NS2 Simulation

Performance Modeling Lecture #9

Slides adapted from Polly Huang and Mark Claypool

Discrete-Event Simulations (1 of 3)

- Continuous events are simulations like weather or chemical reactions, while computers usually discrete events
- Typical components:
- *Event scheduler* – linked list of events
  - Schedule event *X* at time *T*
  - Hold event *X* for interval dt
  - Cancel previously scheduled event *X*
  - Hold event *X* indefinitely until scheduled by other event
  - Schedule an indefinitely scheduled event
  - Note, event scheduler executed often, so has significant impact on performance

Discrete-Event Simulations (1 of 3)

- *Simulation clock and time advancing*
  - Global variable with time
  - Scheduler advances time
    - *Unit time* – increments time by small amount and see if any events
    - *Event-driven* – increments time to next event and executes (typical)
- *System state variables*
  - Global variables describing state
  - Can be used to save and restore

Discrete-Event Simulations (2 of 3)

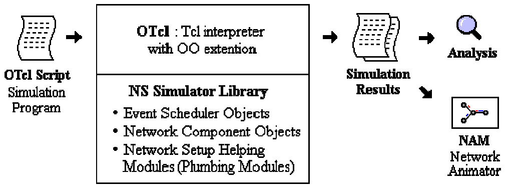
- *Event routines*
  - Specific routines to handle event
  - Ex: job arrival, job scheduling, job departure
  - Often handled by call-back from event scheduler
- *Input routines*
  - Get input from user (or config file, or script)
  - Often get all input before simulation starts
  - May allow range of inputs (from 1-9 ms) and number or repetitions, etc.

Discrete-Event Simulations (3 of 3)

- *Report generators*
  - Routines executed at end of simulation, final result and print
  - Can include graphical representation, too
  - Ex: may compute total wait time in queue or number of processes scheduled

Discrete Event Simulation Example NS - (1 of 4)

- NS-2, network simulator
  - Government funded initially, Open source
- Wildly popular for IP network simulations



OTcl Script Simulation Program

OTcl : Tcl interpreter with OO extention

NS Simulator Library
- Event Scheduler Objects
- Network Component Objects
- Network Setup Helping Modules (Plumbing Modules)

Simulation Results

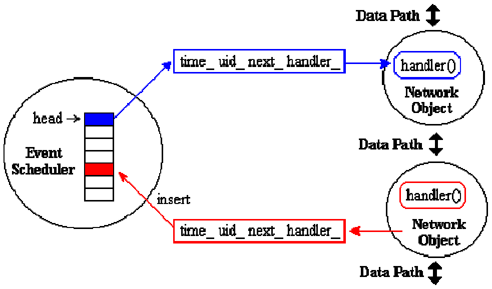
Analysis

NAM Network Animator

(http://perform.wpi.edu/NS/)

Discrete Event Simulation Example NS - (2 of 4)

(Event scheduler is core of simulator)



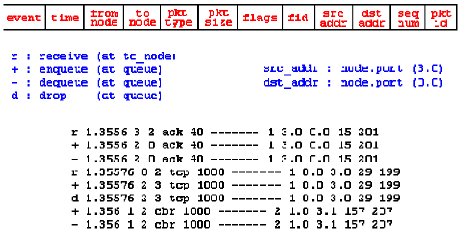
Discrete Event Simulation Example NS - (3 of 4)

```
# Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

# Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        # Close the trace file
        close $nf
        #Execute NAM on file
        exec nam out.nam &
        exit 0
}
```

```
# Setup a FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

# Initial schedule events
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

# Finish after 5 sec (sim time)
$ns at 5.0 "finish"

# Run the simulation
$ns run
```

Discrete Event Simulation Example NS - (4 of 4)

- Output in text file, can be processed with Unix command line tools



event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)                    src_addr : node.port (3.0)
- : dequeue (at queue)                    dst_addr : node.port (0.0)
d : drop    (at queue)
```

```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

(Objects and script can have custom output, too)

ns-2

- Discrete event simulator
- Packet level
- Link layer and up
- Wired and wireless

Development Status

- Columbia NEST
- UCB REAL
- ns-1
- ns-2 (as of 2001…)
  - 100K lines of C++ code
  - 70K lines of otcl support code
  - 30K lines of test suites
  - 20K lines of documentation

First Words of Caution

- While we have considerable confidence in ns, ns is **not a polished** and finished product, but the result of an ongoing effort of research and development. In particular, bugs in the software are still being discovered and corrected.

