


Copyright © William C. Cheng




ns-2, the network simulator

- ↳ a discrete event simulator
- ↳ simple model
- ↳ focused on modeling network protocols
- ↳ wired, wireless, satellite
- ↳ TCP, UDP, multicast, unicast
- ↳ web, telnet, ftp
- ↳ ad hoc routing, sensor networks
- ↳ infrastructure: stats, tracing, error models, etc.

Computer Communications - CSC1 551

Copyright © William C. Cheng




ns history

- ↳ Began as REAL in 1989
- ↳ ns by Floyd and McCanne at LBL
- ↳ ns-2 by McCanne and the VINT project (LBL, PARC, UCB, USC/ISI)
- ↳ currently maintained at USC/ISI, with input from Floyd et al.

Computer Communications - CSC1 551

Copyright © William C. Cheng




ns models

- ↳ Traffic models and applications:
 - ↳ web, FTP, telnet, constant-bit rate, Real Audio
- ↳ Transport protocols:
 - ↳ unicast: TCP (Reno, Vegas, etc.), UDP
 - ↳ multicast: SRM
- ↳ Routing and queuing:
 - ↳ wired routing, ad hoc rig and directed diffusion
 - ↳ queuing protocols: RED, drop-tail, etc.
- ↳ Physical media:
 - ↳ wired (point-to-point, LANs), wireless (multiple propagation models), satellite

Computer Communications - CSC1 551

Copyright © William C. Cheng



CS551

NS Tutorial


Slides Developed by:

John Heidemann
johnh@isi.edu

USC/ISI

Computer Communications - CSC1 551

Copyright © William C. Cheng




ns goals

- ↳ support networking research and education
- ↳ protocol design, traffic studies, etc.
- ↳ protocol comparison
- ↳ provide a collaborative environment
- ↳ freely distributed, open source
- ↳ share code, protocols, models, etc.
- ↳ allow easy comparison of similar protocols
- ↳ increase confidence in results
- ↳ more people look at models in more situations
- ↳ experts develop models
- ↳ multiple levels of detail in one simulator

Computer Communications - CSC1 551

Copyright © William C. Cheng



"ns" components

- ↳ ns, the simulator itself
- ↳ nam, the Network Animator
- ↳ visualize ns (or other) output
- ↳ GUI input simple ns scenarios
- ↳ pre-processing:
 - ↳ traffic and topology generators
- ↳ post-processing:
 - ↳ simple trace analysis, often in Awk, Perl, or Tcl

Computer Communications - CSC1 551

Copyright © William C. Cheng

7

ns status

- platforms: basically all Unix and Windows
- size: about 200k loc each C++ and Tcl, 350 page manual
- user-base: >1k institutions, >10k users
- releases about every 6 months, plus daily snapshots

ns status

- platforms: basically all Unix and Windows
- size: about 200k loc each C++ and Tcl, 350 page manual
- user-base: >1k institutions, >10k users
- releases about every 6 months, plus daily snapshots

Copyright © William C. Cheng

8

Outlines

- Concepts
- Essentials
- Getting Started
- Fundamental tcl, otcI and ns

Copyright © William C. Cheng

12

ns Software Structure: object orientation

- Object oriented: lots of code reuse (ex. TCP + TCP variants)
- Some important objects:
 - nsObject: has recv() method
 - Connector: has target() and drop()
 - BiConnector: uptarget() & downtarget()

Copyright © William C. Cheng

9

Discrete Event Simulation

- model world as events
- simulator has list of events
- process: take next one, run it, until done
- each event happens in an instant of virtual (simulated) time, but takes an arbitrary amount of real time
- ns uses simple model: single thread of control => no locking or race conditions to worry about (very easy)

Copyright © William C. Cheng

11

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)

Copyright © William C. Cheng

10

Discrete Event Examples

Consider two nodes on an Ethernet:

time	simple queuing model:
t=1:	A enqueues pkt on LAN
t=1.01:	LAN dequeues pkt and triggers B

time	detailed CSMA/CD model:
t=1.0:	A sends pkt to NIC
t=1.005:	A's NIC starts carrier sense, starts tx
t=1.006:	B's NIC begins reciving pkt
t=1.01:	B's NIC concludes pkt
t=1.01:	B's NIC passes pkt to app

Copyright © William C. Cheng

14

OTcl and C++: The Duality

OTcl (object variant of Tcl) and C++ share class hierarchy

OTcl is glue library that makes it easy to share functions, variables, etc.

