconds (0.7)); // start at 0.7 simulation time
apps.Stop (Seconds (20)); // stop at 20s simulation time
dce.SetBinary ("iperf"); // Launch iperf server on node 2
dce.ResetArguments (); // clean arguments
dce.ResetEnvironment (); // clean environment
dce.AddArgument ("-s"); // server
dce.AddArgument ("-P"); // number of parallel servers
dce.AddArgument ("1");
apps = dce.Install (nodes.Get (2));
apps = dce.Install (nodes.Get (2));
apps.Start (Seconds (0.6));
Step by step example
- iperf with linux stack (IV)

// Simulation stop time
Simulator::Stop (Seconds (40.0));

// Run
Simulator::Run ();

// Stop
Simulator::Destroy ();

return 0;
}