

## ns-2 Tutorial

Polly Huang  
AT&T Labs Research  
huang@catarina.usc.edu  
<http://netweb.usc.edu/huang>  
11 August, 1999

1

## Essentials & Getting Started

2

## Outlines

- **Essentials**
- Getting Started
- Fundamental tcl, otcl and ns
- Case Studies
  - Web, TCP, Routing, Queuing

3

## Object-Oriented

- + Reusability
- + Maintainability
- Careful Planning Ahead

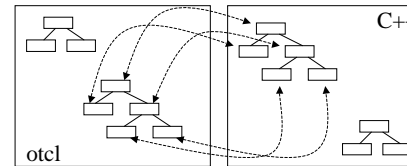
4

## C++ and otcl Separation

- C++ for data
  - per packet action
- otcl for control
  - periodic or triggered action
- + Compromise between composibility and speed
- Learning & debugging

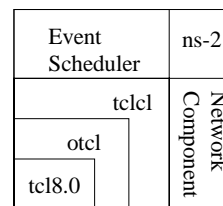
5

## otcl and C++: The Duality



6

## tcl Interpreter With Extents



- otcl: Object-oriented support
- tclcl: C++ and otcl linkage
- Discrete event scheduler
- Data network (the Internet) components

7

## Outlines

- Essentials
- **Getting Started**
- Fundamental tcl, otcl and ns
- Case Studies
  - Web, TCP, Routing, Queuing

8

## Installation

- Getting the pieces
  - tcl/tk8.0, otcl, tclcl, ns-2, (and nam-1)
- <http://www-mash.cs.berkeley.edu/ns/ns-build.html>
- ns-users@mash.cs.berkeley.edu
  - majordomo@mash.cs.berkeley.edu
  - subscribe ns-users yourname@address

9

## 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%
```

10

## 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%
```

11

## Outlines

- Essentials
- Getting Started
- **Fundamental tcl, otcl and ns**
- Case Studies
  - Web, TCP, Routing, Queuing, Wireless

12

## Fundamentals

- tcl
- otcl:
  - <ftp://ftp.tns.lcs.mit.edu/pub/otcl/doc/tutorial.html>
- ns-2
  - <http://www-mash.cs.berkeley.edu/ns/nsDoc.ps.gz>
  - <http://www-mash.cs.berkeley.edu/ns/ns-man.html>

13

## 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

14

## Basic otcl

```
Class mom  
mom instproc greet {} {  
    $self instvar age_  
    puts "$age_ years old mom:  
    How are you doing?"  
}  
  
    $a greet  
    $b greet  
  
Class kid -superclass mom  
kid instproc greet {} {  
    $self instvar age_  
    puts "$age_ years old kid:  
    What's up, dude?"  
}
```

15

## Basic ns-2

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

16

## 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

17

## 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

18

## 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

19

## Computing routes

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

20

## 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

21

## 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

22

## 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

23

## 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]

24

## 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)

25

## 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

26

## 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]

27

## Outlines

- Essentials
- Getting Started
- Fundamental tcl, otcl and ns-2
- **Case Studies**

28

## Case Studies

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

29

## Visualization Tools

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

30

## ns-2 Internal

31

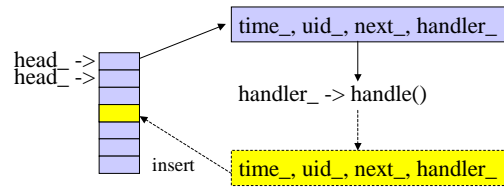
## Internals

- Discrete Event Scheduler
- Network Topology
- Routing
- Transport
- Application
- Packet Flow
- Packet Format