Computer Communications - CSC1 551

Copyright © William C. Cheng

16

Installation and Documentation

- ↳ download ns-allinone (if you have your own machine, do *not* build this on USC servers)
- ↳ includes Tcl, OTcl, Tccl, ns, nam, etc.
- ↳ run ns and nam on ISD machines:
 - ~csc1551/ns
 - ~csc1551/nam
- ↳ mailing list: ns-users@isli.edu
- ↳ documentation (see url above)
- ↳ Marc Gries tutorial
- ↳ ns manual

Computer Communications - CSC1 551

Copyright © William C. Cheng

18

Hello World, Deconstructed

```
set ns [new Simulator]
create a simulator, put in var ns
$ns at 1 "puts \"Hello World!\""
schedule an event at time t=1 to print HW
$ns at 1.5 "exit"
and exit at a later time
$ns run
run time simulator
```

Computer Communications - CSC1 551

Copyright © William C. Cheng

13

ns Software Structure: C++ and OTcl

- ↳ Uses two languages
- ↳ C++ for packet-processing
- ↳ fast to run, detailed, complete control
- ↳ OTcl for control
- ↳ simulation setup, configuration, occasional actions
- ↳ fast to write and change
- ↳ pros: trade-off running vs. writing speed, powerful/documented config language
- ↳ cons: two languages to learn and debug in

Computer Communications - CSC1 551

Copyright © William C. Cheng

15

Outlines

- ↳ Concepts
- ↳ Essentials
- ↳ Getting Started
- ↳ Fundamental Tcl, otcl and ns

Computer Communications - CSC1 551

Copyright © William C. Cheng

17

Hello World

```
simple.tcl:
set ns [new Simulator]
$ns at 1 "puts \"Hello World!\""
$ns run
$ns at 1.5 "exit"
$ns at 1 "puts \"Hello World!\""
$ns run
nunk1 74% ~csc1551/ns simple.tcl
Hello World!
nunk1 75%
Think C++:
Simulator *ns=new Simulator;
ns->at(1, "puts \"Hello World!\"");
ns->at(1.5, "exit");
ns->run();
```

Computer Communications - CSC1 551

Copyright © William C. Cheng

Basic Tcl

Also lists, associative arrays, etc.
 ⇒ can use a real programming language to build network topologies, traffic models, etc.

```

variables:
set x 10
puts "x is $x"

functions and expressions:
set y [pow x 2]
set y [expr x*x]

control flow:
if {$x > 0} { return $x }
else { return [expr -$x] }
while { $x > 0 } {
  puts $x
  incr x -1
}

procedures:
proc pow {x n} {
  if {$n == 1} { return $x }
  set part [pow x [expr $n-1]]
  return [expr $x*$part]
}

```

20

Copyright © William C. Cheng

Basic ns-2

- Creating the event scheduler
- [Turn on tracing]
- Creating network
- Setting up routes
- Inserting errors
- Creating transport connection
- Create traffic

22

Copyright © William C. Cheng

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

24

Copyright © William C. Cheng

Outlines

- Concepts
- Essentials
- Getting Started
- *Fundamental tcl, otc and ns*

19

Copyright © William C. Cheng

Basic otc

⇒ can easily make variations of existing things (TCP, TCP/Reno)

```

class Person {
  # constructor:
  Person instproc init {age} {
    set age_ $age
  }
  # method:
  Person instproc greet {} {
    puts "Sage_years old: How are you doing?"
  }
  # subclasses:
  class kid -superclass Person {
    kid instproc greet {} {
      $self instvar age_
      puts "Sage_years old kid: what's up, dude?"
    }
  }
  set a [new Person 45]
  set b [new Kid 15]
  $a greet
  $b greet
}

```

21

Copyright © William C. Cheng

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

23

Copyright © William C. Cheng

25

Computing routes

- Unicast
 - \$ns rtpproto <type>
 - <type>: Static, Session, DV, cost, multi-path
- Multicast
 - \$ns multicast
 - right after [new Simulator]
 - \$ns mrtproto <type>
 - <type>: ChRcast, DM, ST, BST

simple two layers: transport and app

- transports: TCP, UDP, etc.
- applications: (agents) ftp, telnet, etc.

