Symbolic NS-3 for Exhaustive Testing

Jianfei Shao, Minh Vu, Mingrui Zhang, Lisong Xu

University of Nebraska-Lincoln

NS-3 Applications

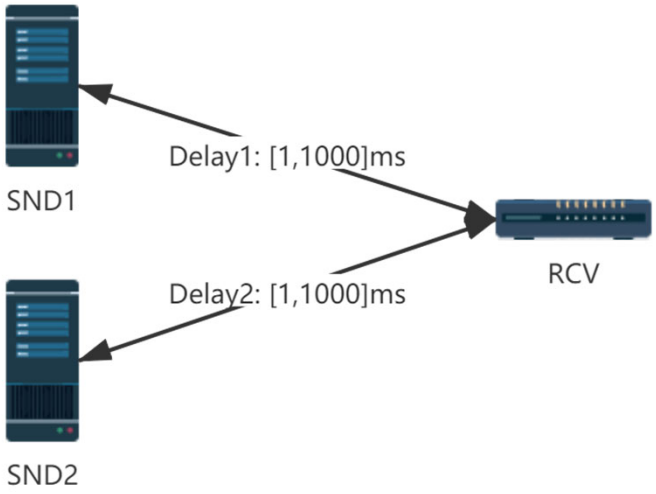
- Common NS-3 users:
  - Normal-case performance evaluation
    - Evaluate the performance of a network protocol in normal cases
- In our project:
  - Worst-case performance evaluation
    - Find the worst-case performance of a network protocol among all possible cases
  - Correctness testing
    - Test whether a network protocol works correctly in all possible cases
  - These two belong to the class of *exhaustive testing*

Outline

- Toy example for exhaustive testing
- Use current NS-3 for exhaustive testing
- Use our proposed symbolic NS-3 for exhaustive testing
- Implementation of symbolic NS-3
- Demo

Toy example (worst-case performance evaluation)

- SND1 and SND2 each sends a packet to RCV.
- Performance metric: Delay difference Diff = Delay1 – Delay2
- Worst-case performance: Find the maximum Diff
- Challenge: Total number of cases of Delay1 and Delay2: 1000*1000=10^6



Use current NS-3 for exhaustive testing

- NS-3 script: simulate for each input (delay1,delay2)
- Shell script: repeat the NS-3 script for all possible 10^6 inputs (delay1,delay2).

NS-3 script

```
main (int argc, char *argv[])
{
  Time::SetResolution (Time::MS);

  uint32_t delay1 = 0;
  uint32_t delay2 = 0;
  CommandLine cmd (__FILE__);
  cmd.AddValue ("delay1", "The delay for link between snd1 and rcv.", delay1);
  cmd.AddValue ("delay2", "The delay for link between snd2 and rcv.", delay2);
  cmd.Parse (argc, argv);
```

- Initialize Delay

```
  std::vector<PointToPointHelper> pointToPoint (2);
  pointToPoint[0].SetChannelAttribute ("Delay", TimeValue (Time (delay1)));
  pointToPoint[1].SetChannelAttribute ("Delay", TimeValue (Time (delay2)));
```

- Set up Delay

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

  std::vector<NodeContainer> nodeAdjacencyList (2);
  nodeAdjacencyList[0] = NodeContainer (nodes.Get (0), nodes.Get (2));
  nodeAdjacencyList[1] = NodeContainer (nodes.Get (1), nodes.Get (2));

  std::vector<NetDeviceContainer> devices (2);
  devices[0] = pointToPoint[0].Install (nodeAdjacencyList[0]);
  devices[1] = pointToPoint[1].Install (nodeAdjacencyList[1]);

  InternetStackHelper stack;
  stack.Install (nodes);
```

- Create Topology

```
  Ipv4AddressHelper address;
  std::vector<Ipv4InterfaceContainer> interfaces (2);
  for (uint32_t i = 0; i < 2; i++)
    {
      std::ostringstream subset;
      subset << "10.1." << i + 1 << ".0";
      address.SetBase (subset.str ().c_str (), "255.255.255.0");
      interfaces[i] = address.Assign (devices[i]);
    }
```

```
  UdpServerHelper server (2333);

  ApplicationContainer rcv = server.Install (nodes.Get (2));
  rcv.Start (Seconds (1.0));
  rcv.Stop (Seconds (10.0));

  UdpClientHelper snd1 (interfaces[0].GetAddress (1), 2333);
  snd1.SetAttribute ("MaxPackets", UintegerValue (1));