32

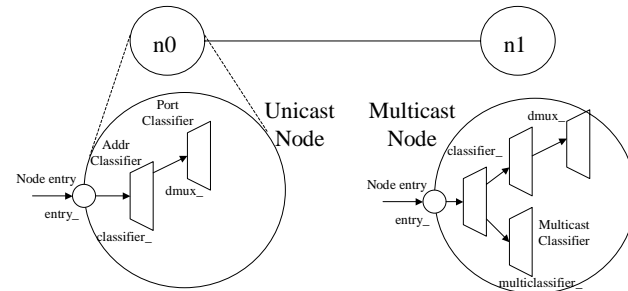


## Discrete Event Scheduler



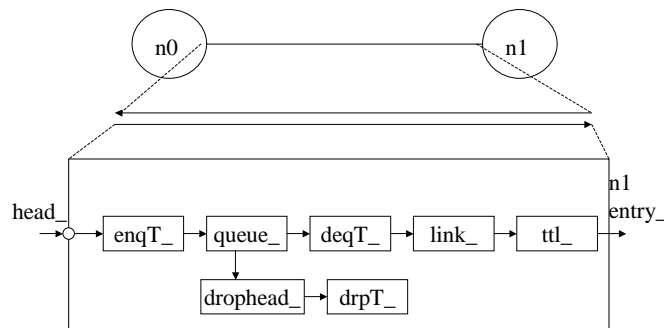
33

## Network Topology - Node



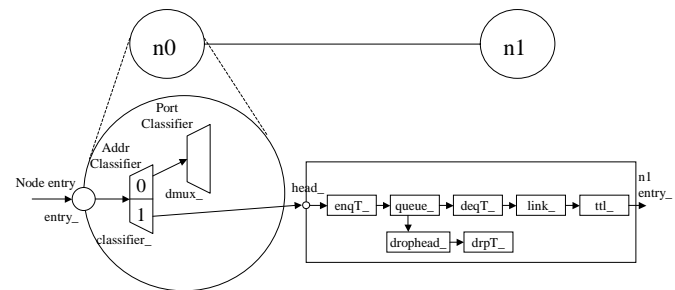
34

## Network Topology - Link



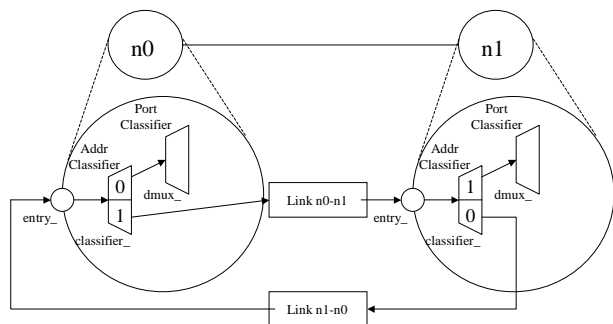
35

## Routing



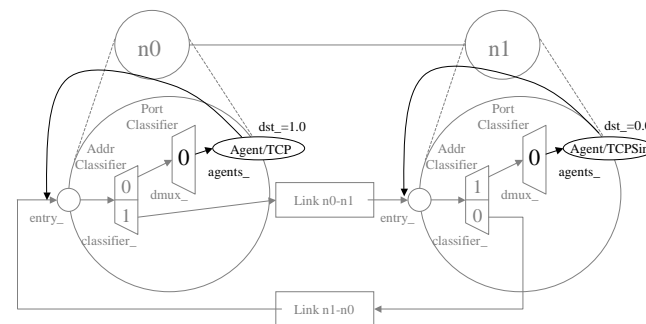
36

## Routing (cont.)



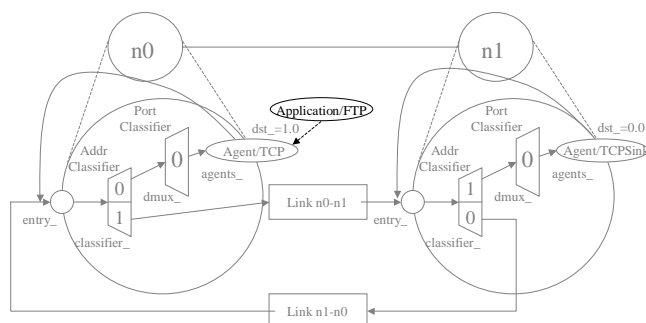
37

## Transport



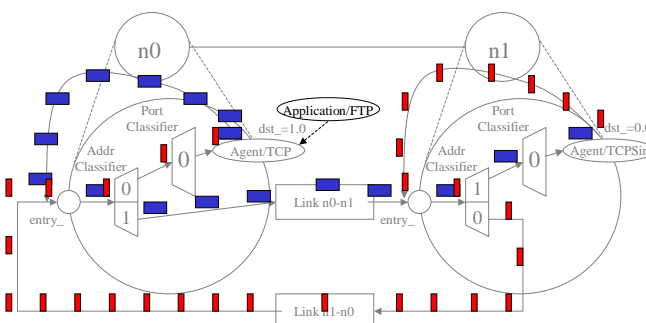
38

## Application



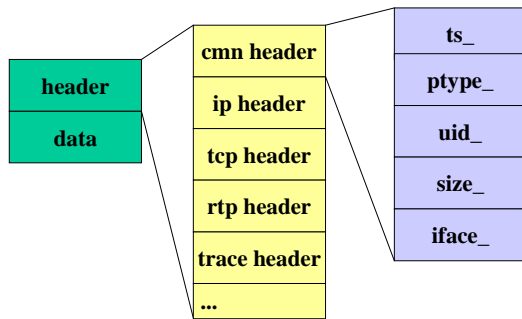
39

## Packet Flow



40

## Packet Format



41

## Extending ns-2 Simulator

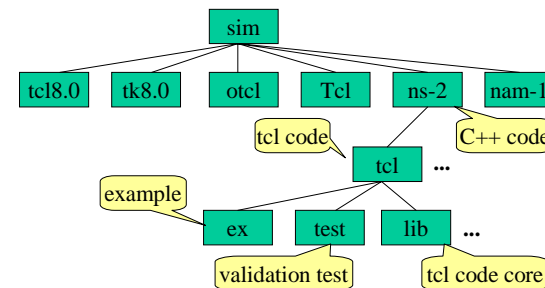
42

## Outline

- **Making changes**
- Creating new components

43

## ns-2 Directory Structure



44

## Making Changes in C++ Space

- Existing code
  - recompile
- Addition
  - change Makefile and recompile

45

## Making Changes in otcl Space

- Existing code
  - recompile