Second Words of Caution

- Users of ns are responsible for verifying for themselves that their simulations are not invalidated by **bugs**. We are working to help the users with this by significantly expanding and automating the validation tests and demos.

Third Words of Caution

- Similarly, users are responsible for verifying for themselves that their simulations are not invalidated because the **model** implemented in the simulator is not the model that they were expecting. The ongoing ns Notes and Documentation should help in this process.

Object-Oriented

+ Reusability
+ Maintainability

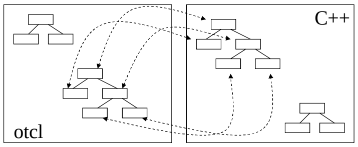
– Careful planning ahead
– Performance

C++ and otcl Separation

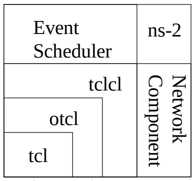
- C++ for data
  - per packet action
- otcl for control
  - periodic or triggered action

+ Compromize between composibility and speed
– Learning & debugging

otcl and C++: The Duality



tcl Interpreter With Extents



- otcl: object-oriented support
- tclcl: C++ and otcl linkage
- Discrete event scheduler
- Data network components

Hello World - Interactive Mode

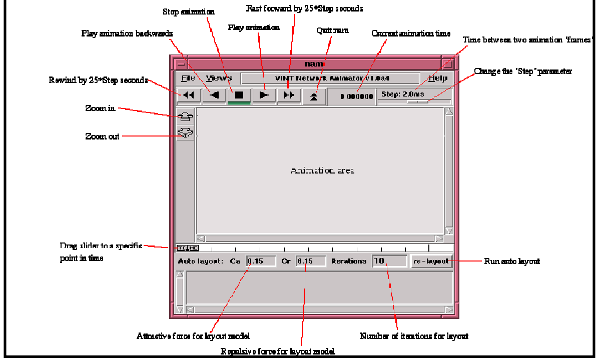
```
swallow 71% ns
% set ns [new Simulator]
_o3
% $ns at 1 “puts \“Hello World!\””
1
% $ns at 1.5 “exit”
2
% $ns run
Hello World!
swallow 72%
```

Hello World - Passive Mode

simple.tcl

```
   set ns [new Simulator]
   $ns at 1 “puts \“Hello World!\””
   $ns at 1.5 “exit”
   $ns run
swallow 74% ns simple.tcl
Hello World!
swallow 75%
```

nam


Stop animation
Fast forward by 25*Step seconds
Play animation backwards
Play animation
Quit nam
Current animation time
Time between two animation 'frames'
Rewind by 25*Step seconds
Change the 'Step' parameter
Zoom in
Zoom out
nam
File
Views
VINT Network Animator v1.0a4
Help
0.000000
Step: 2.0ms
Animation area
Drag slider to a specific point in time
TIME
Auto layout:
Ca
0.15
Cr
0.15
Iterations
10
re-layout
Run auto layout
Attractive force for layout model
Repulsive force for layout model
Number of iterations for layout

Fundamentals

- tcl
- otcl
  - ftp://ftp.tns.lcs.mit.edu/pub/otcl/doc/tutorial.html
- ns-2
  - http://www.isi.edu/nsnam/ns/ns_doc.ps.gz
  - http://www.isi.edu/nsnam/ns/ns_doc.pdf
  - http://www.isi.edu/nsnam/ns/doc/index.html

Basic tcl

```
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
          if {$k < 5} {
                   puts “k < 5, pow= [expr pow($d, $k)]”
          } else {
                   puts “k >= 5, mod= [expr $d % $k]”
          }
    }
}
test
```

Basic otcl

```
Class mom
mom instproc init {age} {
    $self instvar age_
    set age_ $age
}

mom instproc greet {} {
    $self instvar age_
    puts “$age_ years old mom:
    How are you doing?”
}
```

```
Class kid -superclass mom
kid instproc greet {} {
    $self instvar age_
    puts “$age_ years old kid:
    What’s up, dude?”
}

set a [new mom 45]
set b [new kid 15]

$a greet
$b greet
```

Basic ns-2

- Creating the event scheduler
- [Tracing]
- Creating network
- Computing routes
- Creating connection
- Creating traffic
- Inserting errors

