Network Simulator 3 (Ns3) at Missouri S&T

Maciej Zawodniok

Introduction

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

News

- Upcoming deadlines:
  - Lab A.2 due Sept 20 (Fix GNU Radio graphs)
  - Homework 3 due on Sept 22nd **CLASSTIME**
  - Exam 1 – tentatively scheduled for Sept 24th
  - Project mid-term report - due on Oct 2nd
- A general note on the homework and lab assignments:
  - If an assumption is not explicitly specified, you can choose one of the options, BUT you need to write it in the solution!
- Early warning
  - Lab B-1 (Ns3) due on October 8th

MISSOURI S&T

Today's Outline

- **Ns3 intro**
- Simulation code
- Demo

Links and prerequisites

- Ns3 setup/install
  - http://www.nsnam.org
  - Installation procedure and required software:
    - http://www.nsnam.org/wiki/Installation
  - Suggested to use Linux system on your laptop/desktop
    - Mac OS X works fine too
    - Use Ubuntu/Linux on Windows 10 via WSL (lab 1&2)
- NetAnim – part of Ns3 download
  - https://www.nsnam.org/wiki/NetAnim_3.108

MISSOURI S&T

Compilation and Execution

- Download (ns-allinone-3.34 version)
- Unpack source

```
tar –xvf ns-allinone-3.34.tar.bz2
```

- Build the source first time

```
cd ns-allinone-3.34
./build.py
```

- Inside “ns3-34” folder, copy one of the examples to “scratch” subfolder

```
cd ns-3.34
cp examples/wireless/wifi-simple-adhoc.cc scratch
```

- Test the code (execute=rebuild + run)

cp scratch

```
../waf
../waf –-run wifi-simple-adhoc
```

  - New files should show up in the main folder (ns-3.34)

```
ls -altr
```

  - *.pcap files should be there (can be opened with “Wireshark”

Today's Outline

- Ns3 intro
- **Simulation code**
- Demo

Ns3 Simulation Examples

- Many examples available
  - "examples/wireless" folder
  - "wifi-hidden-terminal.cc"
    - It has good step-by-step explanation on how the simulation is setup;
    - Also, it has example of using FlowMonitor
  - "wifi-simple-adhoc-grid.cc"
    - There is example of setting a network grid and running a mulit-hop network
  - "wifi-adhoc.cc"
    - Example of manually positioning nodes

wifi-hidden-terminal.cc (1/12)

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2010 IITP RAS
 *
 * This program is free software; you can redistribute it and/or
modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 * Authors: Pavel Boyko <boyko@iitp.ru>
 */
```

MISSOURI S&T

wifi-hidden-terminal.cc (2/12)

```
/*
 * Classical hidden terminal problem and its RTS/CTS
solution.
 *
 * Topology: [node 0] <-- -50 dB --> [node 1] <-- -50
dB --> [node 2]
 *
 * This example illustrates the use of
 *  - Wifi in ad-hoc mode
 *  - Matrix propagation loss model
 *  - Use of OnOffApplication to generate CBR stream
 *  - IP flow monitor
 */
#include "ns3/core-module.h"
#include "ns3/propagation-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/wifi-module.h"
```

MISSOURI S&T

wifi-hidden-terminal.cc (3/12)

```
using namespace ns3;

/// Run single 10 seconds experiment with
    enabled or disabled RTS/CTS mechanism
void experiment (bool enableCtsRts,
    std::string wifiManager)
{
  // 0. Enable or disable CTS/RTS
  UintegerValue ctsThr = (enableCtsRts ?
    UintegerValue (100) : UintegerValue
    (2200));
  Config::SetDefault
    ("ns3::WifiRemoteStationManager::RtsCt
    sThreshold", ctsThr);

  // 1. Create 3 nodes
  NodeContainer nodes;
  nodes.Create (3);
```

wifi-hidden-terminal.cc (4/12)