  - source
- Addition
  - source
  - change Makefile (NS\_TCL\_LIB), tcl/ns-lib.tcl (source) and recompile

46

## Outline

- Making changes
- **Creating new components**

47

## Creating New Components

- Guidelines
- Inheritance Hierarchy
- C++ and otcl Interface
- Examples
  - Network layer
  - Transport layer
  - Application layer

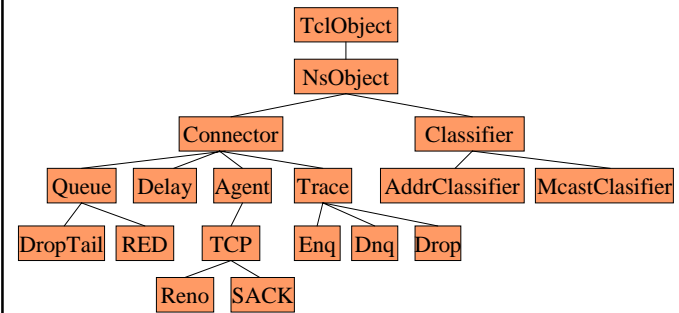
48

## Guidelines

- Decide its inheritance structure
- Create the class and fill in the API virtual functions
- Define otcl linkage functions
- Write the necessary otcl code to access your agent

49

## Class Hierarchy (Partial)



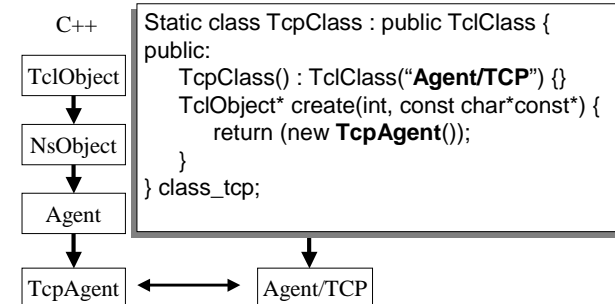
50

## C++ and otcl Linkage

- TclClass
- TclObject: bind() method
- TclObject: command() method
- class Tcl

51

## TclClass



52

## TclObject: bind()

- C++

```
TcpAgent::TcpAgent() {
    bind("window_", $wnd_);
    ...
}
```
- otcl  
Agent/TCP set **window\_ 50**