Traffic

Computer Communications - CSC1 551

Copyright © William C. Cheng

28

Creating Connection: UDP

- source and sink
 - set src [new Agent/UDP]
 - set dst [new Agent/TCP]
- connect to nodes and each other
 - \$ns attach-agent \$n0 \$src
 - \$ns attach-agent \$n1 \$dst
 - \$ns connect \$src \$dst

Creating Connection: TCP

- source and sink
 - set src [new Agent/TCP]
 - set dst [new Agent/TCPSink]
- connect to nodes and each other
 - \$ns attach-agent \$n0 \$src
 - \$ns attach-agent \$n1 \$dst
 - \$ns connect \$src \$dst

Computer Communications - CSC1 551

Copyright © William C. Cheng

30

Creating Traffic: On Top of UDP

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

Computer Communications - CSC1 551

Copyright © William C. Cheng

27

Creating Connection: UDP

- source and sink
 - set src [new Agent/UDP]
 - set dst [new Agent/NULL]
- connect them to nodes, then each other
 - \$ns attach-agent \$n0 \$src
 - \$ns attach-agent \$n1 \$dst
 - \$ns connect \$src \$dst

Creating Connection: TCP

- source and sink
 - set src [new Agent/TCP]
 - set dst [new Agent/TCPSink]
- connect to nodes and each other
 - \$ns attach-agent \$n0 \$src
 - \$ns attach-agent \$n1 \$dst
 - \$ns connect \$src \$dst

Computer Communications - CSC1 551

Copyright © William C. Cheng

29

Creating Traffic: On Top of TCP

- FTP
 - set ftp [new Application/FTP]
 - \$ftp attach-agent \$src
 - \$ns at <time> "\$ftp start"
- Telnet
 - set telnet [new Application/Telnet]
 - \$telnet attach-agent \$src

Computer Communications - CSC1 551

Copyright © William C. Cheng

32

Compare to Real World

- more abstract (much simpler):
- no addresses, just global variables
- connect them rather than name lookup/bind/listen/accept
- easy to change implementation

```

set tsrc2 [new Agent/TCP/Newreno]
set tsrc3 [new Agent/TCP/Vegas]

```

Computer Communications - CSC1 551

Copyright © William C. Cheng

34

Tracing

- Trace packets on all links into test.out
- Trace packets on all links in nam-1 format

```

$ns namtrace-all [open test.nam w]

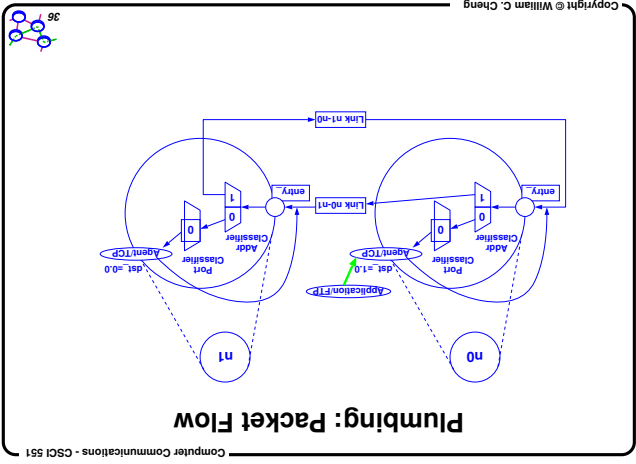
```

```

<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

```

Computer Communications - CSC1 551



Copyright © William C. Cheng

31

Creating Traffic: Trace Driven

- Trace driven
- Binary format
- inter-packet time (msec) and packet size (byte)

```

set file [new Tracefile]
set file filename <file>
set src [new Application/Traffic/Trace]
$src attach-tracefile $file
<file>:

```

Computer Communications - CSC1 551

Copyright © William C. Cheng

33

Inserting Errors

- Creating Error Module
- Inserting 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]

