The Network Simulator-ns-2

Claudia Canali

17/11/03

Tutorial Outline

- Part I
  - NS fundamentals
  - Wired world
  - Walk-thru an example NS script
- Part II
  - Visualization tools & other utilities
  - Help and references

What Is NS?

- Started as variant of REAL in 1989
- Developed at UC Berkely
- Discrete event, packet level simulator
- Written in C++ with Otcl frontend
- Wired, wireless and emulation modes
- Link layer and up for most wired
- Additional lower layers for wireless

Platforms

- Most UNIX and UNIX-like systems
  - Linux
  - FreeBSD
  - SunOS/Solaris
  - HP/SGI (with some tweaking)
- Windows 95/98/NT/ME/2000
  - Tips on build available
  - However validation tests don’t work

Architecture in NS

- Object-oriented (C++ and Otcl)
  - Algorithms over large data sets, per packet handling in C++
  - Configuration, “one-time” stuff in Otcl
  - Fast to run, quick to re-configure
  - Fine grained object composition

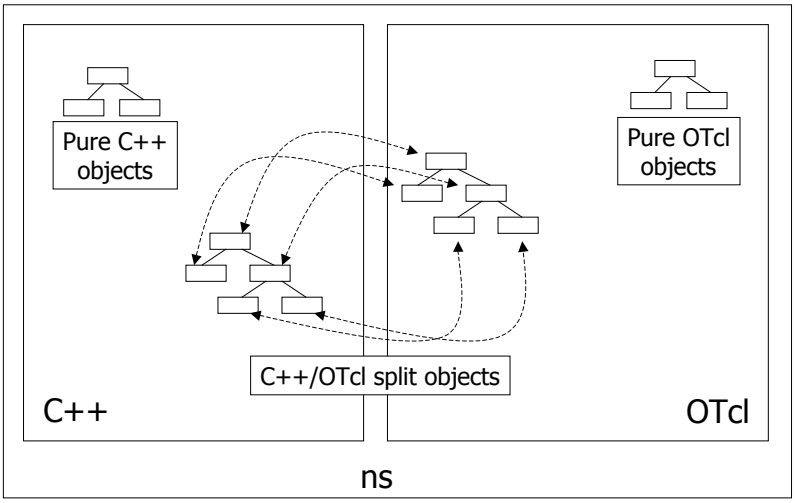
C++ and OTcl Separation

- C++ for “data”
  - Per packet action
- OTcl (Object Tcl Extensions) for control
  - Configuration, “one-time” task

\+ Compromise between composibility and speed

– Learning and debugging

OTcl and C++: The Duality



Basic Tcl (Tool Command Language)

```
set a 43
set b 27
proc test { a b } {
  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]"
    }
  }
}
```

Basic OTcl

```
Class Mom
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 mom [new Mom]
$mom set age_ 45
set kid [new Kid]
$kid set age_ 15

$mom greet
$kid greet
```

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 - Batch Mode

simple.tcl

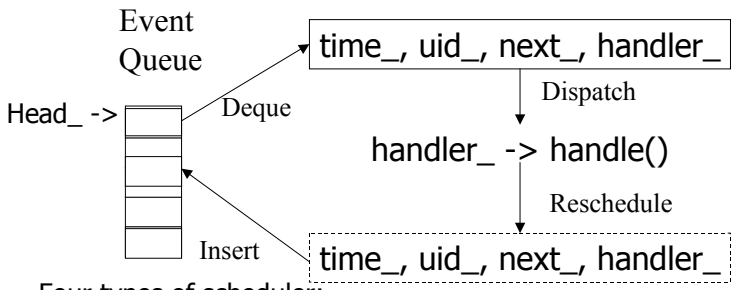
```
set ns [new Simulator]
$ns at 1 "puts \"Hello World!\""
$ns at 1.5 "exit"
$ns run
```

```
swallow 74% ns simple.tcl
Hello World!
swallow 75%
```

Event Driven Simulation

- Scheduler – main controller of events
- Scheduler clock - simulator virtual time
  [$ns_ now] returns the current simulator time
- Event queue - holds events in the order of their firing times
- Events have a firing time and a handler function
- Two types of events possible – packets and "at-events"

