Ns2 Tutorial

Ns2 Tutorial

08.02.2006

Tutorial Outline

- Part I
  - NS fundamentals
  - Wired world
  - Walk-thru an example NS script
- Part II
  - Visualization tools & other utilities
  - Help and references
- Outcome of this tutorial
  - You know the concept of ns2
  - You are able to set-up a network topology on your own (with nam-editor)
  - You are able to add traffic like ftp/telnet or cbr to your simulation
  - You are able to start the ns and the nam.
  - You understand the ns-script code generated by nam-editor

What Is NS?

- Started as REAL in 1989
- Discrete event, packet level simulator
- Written in C++ with Otcl front-end
- 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/XP
  - Best based on cygwin
  - Tips on build available
  - However validation tests don’t work

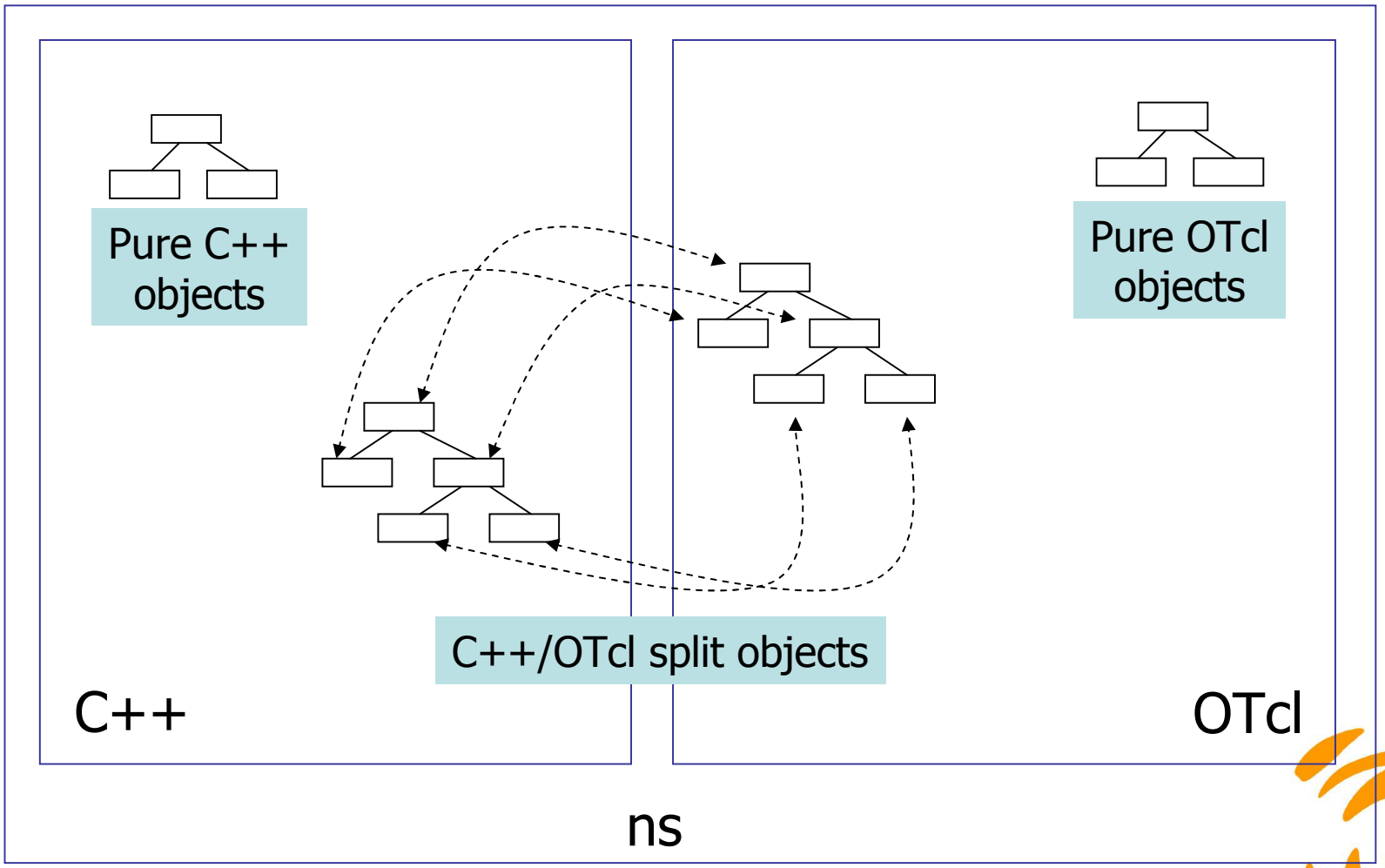
ns Architecture

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

\+ Compromise between composibility and speed

– Learning and debugging

OTcl and C++: The Duality



Basic tcl

```
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]"
     }
   }
}
```

To invoke tclshell, type “tclsh”; To exit the shell type “exit”

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
```

To invoke otclshell, type "otclsh"; To exit the shell type "exit"; to run this example type "ns" (otherwise new will not work!!!)

Hello World - Interactive Mode

