NS-2 Tutorial

Kameswari Chebrolu
Dept. of Electrical Engineering, IIT Kanpur

Motivation for Simulations
- Cheap -- does not require costly equipment
- Complex scenarios can be easily tested
- Results can be quickly obtained – more ideas can be tested in a smaller timeframe
- The real thing isn't yet available
- Controlled experimental conditions
  - Repeatability helps aid debugging
- Disadvantages: Real systems too complex to model

Features of NS-2
- Protocols: TCP, UDP, HTTP, Routing algorithms etc
- Traffic Models: CBR, VBR, Web etc
- Error Models: Uniform, bursty etc
- Radio propagation, Mobility models
- Energy Models
- Topology Generation tools
- Visualization tools
- Extensibility

NS Structure
- NS is an object oriented simulator
- Back end is C++ event scheduler
  - Protocols mostly
- Front end is oTCL
  - Creating scenarios, extensions to C++ protocols
  - Objects created in oTCL have a corresponding object in C++
TCL tutorial

- Variables:
  - set x 1
  - set y $x

- Arrays:
  - set a(0) 1

- Printing:
  - puts “a(0) \n”

- Arithmetic Expression:
  - set z = [expr $y + 5]

- Control Structures:
  - if {$z == 6} then { puts “Correct!” }
    - for {set i =0} {$i < 5} {incr i }{
      - puts “$i * $i equals [expr $i * $i]”
    }

- Procedures:
  - proc sum {a b} {
    - return [expr $a + $b]
  }

How to Start?

- Create simulator object: set ns [new simulator]
- Open a file for writing data for input to nam (network animator)
  - set nf [open out.nam w]
  - $ns namtrace-all $nf
- Finish procedure:
  - proc finish {} {
    - global ns nf
    - close $nf
    - exec nam out.nam &
    - exit 0
  }

Creating topology

- Two nodes connected by a link
- Creating nodes
  - set n0 [$ns node]
  - set n1 [$ns node]
- Creating link between nodes
  - $ns duplex-link $n0 $n1 1Mb 10ms DropTail
Sending data

- Create UDP agent
  ```
  set udp0 [new Agent/UDP]
  $ns attach-agent $n0 $udp0
  ```

- Create CBR traffic source for feeding into UDP agent
  ```
  set cbr0 [new Application/Traffic/CBR]
  $cbr0 set packetSize_ 500
  $cbr0 set interval_ 0.005
  $cbr0 attach-agent $udp0
  ```

- Create traffic sink
  ```
  set null0 [new Agent/Null]
  $ns attach-agent $n1 $null0
  ```

- Connect two agents
  ```
  $ns connect $udp0 $null0
  ```

- Start and stop of data
  ```
  $ns at 0.5 "$cbr0 start"
  $ns at 4.5 "$cbr0 stop"
  ```

Creating TCP Connections

- Create TCP agent and attach it to the node
  ```
  set tcp0 [new Agent/TCP]
  $ns attach-agent $n0 $tcp0
  ```

- Create a Null Agent and attach it to the node
  ```
  set null0 [new Agent/TCPSink]
  $ns attach-agent $n1 $null0
  ```

- Connect the agents
  ```
  $ns connect $tcp0 $null0
  ```

Traffic on top of TCP

- FTP
  ```
  set ftp [new Application/FTP]
  $ftp attach-agent $tcp0
  ```

- Telnet
  ```
  set telnet [new Application/Telnet]
  $telnet attach-agent $tcp0
  ```
Introducing Errors

- Creating Error Module
  - set err [new ErrorModel]
  - $err unit pkt
  - $err set rate_ 0.01
  - $err ranvar [new RandomVariable/Uniform]
  - $err drop-target [new Agent/Null]

- Inserting Error Module
  - $ns lossmodel $err $n0 $n1

Summary

- Simulators help in easy verification of protocols in less time, money
- NS offers support for simulating a variety of protocol suites and scenarios
- Front end is oTCL, back end is C++
- NS is an on-going effort of research and development

Tracing

- All packet trace
  - $ns trace-all [open out.tr w]
    - <event> <time> <from> <to> <pkt> <size>
    - <flowid> <src> <dst> <seqno> <aseqno>
      - + 0.51 0 1 cbr 500 ------- 0 0.0 1.0 0 2
      - - 0.51 0 1 cbr 500 ------- 0 0.0 1.0 0 2
      - r 0.514 0 1 cbr 500 ------- 0 0.0 1.0 0 0

- Variable trace
  - set par [open output/param.tr w]
  - Step attach $par
  - Step trace cwnd
  - Step trace maxseq
  - Step trace rtt