Discrete Event Scheduler



Four types of scheduler:

- List: simple linked list, order-preserving, O(N)
- Heap: O(logN)
- Calendar: hash-based, fastest, default, O(1)
- Real-time: subclass of list, sync with real-time, O(N)

Unit of time: seconds

Elements of Ns-2

- Create the event scheduler
- [Turn on tracing]
- Create network
- Setup routing
- Insert errors
- Create transport connection
- Create traffic

Creating Event Scheduler

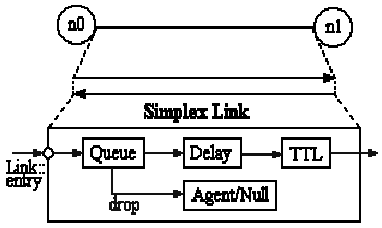
- Create event scheduler
  - set ns [new Simulator]
- Schedule events
  - $ns at <time> <event>
  - <event>: any legitimate ns/tcl commands
- Start scheduler
  - $ns run

Creating Network Topology

- Nodes
  - set n0 [$ns node]
  - set n1 [$ns node]
- Links and queuing
  - $ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>
  - <queue_type>: DropTail, RED, CBQ, FQ, SFQ, DRR
  - $ns duplex-link $n0 $n1 5Mb 2ms DropTail

Links and queuing

- Duplex-link → 2 simple links in both directions



- Delay object simulates the link delay
- DropTail implements FIFO scheduling

Tracing

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

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt Id
d	**1.7312**	**2**	**3**	**tcp**	**1040**	**-------**	**1**	**0.0**	**3.0**	**12**	**471**

- Trace packets on all links in nam format
  - $ns namtrace-all [open test.nam w]
  - $ns namtrace-all-wireless [open wtest.nam w]

Tracing

- Turn on tracing on specific links
  - $ns trace-queue $n0 $n1
  - $ns namtrace-queue $n0 $n1
- Trace-all commands must appear immediately after creating scheduler (links created AFTER will have trace objects inserted)

Tracing

- Event tracing
  - $ns eventtrace-all [$file]
  - Add eventtrace *after* trace-all as trace-all file is used as default
  - Example script: ~ns/tcl/ex/tcp-et.tcl

Inserting Errors

- Creating Error Model and set its packet error rate to 1 percent
  - set loss_module [new ErrorModel]
  - $loss_module set rate_ 0.01
  - $loss_module unit pkt #packets is default
  - $loss_module ranvar [new RandomVariable/Uniform] #uniform is default
  - $loss_module drop-target [new Agent/Null]
- Inserting Error Module
  - $ns lossmodel $loss_module $n0 $n1

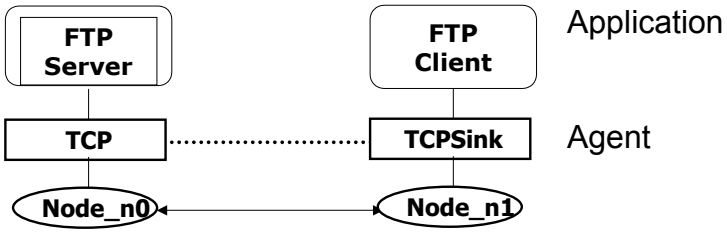
Setup Routing (node type and routing protocol)

- Unicast (default)
  - $ns rtproto <type>
  - <type>: Static, Session, DV, LS, Manual or hierarchical
- Multicast
  - $ns multicast (right after [new Simulator])
  - $ns mrtproto <type>
  - <type>: CtrMcast, DM, ST, BST

Agents & Connections

Agent:

- usually an endpoint/entity
- used in implementation of protocols at various layers



Creating Connection: TCP

- One-way TCP sending agent [Tahoe, Reno, NewReno, Sack, Vegas and Fack]
  - 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
- Telnet
  - set telnet [new Application/Telnet]
  - $telnet attach-agent $tcp

Creating Connection: UDP

#Setup a UDP connection

- set udp [new Agent/UDP]
- $ns attach-agent $n0 $udp
- set null [new Agent/Null]
- $ns attach-agent $n1 $null
- $ns connect $udp $null
- $udp set fid_ 2