```
sunray01>ns
% set ns [new Simulator]
_o3
% $ns at 1 “puts \“Hello World!\””
1
% $ns at 1.5 “exit”
2
% $ns run
Hello World!
sunray01>
```

Hello World - Batch Mode

simple.tcl

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

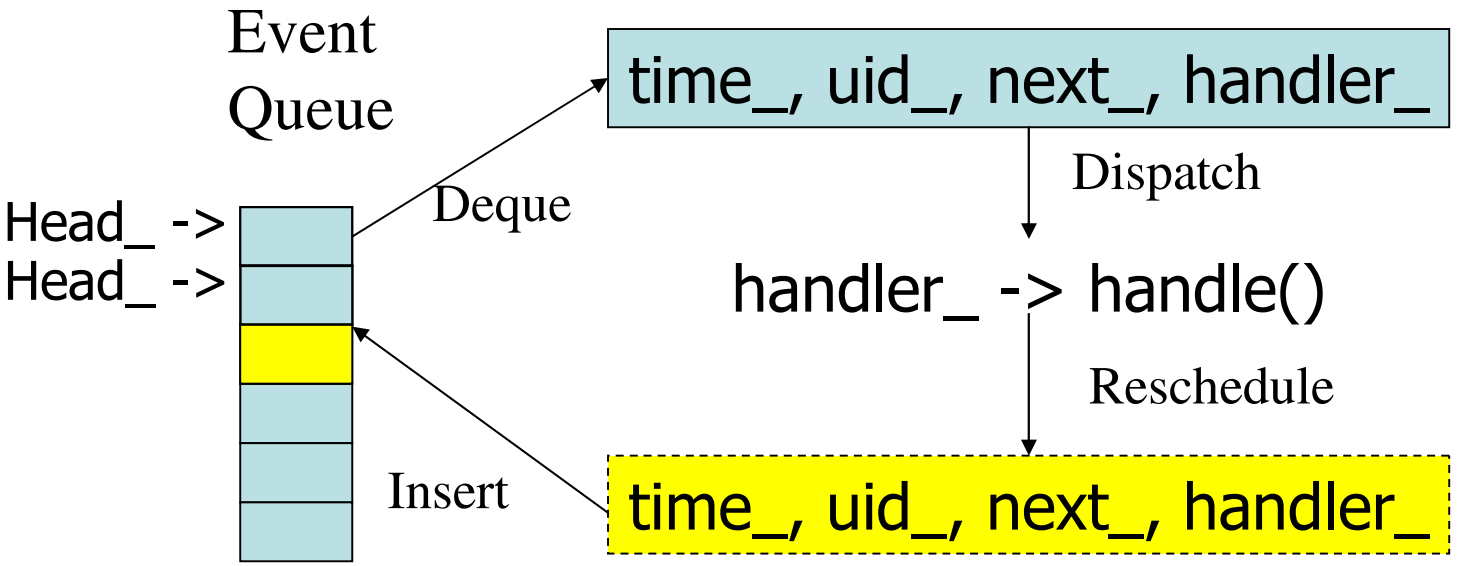
```
sunray01> ns simple.tcl
Hello World!
sunray01>
```

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