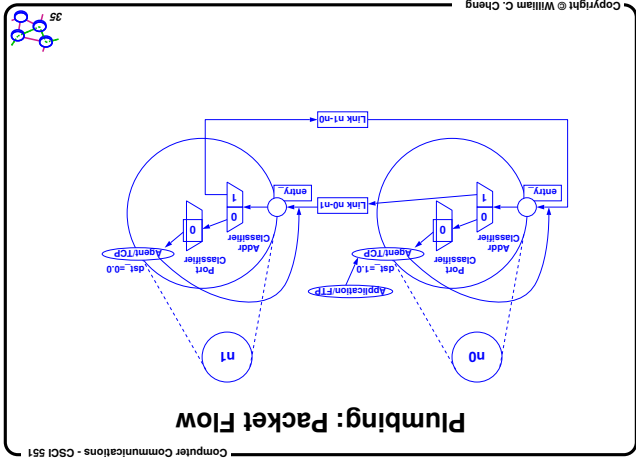
```

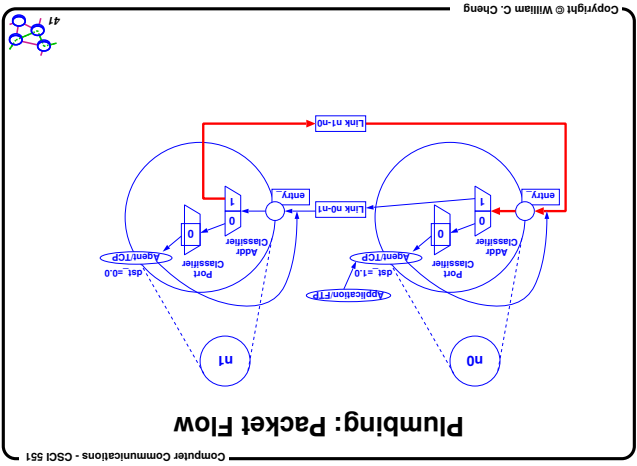
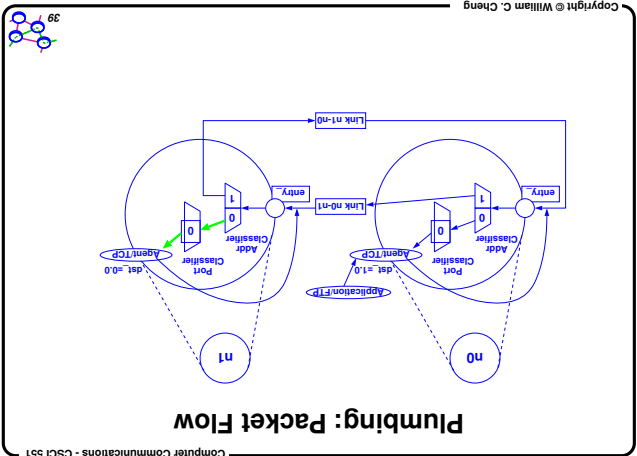
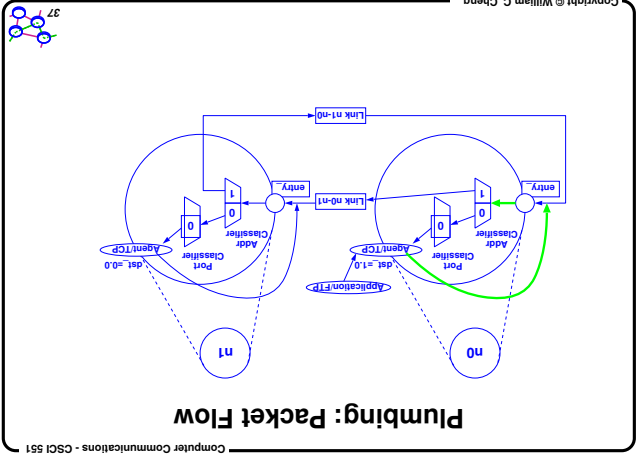
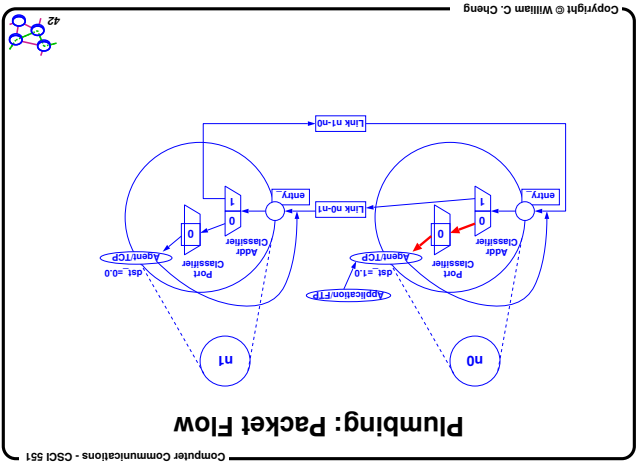
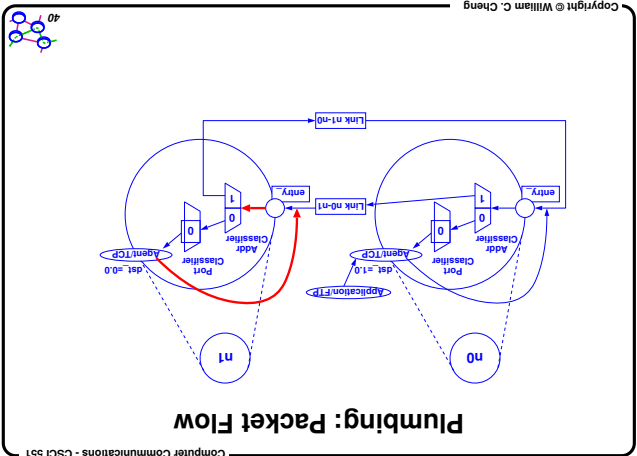
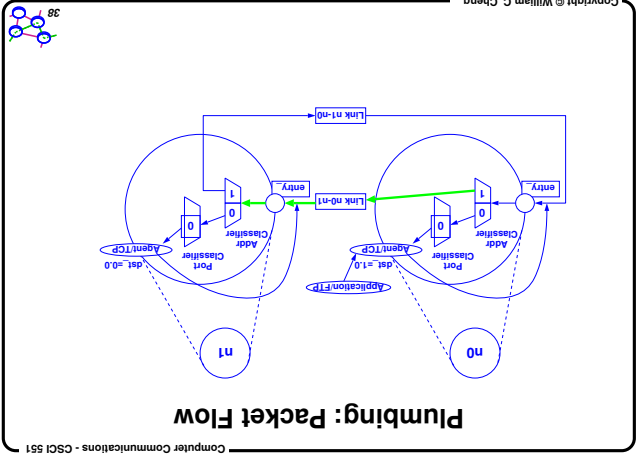
```