```
// 2. Place nodes somehow, this is required by
     every wireless simulation
 for (size_t i = 0; i < 3; ++i)
   {
     nodes.Get (i)->AggregateObject
     (CreateObject<ConstantPositionMobilityModel>
     ());
   }
 // 3. Create propagation loss matrix
 Ptr<MatrixPropagationLossModel> lossModel =
     CreateObject<MatrixPropagationLossModel> ();
 lossModel->SetDefaultLoss (200); // set default
     loss to 200 dB (no link)
 lossModel->SetLoss (nodes.Get (0)-
     >GetObject<MobilityModel>(), nodes.Get (1)-
     >GetObject<MobilityModel>(), 50); // set
     symmetric loss 0 <-> 1 to 50 dB
 lossModel->SetLoss (nodes.Get (2)-
     >GetObject<MobilityModel>(), nodes.Get (1)-
     >GetObject<MobilityModel>(), 50); // set
     symmetric loss 2 <-> 1 to 50 dB
```

MISSOURI S&T

Propagation Loss Matrix Model

wifi-hidden-terminal.cc (5/12)

```
// 4. Create & setup wifi channel
 Ptr<YansWifiChannel> wifiChannel = CreateObject
    <YansWifiChannel> ();
 wifiChannel->SetPropagationLossModel (lossModel);
 wifiChannel->SetPropagationDelayModel (CreateObject
    <ConstantSpeedPropagationDelayModel> ());
 // 5. Install wireless devices
 WifiHelper wifi;
 wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
 wifi.SetRemoteStationManager ("ns3::" + wifiManager +
    "WifiManager");
 YansWifiPhyHelper wifiPhy=YansWifiPhyHelper::Default();
 wifiPhy.SetChannel (wifiChannel);
 WifiMacHelper wifiMac;
 wifiMac.SetType ("ns3::AdhocWifiMac"); // use ad-hoc MAC
 NetDeviceContainer devices = wifi.Install (wifiPhy,
    wifiMac, nodes);

 // uncomment the following to have athstats output
 // AthstatsHelper athstats;
 // athstats.EnableAthstats(enableCtsRts ? "rtscts-
    athstats-node" : "basic-athstats-node" , nodes);
```

MISSOURI S&T

wifi-hidden-terminal.cc (6/12)

```
// uncomment the following to have pcap output
// wifiPhy.EnablePcap (enableCtsRts ? "rtscts-
   pcap-node" : "basic-pcap-node" , nodes);


// 6. Install TCP/IP stack & assign IP addresses
InternetStackHelper internet;
internet.Install (nodes);
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.0.0.0", "255.0.0.0");
ipv4.Assign (devices);
```

MISSOURI S&T

wifi-hidden-terminal.cc (7/12)

```
// 7. Install applications: two CBR streams
   each saturating the channel
 ApplicationContainer cbrApps;
 uint16_t cbrPort = 12345;
 OnOffHelper onOffHelper
  ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address
  ("10.0.0.2"), cbrPort));
 onOffHelper.SetAttribute ("PacketSize",
  UintegerValue (1400));
 onOffHelper.SetAttribute ("OnTime",
  StringValue
  ("ns3::ConstantRandomVariable[Constant=1]"
  ));
 onOffHelper.SetAttribute ("OffTime",
  StringValue
  ("ns3::ConstantRandomVariable[Constant=0]"
  ));
```

MISSOURI S&T

wifi-hidden-terminal.cc (8/12)

```
// flow 1:  node 0 -> node 1
onOffHelper.SetAttribute ("DataRate",
 StringValue ("3000000bps"));
onOffHelper.SetAttribute ("StartTime", TimeValue
 (Seconds (1.000000)));
cbrApps.Add (onOffHelper.Install (nodes.Get
 (0)));

// flow 2:  node 2 -> node 1
/** \internal
 * The slightly different start times and data
 rates are a workaround
 * for \bugid{388} and \bugid{912}
 */
onOffHelper.SetAttribute ("DataRate",
 StringValue ("3001100bps"));
onOffHelper.SetAttribute ("StartTime", TimeValue
 (Seconds (1.001)));
cbrApps.Add (onOffHelper.Install (nodes.Get
 (2)));
```

MISSOURI S&T

wifi-hidden-terminal.cc (9/12)