```
Example:
$ns at 0.5 "$ftp start"
$ns at 4.5 "$ftp stop"
$ns run
```

Elements of Ns-2

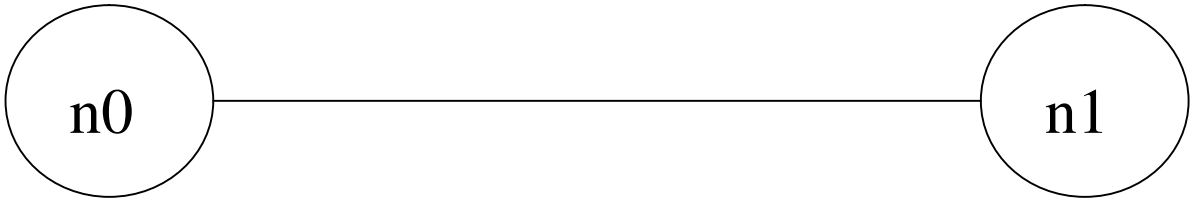
- Create the event scheduler
- [Turn on tracing]
- Create network
- Setup routing
- [Insert errors]
- Create transport connection
- Create traffic

Are you Confused?

- Before getting into even more Details let‘s practice a bit
- Login with your account name
- Open a shell; create subdirectory „ns-test“ and move there
- Type "nam &"

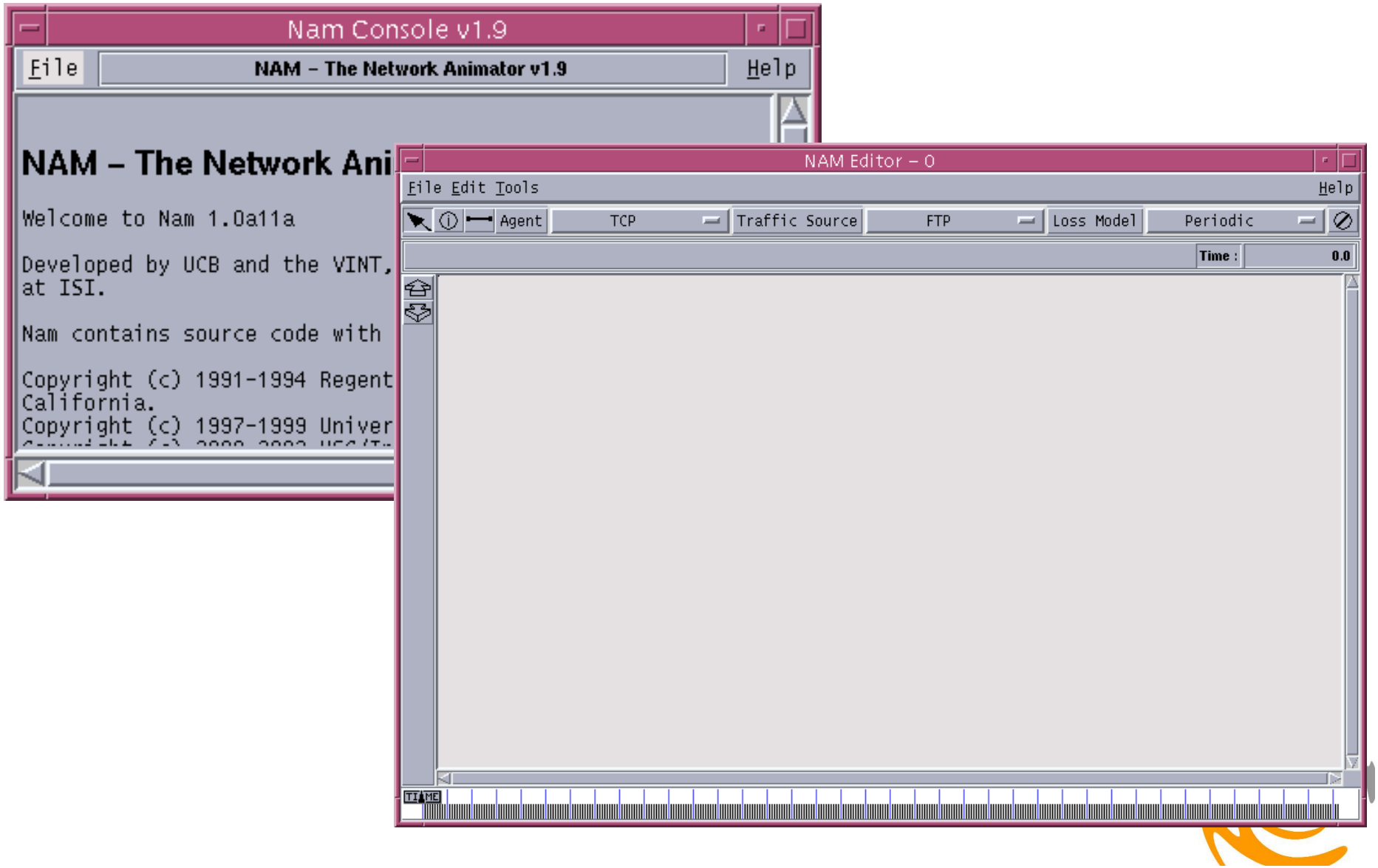