Creating Traffic: On Top of UDP

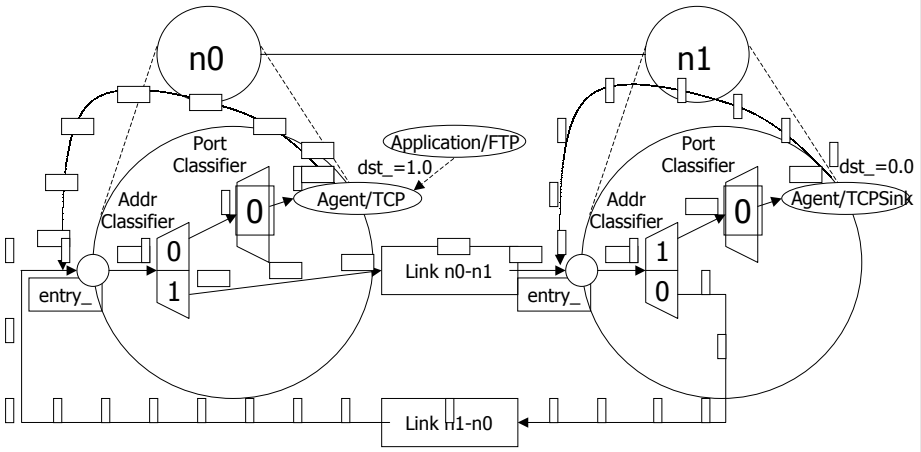
- CBR (Constant Bit Rate)
  - set src [new Application/Traffic/CBR]
- Exponential or Pareto on-off
  - set src [new Application/Traffic/Exponential]
  - set src [new Application/Traffic/Pareto]

#Setup a src over UDP connections

- $src attach-agent $udp
- $src set packet_size_ 1000 #set MSS

Plumbing: Packet Flow

n0 and n1 with network addresses 0 and 1 respectively

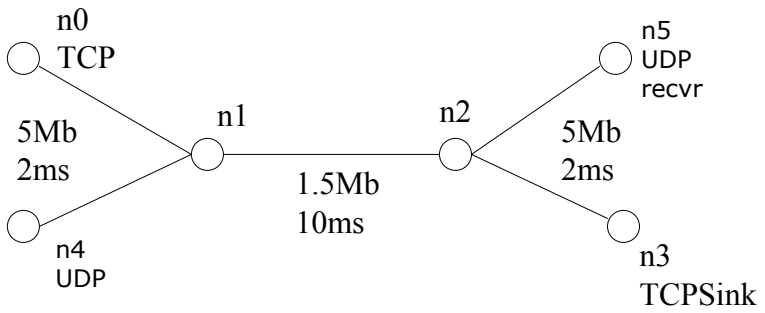


Summary: Generic Script Structure

```
set ns [new Simulator]
# [Turn on tracing]
# Create topology
# Setup packet loss, link dynamics
# Create routing agents
# Create:
#   - multicast groups
#   - protocol agents
#   - application and/or setup traffic sources
# Post-processing procs
# Start simulation
```

Example - TCP

- Simple scenario with TCP and UDP connections



TCP : Step 1

- Scheduler & tracing

```
#Create scheduler
set ns [new Simulator]

#Turn on tracing
set f [open out.tr w]
$ns trace-all $f
set nf [open out.nam w]
$ns namtrace-all $nf
```

→out.tr for simulation analysis
→ out.nam as input for NAM

TCP : Step 2

- Create topology

```
#create nodes
set n0 [$ns node]
set n1 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
```

TCP : Step 3

```
#create links
$ns duplex-link $n0 $n1 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n2 $n3 5Mb 2ms DropTail
$ns queue-limit $n1 $n2 25
$ns queue-limit $n2 $n1 25
```

TCP : Step 4

- Create TCP agents

```
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
```

TCP : Step 5

- Attach traffic

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
#start application traffic
$ns at 1.1 “$ftp start”
```

TCP : Step 6

- End of simulation wrapper (as usual)