$ns lossmodel $loss_module $n0 $n1

```

Computer Communications - CSC1 551





Copyright © William C. Cheng

```

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

```

Summary: Generic Script Structure

Computer Communications - CSC1 551

Copyright © William C. Cheng

```

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

```

TCP : Step 1 Scheduler & tracing

Computer Communications - CSC1 551

Copyright © William C. Cheng

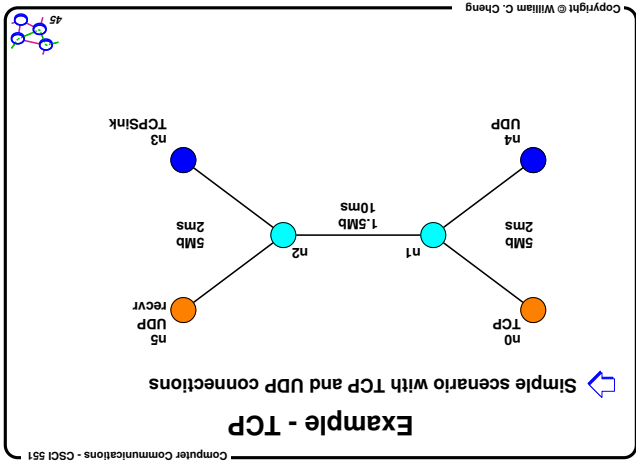
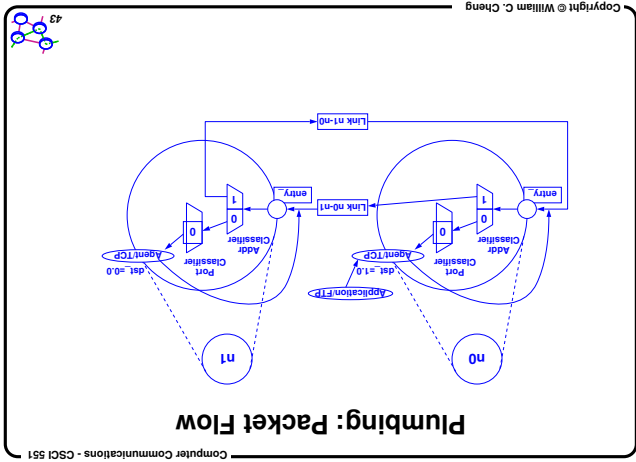
```