Example

- Create the following topology

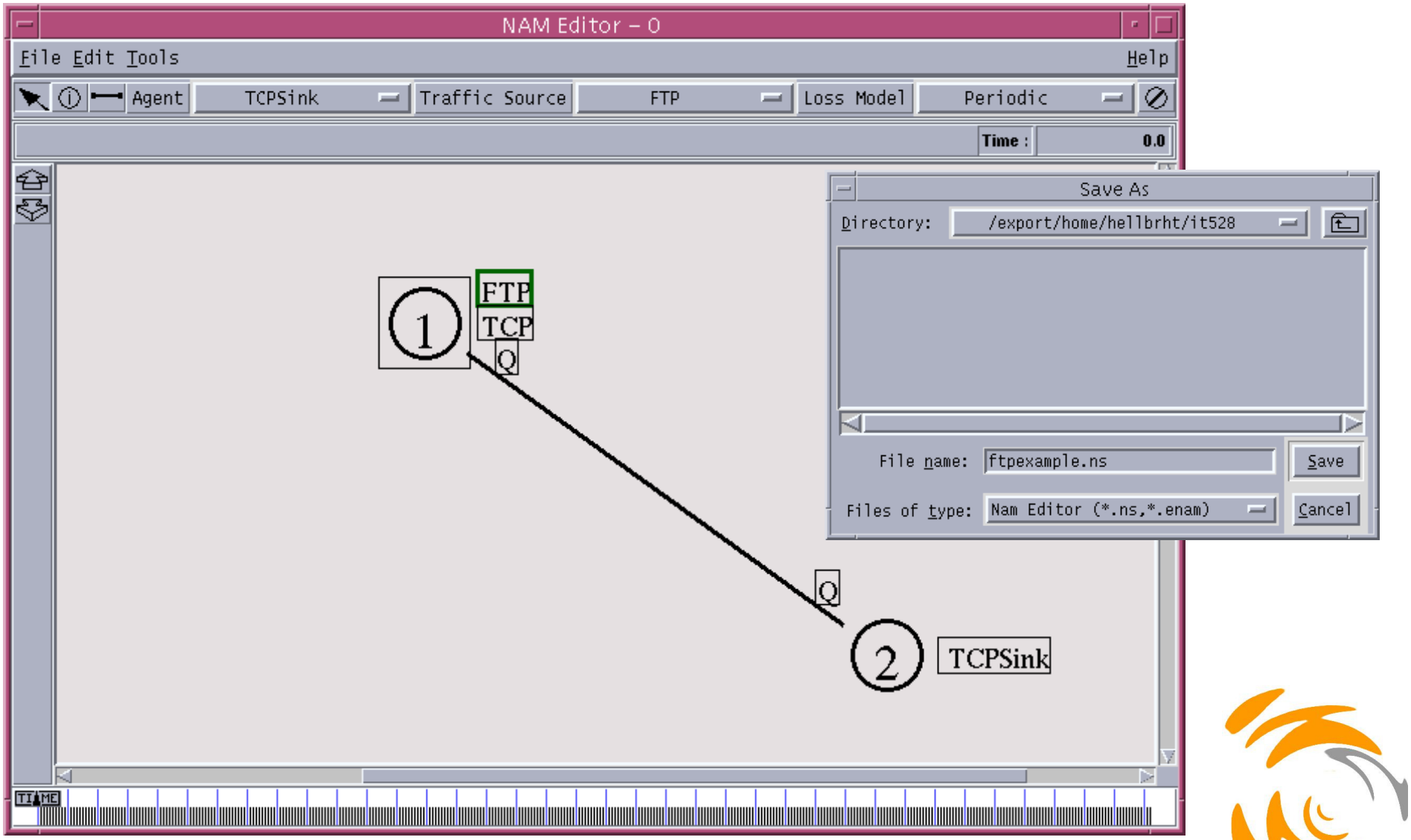


- Attach a TCP Agent and a TCPSink and connect an ftp application on top of TCP
- Save the simulation as „ftpexample.ns“
- Run the simulation
- Start the vizualization tool

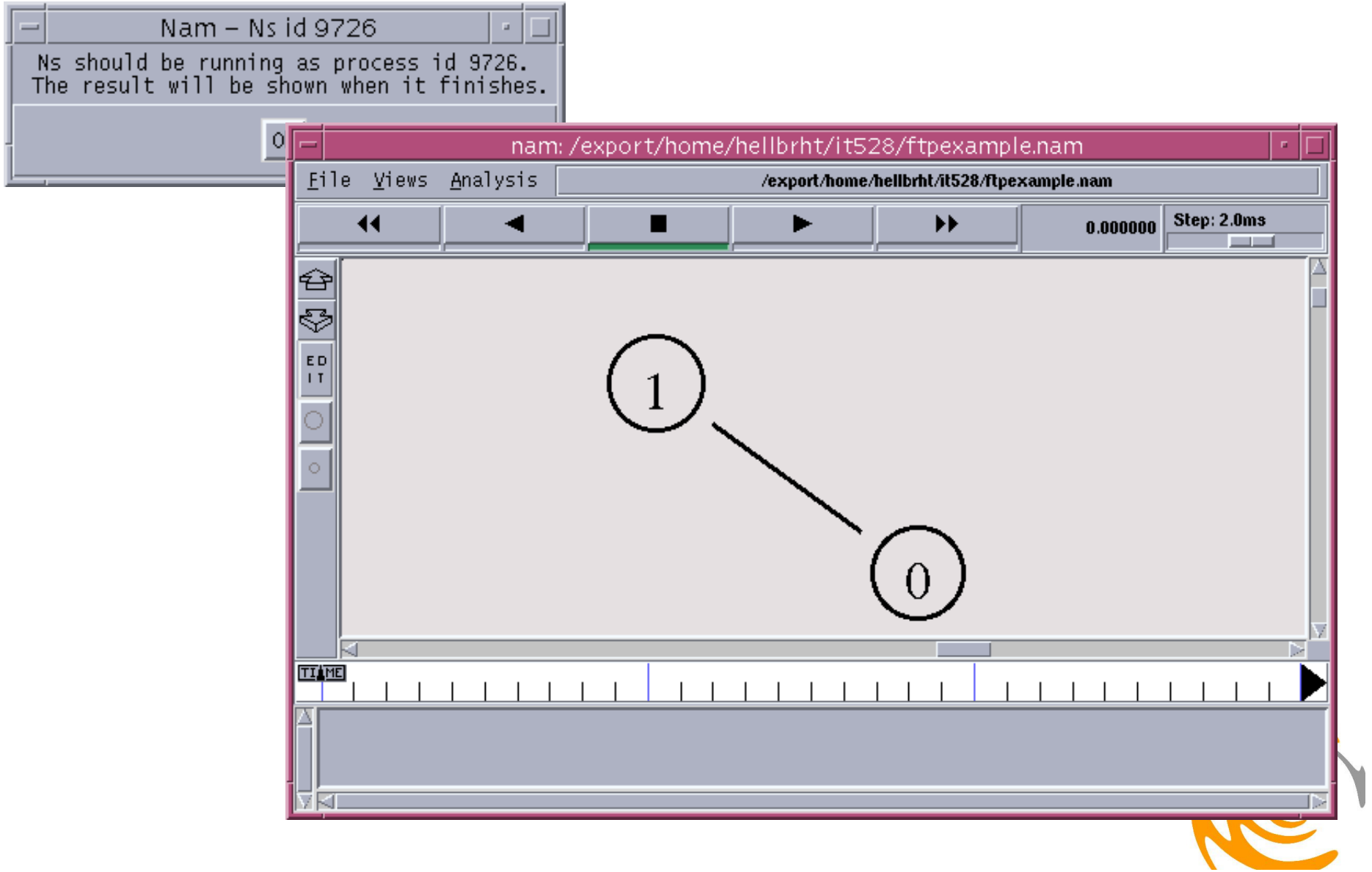
Example (cont.)



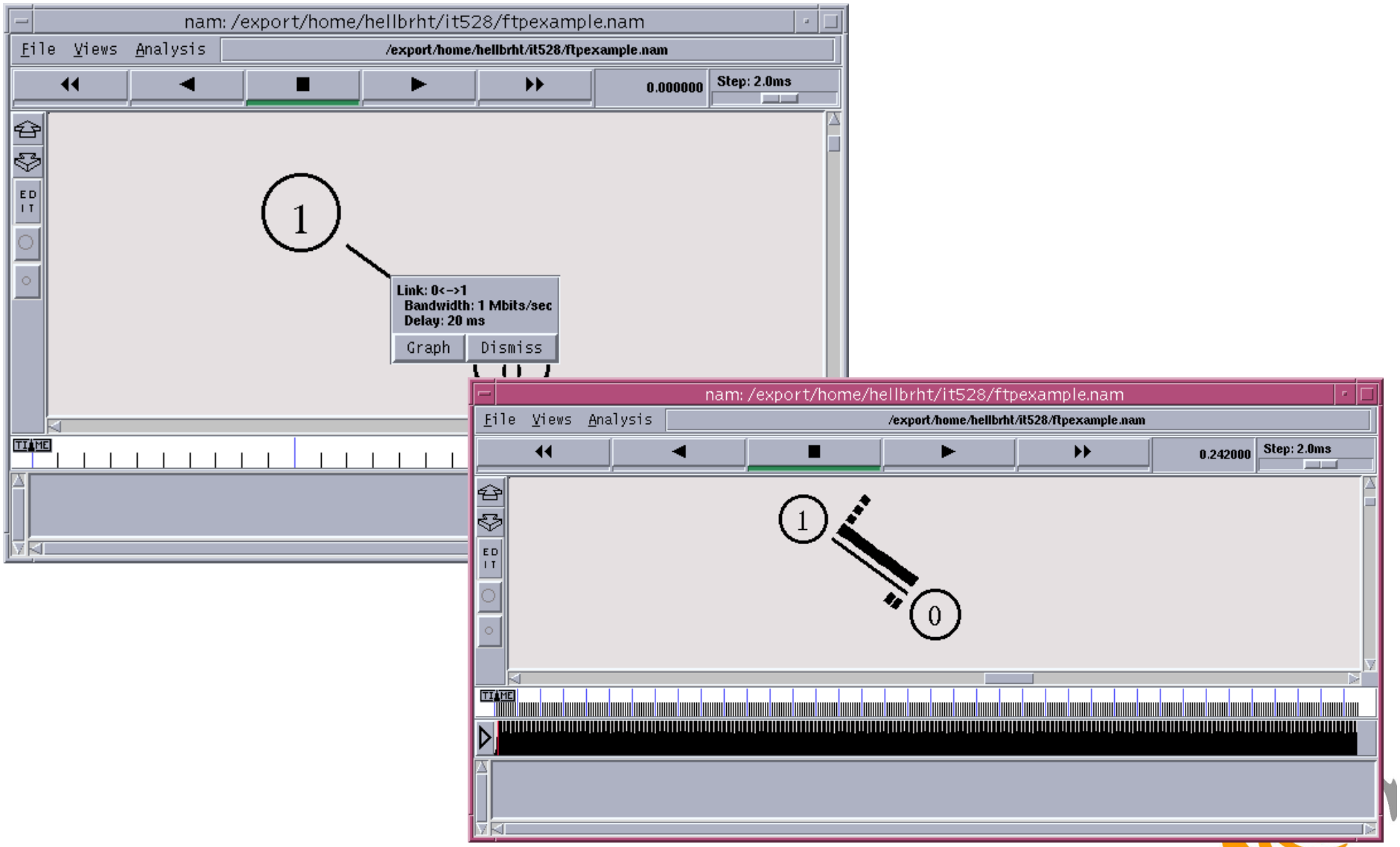
Example (cont.)



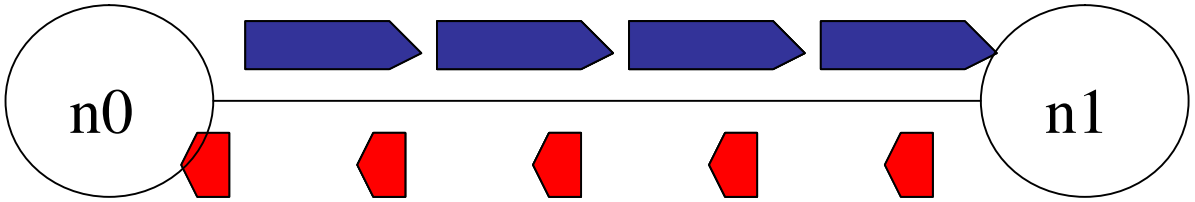
Example (cont.)



Example (cont.)



Example Output and ns script