53

## TclObject: command()

- C++

```
Int TcpAgent::command(int argc, const char*const* argv) {
    if (argc == 3) {
        if (strcmp(argv[1], "advance") == 0) {
            int newseq = atoi(argv[2]);
            ...
            return(TCL_OK);
        }
    }
    return (Agent::command(argc, argv);
}
```
- otcl  
set tcp [new Agent/TCP]  
\$tcp **advance 10**

54

## Class Tcl

- C++

```
Tcl& tcl = Tcl::instance();
if (argc==2) {
    if (strcmp (argv[1], "now") == 0) {
        tcl.resultf("%g", clock());
        return TCL_OK;
    }
    tcl.evalc("puts hello, world");
    tcl.error("command not found");
}
```
- otcl  
foo now  
foo whatever

55

## Debugging

- `printf()` and `puts ""`
- `gdb`
- `tcl debugger`
  - <http://expect.nist.gov/tcl-debug/>
  - place **debug 1** at the appropriate location
  - trap to debugger from the script
  - single stepping through lines of codes
  - examine data and code using Tcl-ish commands

56

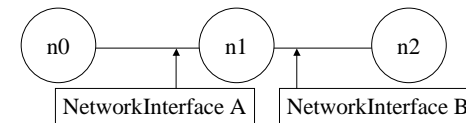
## Case Studies

- Network Layer: Network Interface (packet labeler)
- Transport Layer: TCP Jump Start
- Application Layer: Message agent (ping)

57

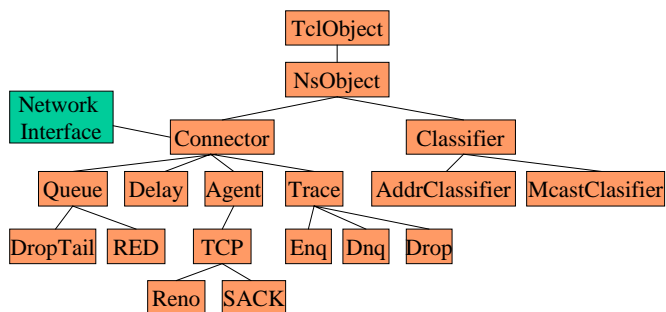
## Case 1: Network Layer

- Network Interface - Packet Labeler



58

## Class Hierarchy



59

## Network Interface Labeler

- class NetworkInterface

– NetworkInterface()

– recv(pkt)

- pkt->iface() = label

– send(pkt)

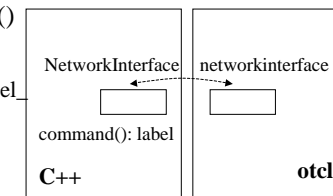
- target->recv(pkt)

– command()

- label argv {label\_ = argv}

```
set iface [new networkinterface]
$iface label A
```

- TclClass(“networkinterface”)



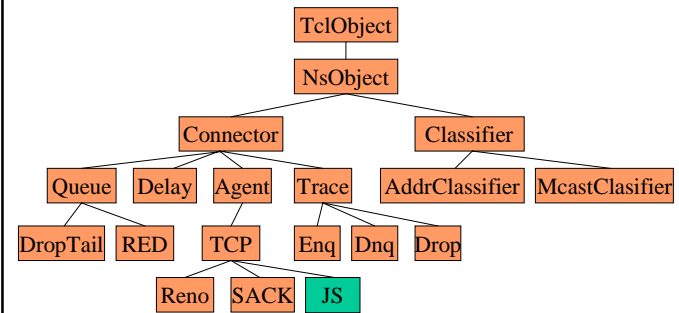
60

## Case 2: Transport Layer

- TCP Jump Start
  - from cwnd += 1
  - to cwnd = maxwin

61

## Class Hierarchy



62

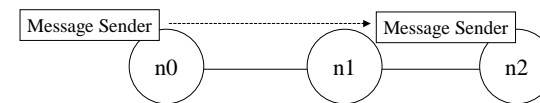
## TCP Jump Start

- class JSTcpAgent
  - openwin()
  - slowdown()
- TclClass("Agent/TCP/JS")