  UdpClientHelper snd2 (interfaces[1].GetAddress (1), 2333);
  snd2.SetAttribute ("MaxPackets", UintegerValue (1));

  ApplicationContainer snd;
  snd.Add(snd1.Install (nodes.Get (0)));
  snd.Add(snd2.Install (nodes.Get (1)));
  snd.Start (Seconds (1.0));
  snd.Stop (Seconds (10.0));
```

- Install Applications

```
  Simulator::Run ();
  Time Diff = Time(delay1)-Time(delay2);
  std::cout<<"Diff is "<<Diff<<std::endl;
  Simulator::Destroy ();
  return 0;
```

Shell script

```
#!/bin/bash
for delay1 in {1..1000}
do
    for delay2 in {1..1000}
    do
    ./waf --run "currentDemo --delay1=$delay1 --delay2=$delay2">>log.out 2>&1
    done
done
```

Symbolic NS-3

- Goal:
  - Efficient exhaustive testing.
- How:
  - Simulate a group of inputs together instead of individually
  - Leverage symbolic execution: S2E

Using Symbolic NS-3 for Exhaustive Testing

- NS-3 script: simulate for symbolic input
  - Different inputs:
    - Current NS-3:concrete input
      - e.g. (delay1 = 1ms, delay2 = 2ms)
    - Symbolic NS-3:symbolic input
      - Definition: a group of concrete inputs
      - e.g. (delay1=[1,1000]ms,delay2=[1,1000]ms)
- Do not need a Linux shell script

Symbolic NS-3 script

```
main (int argc, char *argv[])
{
  Time::SetResolution (Time::MS);

  Ptr<Symbolic> sym1 = CreateObject<Symbolic>();
  sym1->SetAttribute("Min",UintegerValue(1));
  // We have Muitlple way to set a Max or Min.
  // sym1->SetAttribute("Min",UintegerValue(Time(1ms).GetTimeStep()));
  // sym1->SetMin(1);
  // sym1->SetMin(Second(1));
  // sym1->SetMin(Time("1ms"));
  sym1->SetAttribute("Max",UintegerValue(1000));

  Ptr<Symbolic> sym2 = CreateObject<Symbolic>();
  sym2->SetAttribute("Min",UintegerValue(1));
  sym2->SetAttribute("Max",UintegerValue(1000));

  std::vector<PointToPointHelper> pointToPoint (2);
  pointToPoint[0].SetChannelAttribute ("SymbolicDelay", PointerValue (sym1));
  pointToPoint[1].SetChannelAttribute ("SymbolicDelay", PointerValue (sym2));

  NodeContainer nodes;
  nodes.Create (3);

  std::vector<NodeContainer> nodeAdjacencyList (2);
  nodeAdjacencyList[0] = NodeContainer (nodes.Get (0), nodes.Get (2));
  nodeAdjacencyList[1] = NodeContainer (nodes.Get (1), nodes.Get (2));

  std::vector<NetDeviceContainer> devices (2);
  devices[0] = pointToPoint[0].Install (nodeAdjacencyList[0]);
  devices[1] = pointToPoint[1].Install (nodeAdjacencyList[1]);

  InternetStackHelper stack;
  stack.Install (nodes);
```

1 •Initialize Delay

2 •Set up Delay

•Create Topology

```
  Ipv4AddressHelper address;
  std::vector<Ipv4InterfaceContainer> interfaces (2);
  for (uint32_t i = 0; i < 2; i++)
    {
      std::ostringstream subset;
      subset << "10.1." << i + 1 << ".0";
      address.SetBase (subset.str().c_str (), "255.255.255.0");
      interfaces[i] =
          address.Assign (devices[i]);
    }

  UdpServerHelper server (2333);

  ApplicationContainer rcv = server.Install (nodes.Get (2));
  rcv.Start (Seconds (1.0));
  rcv.Stop (Seconds (10.0));

  UdpClientHelper snd1 (interfaces[0].GetAddress (1), 2333);
  snd1.SetAttribute ("MaxPackets", UintegerValue (1));

  UdpClientHelper snd2 (interfaces[1].GetAddress (1), 2333);
  snd2.SetAttribute ("MaxPackets", UintegerValue (1));

  ApplicationContainer snd;
  snd.Add(snd1.Install (nodes.Get (0)));
  snd.Add(snd2.Install (nodes.Get (1)));
  snd.Start (Seconds (1.0));
  snd.Stop (Seconds (10.0));