```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1.5Mb
    10ms DropTail

set tcp [$ns create-connection TCP
    $n0 TCPSink $n1 0]

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.2 "$ftp start"
$ns at 1.2 ”exit"
$ns run
```

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

Tracing

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

```
<event> <time> <from> <to> <pkt> <size> -- <fid> <src>
   <dst> <seq> <attr>
    + 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 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

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

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

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
  - Hooks in routing module to reflect routing changes
- Four models

```
$ns rtmodel Trace <config_file> $n0 $n1
$ns rtmodel Exponential {<params>} $n0 $n1
$ns rtmodel Deterministic {<params>} $n0 $n1
$ns rtmodel-at <time> up|down $n0 $n1
```

- Parameter list

```
[<start>] <up_interval> <down_interval> [<finish>]
```

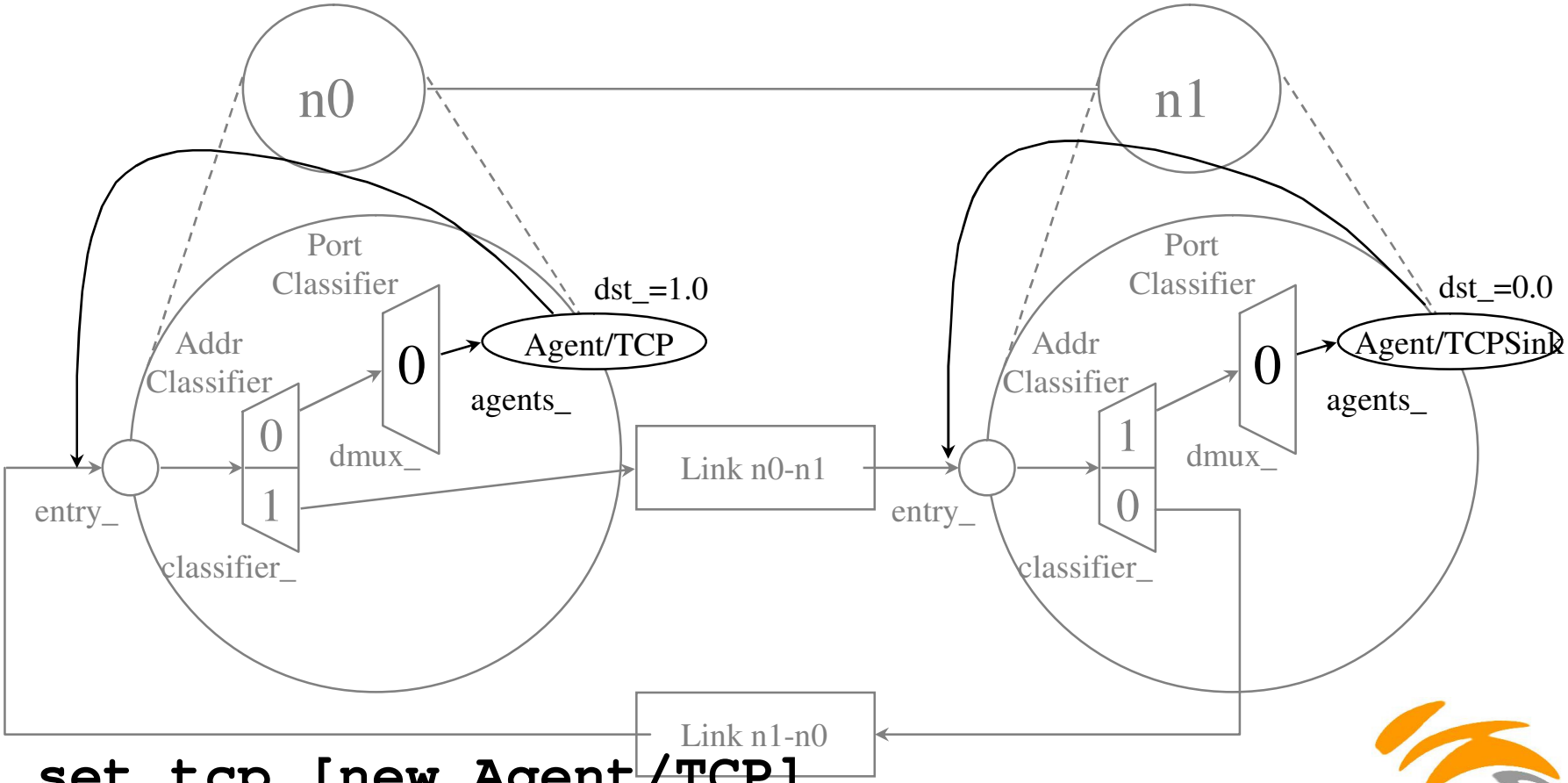
Setup Routing

- Unicast
  - $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

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

Transport

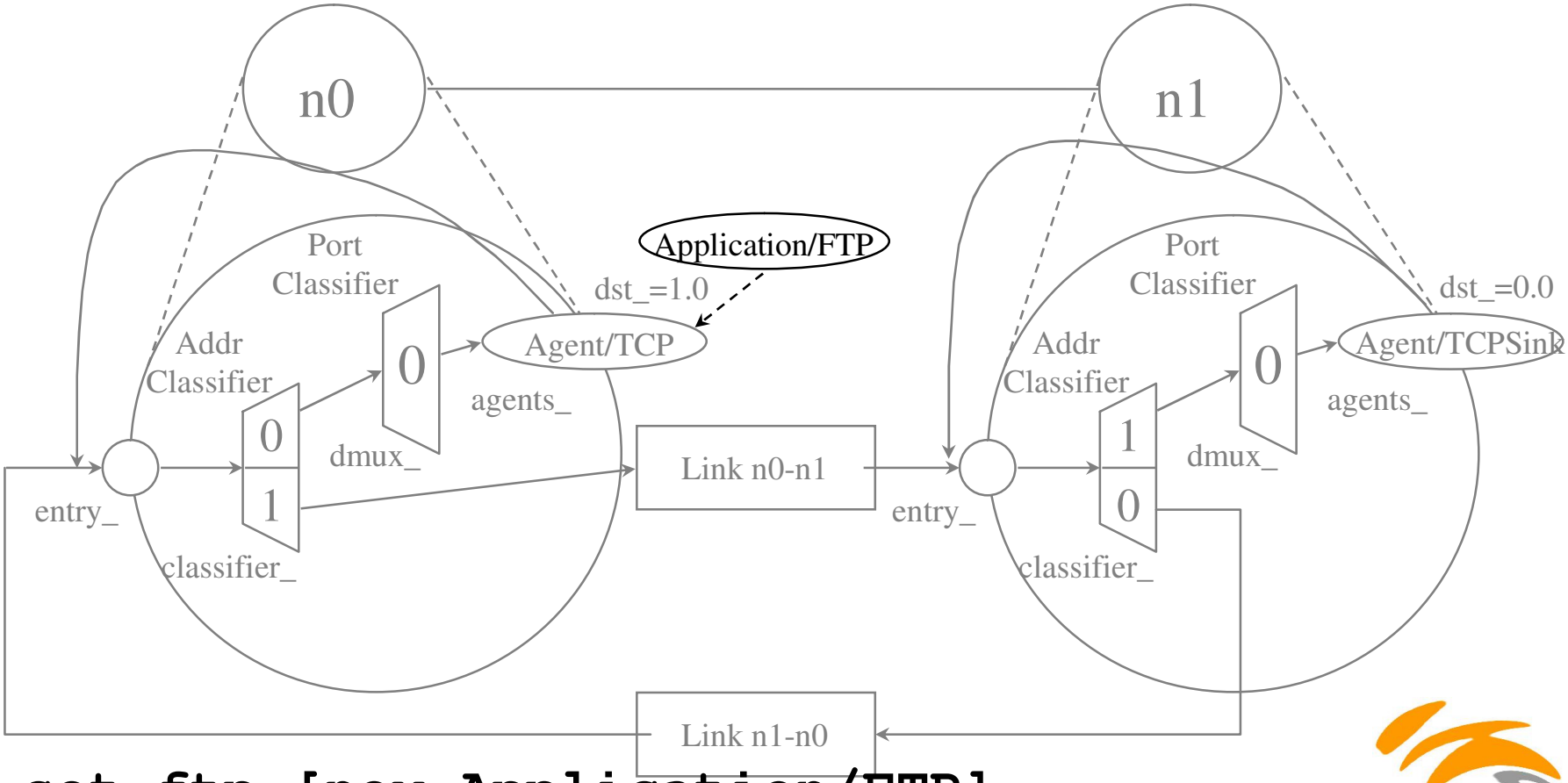



```
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
```

Creating Traffic: On Top of TCP

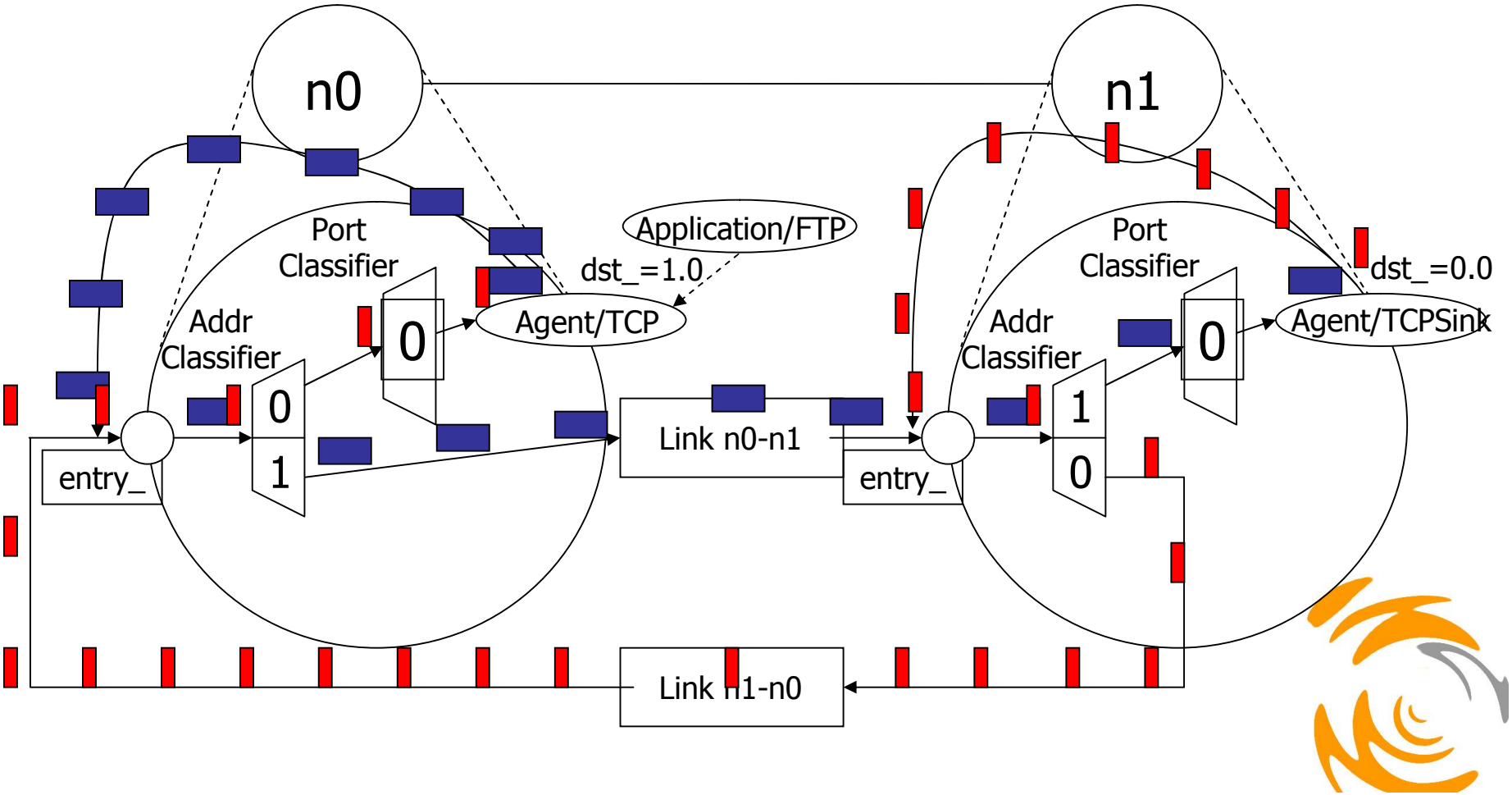
- FTP
  - set ftp [new Application/FTP]
  - $ftp attach-agent $tcp
- Telnet
  - set telnet [new Application/Telnet]
  - $telnet attach-agent $tcp