Creating Event Scheduler

- Create scheduler
  - set ns [new Simulator]
- Schedule event
  - $ns at <time> <event>
  - <event>: any legitimate ns/tcl commands
- Start scheduler
  - $ns run

Tracing

- Trace packets on all links
  - $ns trace-all [open test.out w]

```
<event> <time> <from> <to> <pkt> <size>--<flowid> <src> <dst> <seqno> <aseqno>
     + 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
     - 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
     r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0
```

- Trace packets on all links in nam-1 format
  - $ns namtrace-all [open test.nam w]
- Right after ‘set ns [new Simulator]’

Creating Network

- Nodes
  - set n0 [$ns node]
  - set n1 [$ns node]
- Links & Queuing
  - $ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>
  - <queue_type>: DropTail, RED, CBQ, FQ, SFQ, DRR

Tracing Specific links

- $ns trace-queue $n0 $n1
- $ns namtrace-queue $n0 $n1

Creating Network: LAN

- LAN
  - $ns make-lan <node_list> <bandwidth> <delay> <ll_type> <ifq_type> <mac_type> <channel_type>
  - <ll_type>: LL
  - <ifq_type>: Queue/DropTail,
  - <mac_type>: MAC/802_3
  - <channel_type>: Channel

Computing routes

- Unicast
  - $ns rtproto <type>
  - <type>: Static, Session, DV, cost, multi-path

Creating Connection: UDP

- UDP
  - set udp [new Agent/UDP]
  - set null [new Agent/NULL]
  - $ns attach-agent $n0 $udp
  - $ns attach-agent $n1 $null
  - $ns connect $udp $null

Creating Connection: TCP

- TCP
  - set tcp [new Agent/TCP]
  - set tcpsink [new Agent/TCPSink]
  - $ns attach-agent $n0 $tcp
  - $ns attach-agent $n1 $tcpsink
  - $ns connect $tcp $tcpsink

Creating Traffic: On Top of TCP

- FTP
  - set ftp [new Application/FTP]
  - $ftp attach-agent $tcp
  - $ns at <time> “$ftp start”
- Telnet
  - set telnet [new Application/Telnet]
  - $telnet attach-agent $tcp

Creating Traffic: On Top of UDP

- CBR
  - set src [new Application/Traffic/CBR]
- Exponential or Pareto on-off
  - set src [new Application/Traffic/Exponential]
  - set src [new Application/Traffic/Pareto]

Creating Traffic: Trace Driven

- Trace driven
  - set tfile [new Tracefile]
  - $tfile filename <file>
  - set src [new Application/Traffic/Trace]
  - $src attach-tracefile $tfile
- <file>:
  - Binary format
  - inter-packet time (msec) and packet size (byte)

Inserting Errors

- Creating Error Module
  - set loss_module [new ErrorModel]
  - $loss_module set rate_ 0.01
  - $loss_module unit pkt
  - $loss_module ranvar [new RandomVariable/Uniform]
  - $loss_module drop-target [new Agent/Null]
- Inserting Error Module
  - $ns lossmodel $loss_module $n0 $n1

Network Dynamics

- Link failures
  - route changes reflected automatically
  - can emulate node failure

Four Models

- $ns rtmodel-at <time> <up|down> $n0 $n1
- $ns rtmodel Trace <config_file> $n0 $n1
- $ns rtmodel <model> <params> $n0 $n1
- <model>: Deterministic, Exponential
- <params>: [<start>] <up_interval> <down_interval> [<finish>]

Outlines

- Essentials
- Getting Started
- Fundamental tcl, otcl and ns-2
- **Case Studies - TCP, web traffic, RED**

Case Studies

- TCP (tcp.tcl)
- Web (web.tcl & dumbbell.tcl)
- Queuing - RED (red.tcl)

Visualization Tools

- nam-1 (Network AniMator Version 1)
- xgraph

Basic ns-2: Special Topics

- multicast support
- application-level support
- wireless support

Multicast - 5 components

- enable multicast capability
- configure multicast routing
- create a multicast group/sender
- create a multicast receiver
- attach traffic source

Enabling multicast capability

- set ns [new Simulator -multicast on]
- or $ns multicast (right after [new Simulator])

Configuring multicast routing

- $ns mrtproto <type>
- <type>: CtrMcast, DM, ST, BST

Creating a multicast group

- set udp [new Agent/UDP]
- $ns attach-agent $n0 $udp
- set group [Node allocaddr]
- $udp set dst_addr_ $group