  Simulator::Run ();
  Symbolic Diff=sym1-sym2;
  Diff.PrintRange("Diff");
  Simulator::Destroy ();
  s2e_kill_state(0,"Program Terminated");
  return 0;
```

•Install Applications

Difference 1: Create delay variable

- Current NS-3:
  - Create concrete Value
    - e.g. (delay1 = 1ms, delay2 = 2ms)

```
uint32_t delay1 = 0;
uint32_t delay2 = 0;
CommandLine cmd (__FILE__);
cmd.AddValue ("delay1", "The delay for link between snd1 and rcv.", delay1);
cmd.AddValue ("delay2", "The delay for link between snd2 and rcv.", delay2);
cmd.Parse (argc, argv);
```

- Symbolic NS3:
  - Create Symbolic Value
    - a group of concrete inputs
    - delay1=[1,1000]ms,delay2=[1,1000]ms

```
Ptr<Symbolic> sym1 = CreateObject<Symbolic>();
sym1->SetAttribute("Min",UintegerValue(1));
// We have Muitlple way to set a Max or Min.
// sym1->SetAttribute("Min",UintegerValue(Time(1ms).GetTimeStep()));
// sym1->SetMin(1);
// sym1->SetMin(Second(1));
// sym1->SetMin(Time("1ms"));
sym1->SetAttribute("Max",UintegerValue(1000));

Ptr<Symbolic> sym2 = CreateObject<Symbolic>();
sym2->SetAttribute("Min",UintegerValue(1));
sym2->SetAttribute("Max",UintegerValue(1000));
```

Difference 2: Setup link delay

- Current NS-3:
  - Set up concrete delay

```
std::vector<PointToPointHelper> pointToPoint (2);
pointToPoint[0].SetChannelAttribute ("Delay", TimeValue (Time (delay1)));
pointToPoint[1].SetChannelAttribute ("Delay", TimeValue (Time (delay2)));
```

- Symbolic NS-3
  - Set up symbolic delay