Application

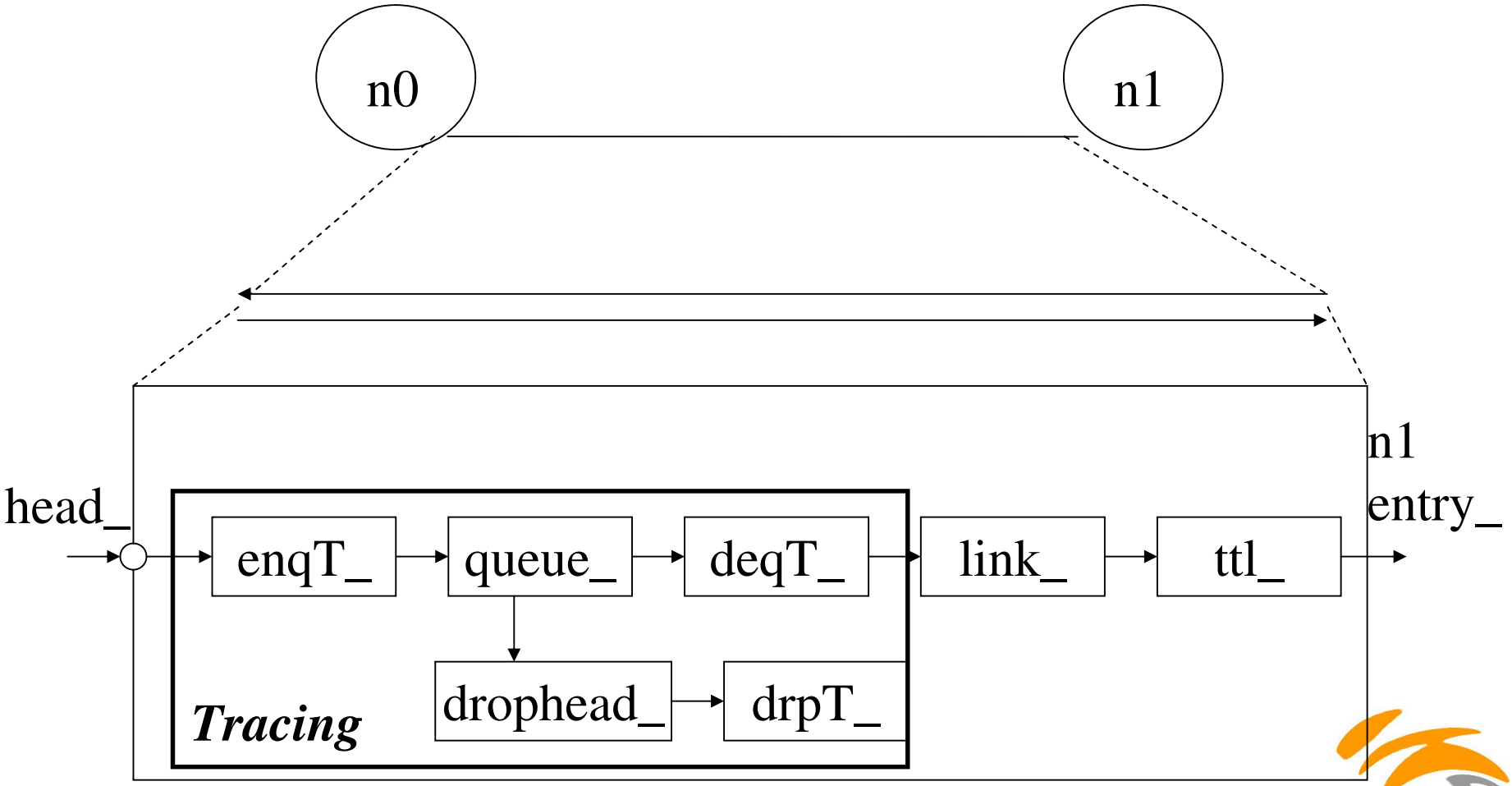



```
set ftp [new Application/FTP]
$tcp attach-agent $ftp
```