Creating a multicast receiver

- set rcvr [new Agent/NULL]
- $ns attach-agent $n1 $rcvr
- $ns at <time> “$n1 join-group $rcvr $group”

Attaching a traffic source

- set cbr [new Application/Traffic/CBR]
- $cbr attach-agent $udp
- $ns at <time> “$cbr start”

Application - 2 components

- two-way TCP
- Application/TcpApp

Application: Two-way TCP

- FullTcp connection
  - set tcp1 [new Agent/TCP/FullTcp]
  - set tcp2 [new Agent/TCP/FullTcp]
  - $ns attach-agent $n1 $tcp1
  - $ns attach-agent $n2 $tcp2
  - $ns connect $tcp1 $tcp2
  - $tcp2 listen

Application: TcpApp

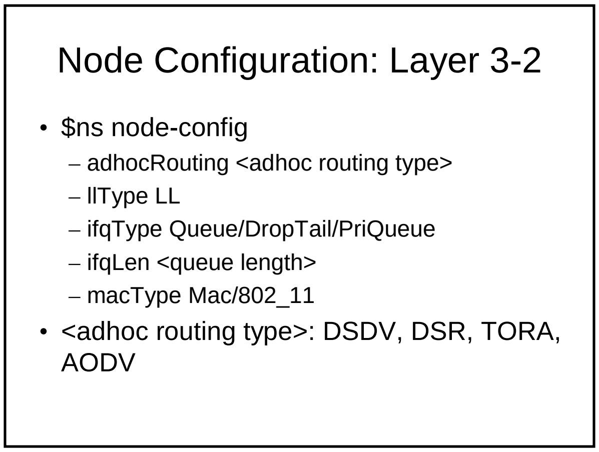
- User data transfer
  - set app1 [new Application/TcpApp $tcp1]
  - set app2 [new Application/TcpApp $tcp2]
  - $app1 connect $app2
  - $ns at 1.0 “$app1 send <data_byte> \”<ns-2 command>\””
  - <ns-2 command>: will be executed when received at the receiver TcpApp

Wireless - 5 components

- setup
- node configuration
  - layer 3-2, layer 1, tracing, energy
- node coordinates
- node movements
- nam tracing

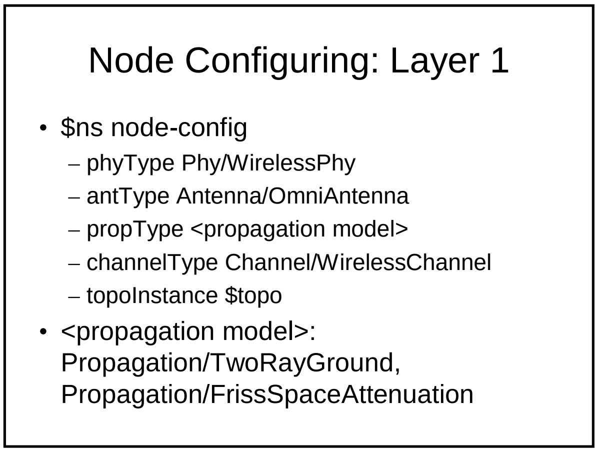
Setup

- set ns [new Simulator]
- set topo [new Topography]
- $topo load_flatgrid <length> <width>



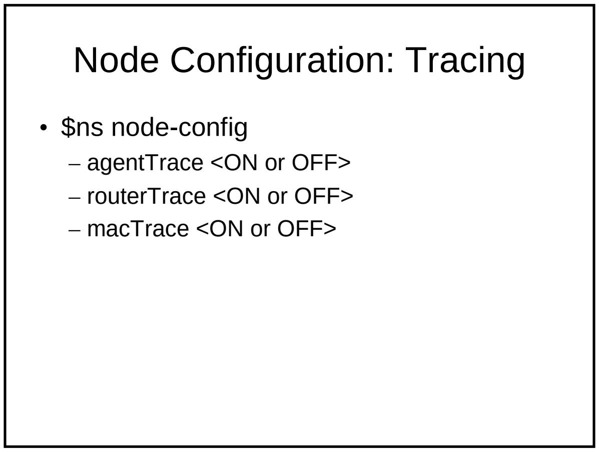
Node Configuration: Layer 3-2

- $ns node-config
  - adhocRouting <adhoc routing type>
  - llType LL
  - ifqType Queue/DropTail/PriQueue
  - ifqLen <queue length>
  - macType Mac/802_11
- <adhoc routing type>: DSDV, DSR, TORA, AODV



