CS378: Computer Networks Lab

Network Simulator

Slides borrowed from:
Kameswari Chebrolu
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, MAC etc
- Traffic Models: CBR, VBR, Web etc
- Error Models: Uniform, bursty etc
- Misc: Radio propagation, Mobility models, Energy Models
- Topology Generation tools
- Visualization tools (NAM), Tracing
NS Structure

- NS is an object oriented discrete-event simulator
  - Simulator maintains list of events and executes one event after another
  - Single thread of control: no locking or race conditions

- Back end is C++ event scheduler
  - Protocols mostly
  - Fast to run, more control

- Front end is oTCL
  - Creating scenarios, extensions to C++ protocols
  - fast to write and change
TCL tutorial

- **Variables:**
  
  ```tcl
  set x 1
  set y $x
  ```

- **Arrays:**
  
  ```tcl
  set a(0) 1
  ```

- **Printing:**
  
  ```tcl
  puts "$a(0) \n"
  ```

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

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

- **Procedures:**
  
  ```tcl
  proc sum {a b} {
    return [expr $a + $b]
  }
  ```
NS programming Structure

- Create the event scheduler
- Turn on tracing
- Create network topology
- Create connections
- Generate traffic
- Insert errors etc
Creating Event Scheduler

• Create event scheduler: set ns [new simulator]

• Schedule an event: $ns at <time> <event>
  – event is any legitimate ns/tcl function

  ```tcl
  proc finish {} {
    global ns nf
    close $nf
    exec nam out.nam &
    exit 0
  }
  ```

• Start Scheduler

  ```tcl
  $ns run
  ```
Tracing

- All packet trace

$ns trace-all [open out.tr w]

<event> <time> <from> <to> <pkt> <size>

------
<flowid> <src> <dst> <seqno> <aseqno>

+ 1.84375 0 2 cbr 210 ------ 0 0.0 3.1 225 610
- 1.84375 0 2 cbr 210 ------ 0 0.0 3.1 225 610
r 1.84471 2 1 cbr 210 ------ 1 3.0 1.0 195 600
r 1.84566 2 0 ack 40 ------ 2 3.2 0.1 82 602
+ 1.84566 0 2 tcp 1000 ------ 2 0.1 3.2 102 611
- 1.84566 0 2 tcp 1000 ------ 2 0.1 3.2 102 611
r 1.84609 0 2 cbr 210 ------ 0 0.0 3.1 225 610
+ 1.84609 2 3 cbr 210 ------ 0 0.0 3.1 225 610
d 1.84609 2 3 cbr 210 ------ 0 0.0 3.1 225 610
Tracing

- Variable trace

```bash
set par [open output/param.tr w]
$tcp attach $par
$tcp trace cwnd_
$tcp trace maxseq_
$tcp trace rtt_
```
Tracing and Animation

- Queue monitoring, Flow monitoring
- Network Animator

```bash
set nf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
    global ns nf
    close $nf
    exec nam out.nam &
    exit 0
}
```
Creating topology

- Two nodes connected by a link
- Creating nodes
  ```bash
  set n0 [$ns node]
  set n1 [$ns node]
  ```
- Creating link between nodes
  ```bash
  $ns <link_type> $n0 $n1 <bandwidth> <delay> <queue-type>
  ```

```bash
$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
  ```
Sending data

- 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
NS Internals

Link n0-n1

Port Classifier
Addr Classifier

Entry
0
1

Agent/TCP
dst_=1.0

Application/FTP
dst_=1.0

Link n1-n0

Port Classifier
Addr Classifier

Entry
0
1

Agent/TCPSink
dst_=0.0
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 (migrated to ns3)
Reference Material

- http://www.isi.edu/nsnam/ns/
  - Marc Greis' tutorial
  - Jae Chung tutorial
  - Ns manual