```
/** \internal
  * We also use separate UDP applications that will send a single
  * packet before the CBR flows start.
  * This is a workaround for the lack of perfect ARP, see
   \bugid{187}
  */
 UdpEchoClientHelper echoClientHelper (Ipv4Address ("10.0.0.2"),
   echoPort);
 echoClientHelper.SetAttribute ("MaxPackets", UintegerValue (1));
 echoClientHelper.SetAttribute ("Interval", TimeValue(Seconds
   (0.1)));
 echoClientHelper.SetAttribute ("PacketSize", UintegerValue (10));
 ApplicationContainer pingApps;

 // again using different start times to workaround Bug 388 and
   Bug 912
 echoClientHelper.SetAttribute ("StartTime", TimeValue (Seconds
   (0.001)));
 pingApps.Add (echoClientHelper.Install (nodes.Get (0)));
 echoClientHelper.SetAttribute ("StartTime", TimeValue (Seconds
   (0.006)));
 pingApps.Add (echoClientHelper.Install (nodes.Get (2)));
```

MISSOURI S&T

wifi-hidden-terminal.cc (10/12)

```
// 8. Install FlowMonitor on all nodes
 FlowMonitorHelper flowmon;
 Ptr<FlowMonitor> monitor =
  flowmon.InstallAll ();

 // 9. Run simulation for 10 seconds
 Simulator::Stop (Seconds (10));
 Simulator::Run ();

 // 10. Print per flow statistics
 monitor->CheckForLostPackets ();
 Ptr<Ipv4FlowClassifier> classifier =
  DynamicCast<Ipv4FlowClassifier>
  (flowmon.GetClassifier ());
 FlowMonitor::FlowStatsContainer stats =
  monitor->GetFlowStats ();
```

MISSOURI S&T

wifi-hidden-terminal.cc (11/12)

```
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i =
        stats.begin (); i != stats.end (); ++i)
  {
    // first 2 FlowIds are for ECHO apps,we don't want to display them
    // Duration for throughput measurement is 9.0 seconds, since
    //   StartTime of the OnOffApplication is at about "second 1" and
    //   Simulator::Stops at "second 10".
    if (i->first > 2)
      {
        Ipv4FlowClassifier::FiveTuple t=classifier->FindFlow(i->first);
        std::cout << "Flow " << i->first - 2 << " (" << t.sourceAddress
          << " -> " << t.destinationAddress << ")\n";
        std::cout << "  Tx Packets: " << i->second.txPackets << "\n";
        std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
        std::cout << "  TxOffered:  " << i->second.txBytes * 8.0 / 9.0
          / 1000 / 1000  << " Mbps\n";
        std::cout << "  Rx Packets: " << i->second.rxPackets << "\n";
        std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
        std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / 9.0
          / 1000 / 1000  << " Mbps\n";
      } }
 // 11. Cleanup
  Simulator::Destroy ();
}
```

wifi-hidden-terminal.cc (12/12)

```
//-----------------------------------
int main (int argc, char **argv)
{
  std::string wifiManager ("Arf");
  CommandLine cmd (__FILE__);
  cmd.AddValue("wifiManager","Set wifi rate manager (Aarf,
     Aarfcd, Amrr, Arf, Cara, Ideal, Minstrel, Onoe, Rraa)",
     wifiManager);
  cmd.Parse (argc, argv);

  std::cout << "Hidden station experiment with RTS/CTS
     disabled:\n" << std::flush;
  experiment (false, wifiManager);
  std::cout << "-----------------------------------------\n";
  std::cout << "Hidden station experiment with RTS/CTS
     enabled:\n";
  experiment (true, wifiManager);

  return 0;
}
```

Today's Outline

- Ns3 intro
- Simulation code
- **Demo**

Program Completed

MISSOURI S&T

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Campus Linux Servers

- If needed – copy files to your home folder (or subfolder)
  - Missouri S&T Linux servers (ECE/CS)
    - rc01xcs213.managed.mst.edu
    - ….
    - List at: https://it.mst.edu/services/linux/hostnames/
  - Please check your account if you need one please contact instructor or IT
  - https://it.mst.edu/services/linux/

VPN to Campus Network Is Needed