Plumbing: Packet Flow



Network Topology - Link



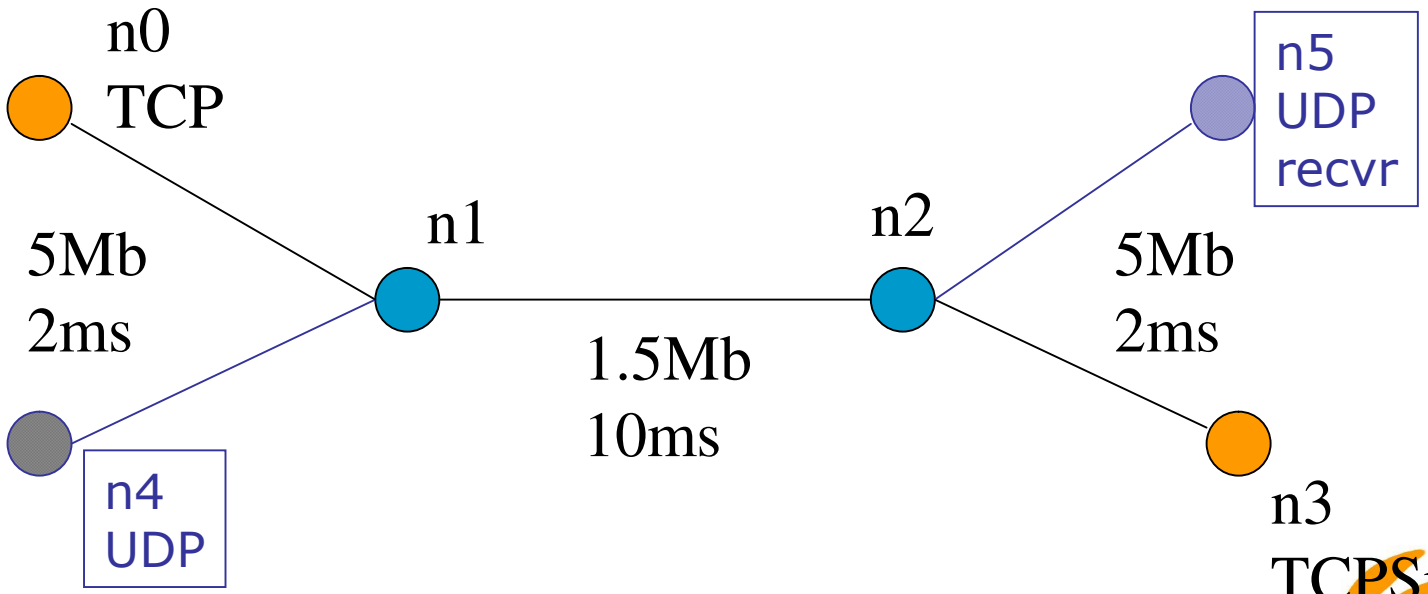

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

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
```

TCP : Step 2

- Create topology

```
#create nodes
set n0 [$ns node]
set n1 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$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
   … (for UDP)
```

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
```

… (for UDP)

TCP : Step 5

- Attach traffic

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

… (for UDP)

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 (Network AniMator)

- Packet-level animation
- Well-supported by ns
- Commandline start: “nam <nam-tracefile>”

Xgraph

- Convert trace output into xgraph format
- Commandline start: “xgraph”

Gnuplot

- Convert ASCII Text e.g. trace into 2D/3D plot
- Commandline start: “gnuplot <scriptfile>”

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 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 $n(0) $n(1) orient right
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient down
$ns duplex-link-op $n(3) $n(4) orient 60deg
```

- If anything missing → automatic layout

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"
```

Ns Features

- **Areas in wired domain**
  - LANs
  - diffserv/intserv
  - Multicast
  - Full TCP
  - Applications like web-caching
- **Wireless domain**
  - Ad hoc routing
  - Mobile IP
  - Satellite networking
  - Directed diffusion (sensor networks)
- Emulator
  - Connect simulator in a real network
  - Can receive and send out live packets from/into the real world

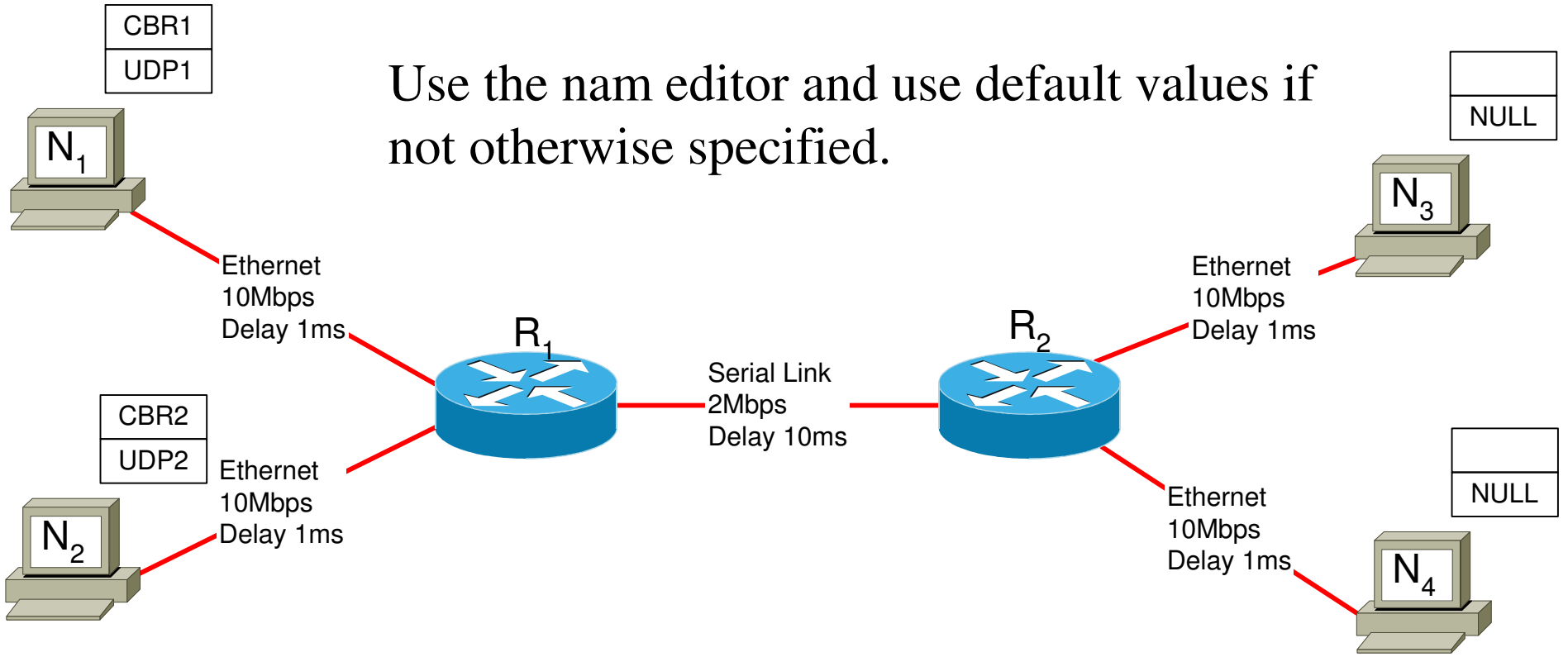
Resources

- 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)

Let's practice a bit II



Use the nam editor and use default values if not otherwise specified.

CBR1: starts at 0s mark it blue
CBR2: start at 0.1s mark it red

Is the distribution fair?

The End