```
std::vector<PointToPointHelper> pointToPoint (2);
pointToPoint[0].SetChannelAttribute ("SymbolicDelay", PointerValue (sym1));
pointToPoint[1].SetChannelAttribute ("SymbolicDelay", PointerValue (sym2));
```

Running Time

- Current NS-3:
  - Time for one input (delay1,delay2):
    - 0.5 seconds.
  - Total time:
    - 0.5 seconds * 10^6 = 6 days.

- Symbolic NS-3:
  - Total time:
    - 33 seconds

Output

- Current NS-3 :

  Diff is +0 ms

  Diff is -1 ms

  Diff is -2 ms

  .....

  Diff is +999 ms

  .....

  Diff is +0 ms

- Maximum delay difference: 999 ms

- Symbolic NS-3 :

  The range of Diff is [-999,-1]

  The range of Diff is [1,999]

  The range of Diff is [0,0]

- Maximum delay difference: 999 ms

Implementation

- New classes:
  - Classes: Symbolic
  - Goal: as general as possible
- Changes to existing classes:
  - Classes: Point-to-point
  - Goal: modified current NS-3 as little as possible

New class: Symbolic

Goal: as general as possible

- Attribute:
  - m_symbolic
  - m_min
  - m_max
- Functions:
  - Symbolic packet delay:
    - GetSymbolicTime()
  - Symbolic packet header:
    - GetSymbolicIpv4Address()
  - Symbolic range:
    - SetMin()
    - SetMax()
    - PrintRange()

Modification to existing classes

Goal: modified current NS-3 as little as possible

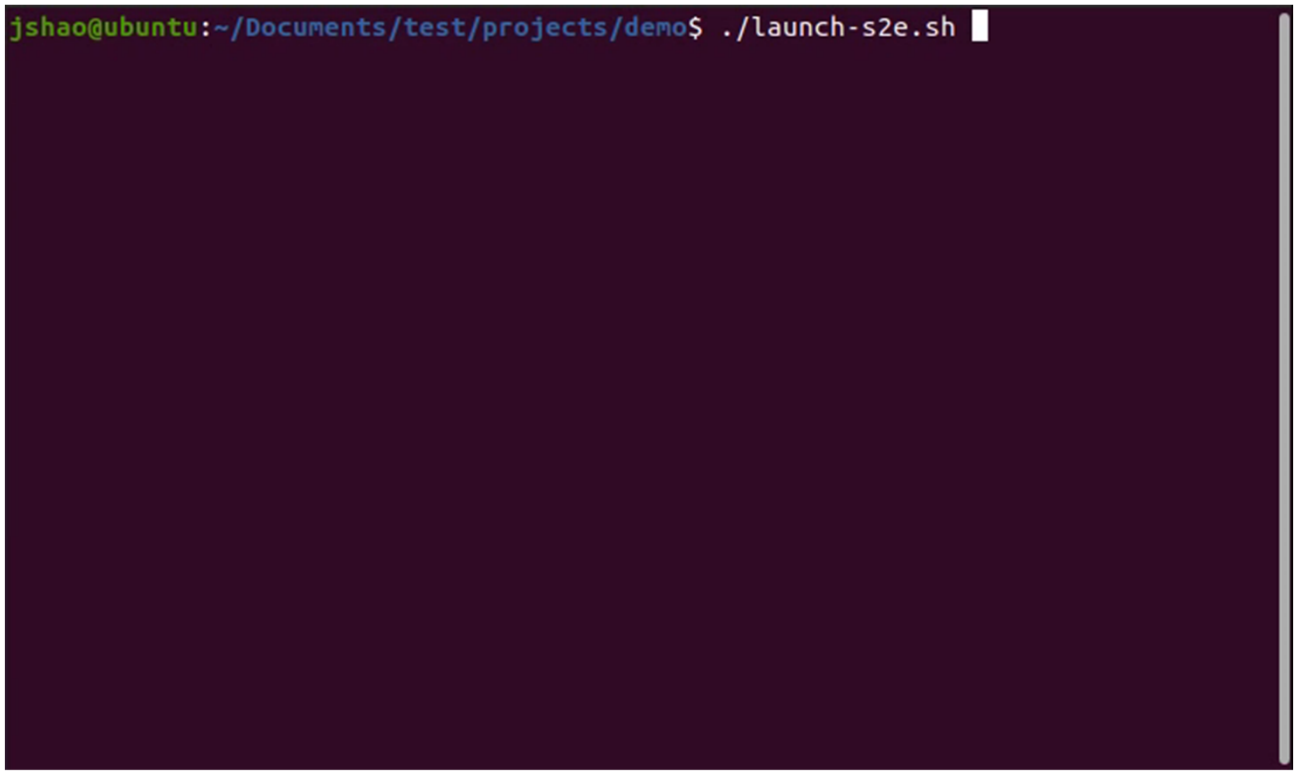
```
private:
  /** Each point to point link has exactly two net devices. */
  static const std::size_t N_DEVICES = 2;

  Time          m_delay;    //!< Propagation delay
  std::size_t        m_nDevices; //!< Devices of this channel
  #ifdef SYMBOLIC
  Ptr<Symbolic> m_symbolicDelay;  //!< Symbolic propagation delay
  #endif
```

↑ Add one attribute m_symbolicdelay to Point-To-Point link

Replace the propagation delay →

```
bool
PointToPointChannel::TransmitStart (
  Ptr<const Packet> p,
  Ptr<PointToPointNetDevice> src,
  Time txTime)
{
  NS_LOG_FUNCTION (this << p << src);
  NS_LOG_LOGIC ("UID is " << p->GetUid () << ")");

  NS_ASSERT (m_link[0].m_state != INITIALIZING);
  NS_ASSERT (m_link[1].m_state != INITIALIZING);

  uint32_t wire = src == m_link[0].m_src ? 0 : 1;
  //if No pointer then we use the normal ns3
  #ifdef SYMBOLIC
  if(m_symbolicDelay>0)
  {
    m_delay = m_symbolicDelay->GetSymbolicTime();
  }

  #endif

  Simulator::ScheduleWithContext (m_link[wire].m_dst->GetNode ()->GetId (),
                                  txTime + m_delay, &PointToPointNetDevice::Receive,
                                  m_link[wire].m_dst, p->Copy ());

  // Call the tx anim callback on the net device
  m_txrxPointToPoint (p, src, m_link[wire].m_dst, txTime, txTime + m_delay);
  return true;
}
```

Demo: running symbolic NS-3 for toy example



Important links:

- Symbolic Network Simulator
  - https://cse.unl.edu/~xu/research/SymbolicNS3.html
- S2E platform:
  - http://s2e.systems/
- Demo:
  - https://github.com/JeffShao96/Symbolic-NS3

Thank You