#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 3

Computer Communications - CSC1 551



Copyright © William C. Cheng

```


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

```

TCP : Step 2 Create topology

Computer Communications - CSC1 551

Copyright © William C. Cheng



TCP : Step 4

```

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

```

Create TCP agents

```


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

```

Attach traffic

Computer Communications - CSC1 551

Copyright © William C. Cheng



TCP : Step 5

```

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

```

Attach traffic

TCP : Step 6

```

$ns at 2.0 "finish"
proc finish {} {
    global ns f
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}
$ns run

```


End of simulation wrapper (as usual)

Viz Tools

- Nam-1 (Network Animator Version 1)
- Packet-level animation
- Well-supported by ns
- Xgraph
- Convert trace output into xgraph format

Computer Communications - CSC1 551

Copyright © William C. Cheng



Ns-nam Interface

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

Nam Interface: Color

```

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

```

Color mapping

```


$tcp0 set fid_40 # red packets
$tcp1 set fid_41 # blue packets

```

Color ↔ flow id association

Computer Communications - CSC1 551

Copyright © William C. Cheng



Nam Interface: Color

```

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

```

Color mapping

```


$tcp0 set fid_40 # red packets
$tcp1 set fid_41 # blue packets

```

Color ↔ flow id association

Computer Communications - CSC1 551

Copyright © William C. Cheng




Ns-nam Interface

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

Computer Communications - CSC1 551

Copyright © William C. Cheng



TCP : Step 4

```

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

```

Create TCP agents

```


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

```

Attach traffic

Computer Communications - CSC1 551

Copyright © William C. Cheng



TCP : Step 5

```

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

```

Attach traffic

TCP : Step 6

```

$ns at 2.0 "finish"
proc finish {} {
    global ns f
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}
$ns run

```


End of simulation wrapper (as usual)

Viz Tools

- Nam-1 (Network Animator Version 1)
- Packet-level animation
- Well-supported by ns
- Xgraph
- Convert trace output into xgraph format

Computer Communications - CSC1 551

Copyright © William C. Cheng




Ns-nam Interface

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

Computer Communications - CSC1 551

Copyright © William C. Cheng



Nam Interface: Color

```

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

```

Color mapping

```

$tcp0 set fid_40 # red packets
$tcp1 set fid_41 # blue packets

```

Color ↔ flow id association

Computer Communications - CSC1 551

Copyright © William C. Cheng

55

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 \"
```
- Color


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


```
$ns duplex-link-op $n0 $n1 label "abcd"
```
- Dynamics (automatically handled)


```
$ns rtmodel Deterministic {2.0 0.9 0.1} $n0 $n1
```
- Asymmetric links not allowed

Computer Communications - CSC1 551

Copyright © William C. Cheng

58

Nam Interface: Protocol State

Monitor values of agent variables

```
$ns add-agent-trace $rm0 srm_agent0
$ns monitor-agent-trace $rm0
$rm0 tracevar C1
$rm0 tracevar C2
# ...
$ns delete-agent-trace $tcp1
```

Computer Communications - CSC1 551

Copyright © William C. Cheng

60

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-Debugging.html>
- Topology generator
 - <http://www.isi.edu/nsnam/ns-topogen.html>
- Scenario generator
 - <http://www.isi.edu/nsnam/ns-scenegeration.html>

Computer Communications - CSC1 551

Copyright © William C. Cheng

57

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

Computer Communications - CSC1 551

Copyright © William C. Cheng

59

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"
```

Computer Communications - CSC1 551

Copyright © William C. Cheng

60

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-Debugging.html>
- Topology generator
 - <http://www.isi.edu/nsnam/ns-topogen.html>
- Scenario generator
 - <http://www.isi.edu/nsnam/ns-scenegeration.html>

Computer Communications - CSC1 551

Copyright © William C. Cheng

64

Resources (cont..)

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

Computer Communications - CSCI 551

Copyright © William C. Cheng

63

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

Computer Communications - CSCI 551

Copyright © William C. Cheng

62

Other Ns Features

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

Computer Communications - CSCI 551

Copyright © William C. Cheng

61

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)

Computer Communications - CSCI 551