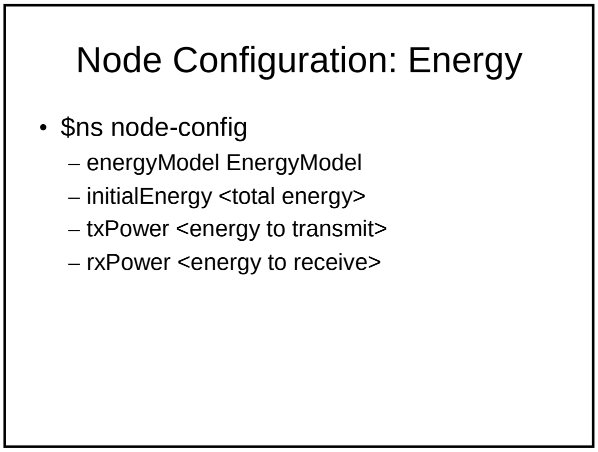
Node Configuring: Layer 1

- $ns node-config
  - phyType Phy/WirelessPhy
  - antType Antenna/OmniAntenna
  - propType <propagation model>
  - channelType Channel/WirelessChannel
  - topoInstance $topo
- <propagation model>: Propagation/TwoRayGround, Propagation/FrissSpaceAttenuation



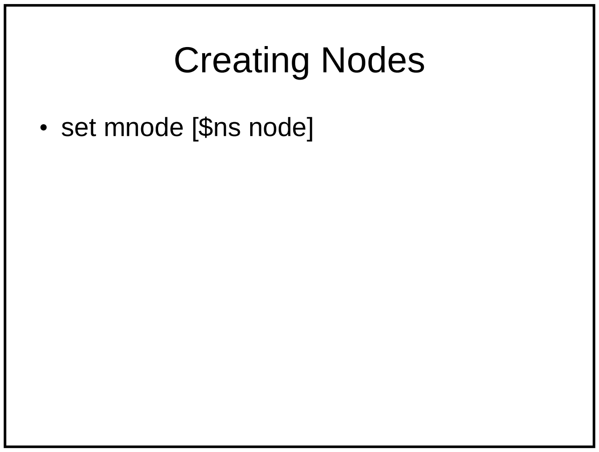
Node Configuration: Tracing

- $ns node-config
  - agentTrace <ON or OFF>
  - routerTrace <ON or OFF>
  - macTrace <ON or OFF>



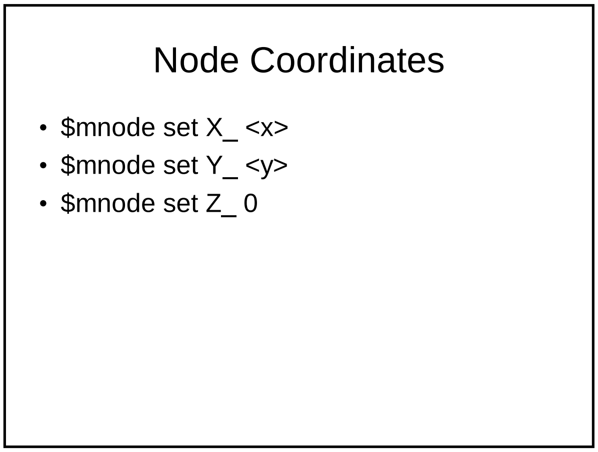
Node Configuration: Energy

- $ns node-config
  - energyModel EnergyModel
  - initialEnergy <total energy>
  - txPower <energy to transmit>
  - rxPower <energy to receive>



Creating Nodes

- set mnode [$ns node]



Node Coordinates

- $mnode set X_ <x>
- $mnode set Y_ <y>
- $mnode set Z_ 0

Node Movement

- Disable random motion
  - $mnode random-motion 0
- Specified
  - $ns at 1.0 “$mnode setdest <x> <y> <speed>”
- Random
  - $ns at 1.0 “$mnode start”

Tracing

- at the beginning
  - $ns namtrace-all-wireless [open test.nam w] <length> <width>
- initialize nodes
  - $ns initial_node_position $mnode 20

Case Studies

- multicast (mcast.tcl)
- wireless (wireless-udp.tcl, wireless-tcp.tcl)