```
$ns at 2.0 “finish”
proc finish {} {
   global ns f nf
   close $f
   close $nf
   puts “Running nam…”
   exec nam out.nam &
   exit 0
}
$ns run
```

Viz Tools

Nam-1 (Network Animator Version 1.9)

- Packet-level animation
- Well-supported by ns

Xgraph

- Convert trace output into xgraph format

Ns-nam Interface

- Color
- Node manipulation
- Link manipulation
- Topology layout
- Protocol state
- Misc

Nam Interface: Color

- Color mapping

```
$ns color 40 red
$ns color 41 blue
$ns color 42 chocolate
```

- Color ↔ flow id association

```
$tcp0 set fid_ 40 ;# red packets
$tcp1 set fid_ 41 ;# blue packets
```

Nam Interface: Nodes

- Color node

```
$node color red
```

- Shape (can't be changed after sim starts)

```
$node shape box      ;# circle, box, hexagon
```

- Marks (concentric "shapes")

```
$ns at 1.0 "$n0 add-mark m0 blue box"
$ns at 2.0 "$n0 delete-mark m0"
```

- Label (single string)

```
$ns at 1.1 "$n0 label \"web cache 0\""
```

Nam Interfaces: Links

- Color

```
$ns duplex-link-op $n0 $n1 color "green"
```

- Label

```
$ns duplex-link-op $n0 $n1 label "abced"
```

- Dynamics (automatically handled)

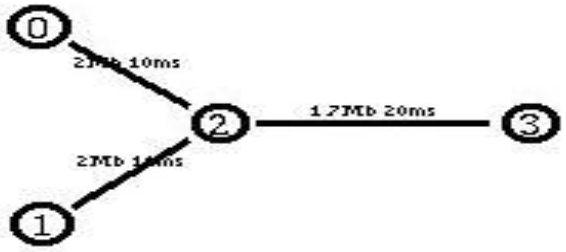
```
$ns rtmodel Deterministic {2.0 0.9 0.1} $n0 $n1
```

- Asymmetric links not allowed

Nam Interface: Topo Layout

- "Manual" layout: specify everything

```
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```



Nam Interface: Protocol State

- Monitor values of agent variables

```
$ns add-agent-trace $srm0 srm_agent0
$ns monitor-agent-trace $srm0
$srm0 tracevar C1_
$srm0 tracevar C2_
# … …
$ns delete-agent-trace $tcp1
```

Nam Interface: Misc

- Annotation
  - Add textual explanation to your sim

```
$ns at 3.5 "$ns trace-annotate \"packet drop\""
```

- Set animation rate

```
$ns at 0.0 "$ns set-animation-rate 0.1ms"
```

Other Utilities in Ns

- Nam editor
  Available as part of nam-1
- Tcl debugger
  For source and documentation, see
  http://www.isi.edu/nsnam/ns/ns-debugging.html
- Topology generator
  http://www.isi.edu/nsnam/ns/ns-topogen.html
- Scenario generator
  http://www.isi.edu/nsnam/ns/ns-scengeneration.html

Other Ns Features

- Other areas in wired domain
  - LANs
  - Diffserv
  - Multicast
  - Full TCP
  - Applications like web-caching
- Wireless domain
  - Ad hoc routing
  - Mobile IP
  - Satellite networking
  - Directed diffusion (sensor networks)

Other Ns Features

- Emulator
  - Connect simulator in a real network
  - Can receive and send out live packets from/into the real world

Resources

- Ns distribution download
  - http://www.isi.edu/nsnam/ns/ns-build.html
- Installation problems and bug-fix
  - http://www.isi.edu/nsnam/ns/ns-problems.html
- Ns-users mailing list
  - Ns-users@isi.edu
  - See http://www.isi.edu/nsnam/ns/ns-lists.html
  - Archives from above URL

Resources (contd..)

- Marc Greis' tutorial
  - http://www.isi.edu/nsnam/ns/tutorial
- Ns-users archive
- Ns-manual
  - http://www.isi.edu/nsnam/ns/ns-documentation.html
- Tcl (Tool Command Language)
  - http://dev.scriptics.com/scripting
  - Practical programming in Tcl and Tk, Brent Welch
- Otcl (MIT Object Tcl)
  - ~otcl/doc/tutorial.html (in distribution)