63

## Case 3: Application Layer

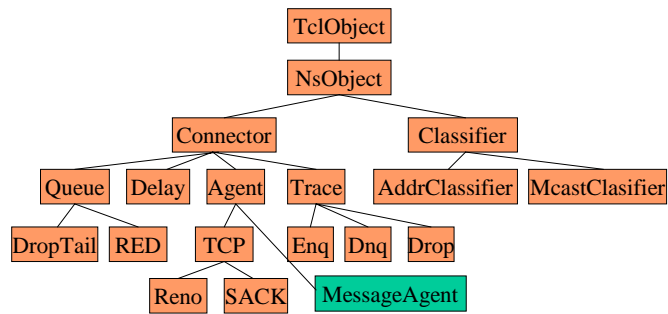
- Message sender (ping)



64



## Class Hierarchy

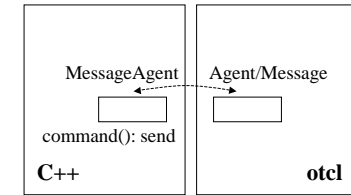


65

## Message Sender

- class MessageAgent

- MessageAgent()
- recv()
- send()
- command()
- send {send() }



- TclClass("Agent/Message")

```

set msg [new Agent/Message]
$msg send
  
```

66

## Special Topics

67

## Special Topics

- Application Level Support
- Topology and Scenario Generation
- Wireless/Mobility Support

68

## 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

69

## 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

70

## Topology Generation

- <http://www-mash.cs.berkeley.edu/ns/ns-topogen.html>
- | Packages | Graph Type    | Edge Method  |
|----------|---------------|--------------|
| • ntg    | • n-level     | • prob       |
| • GT-ITM | • flat, ts, n | • various    |
| • TIERS  | • 3-level     | • spanning t |
| • rtg    | • flat        | • waxman     |

71

## Scenario Generation

- <http://www-mach.cs.berkeley.edu/ns/ns-scengeneration.html>
- agent-gen-script.tcl
- Source generator files
  - source topo-gen.tcl
  - source agent-gen.tcl
  - source route-gen.tcl

72

## topo-gen.tcl

- GT-ITM
- topology
  - outfile <file>
  - type <graph\_type> : random or transit\_stub
  - nodes <num\_nodes>
  - connection\_prob <probability>

73

## route-gen.tcl

- Routing
  - outfile <file>
  - unicast <ucast\_type>
  - multicast <mcast\_type>

74

## agent-gen.tcl

- Agents
  - outfile <file>
  - transport <transport\_type>
  - src <application\_type>
  - sink <transport\_sink\_type>
  - num <num\_connections or %>
  - start <start\_time>
  - stop <stop\_time>

75

## Wireless Support: Setup

- set ns [new Simulator]
- set chan [new Channel/WirelessChannel]
- set prop [new Propagation/TwoRayGround]
- set topo [new Topography]
- \$topo load\_flatgrid <length> <width>
- \$prop topology \$topo

76

## Wireless Support: MobileNode

- Creating mobile nodes
  - set mnode [<routing>-create-mobile-node <node\_id>]
  - <routing>: dsdv or dsr
- Node coordinates
  - \$mnode set X\_ <x>
  - \$mnode set Y\_ <y>
  - \$mnode set Z\_ 0

77

## Mobility Support: Movement

- Specified
  - \$ns at 1.0 "\$mnode setdest <x> <y> <speed>"
- Random
  - \$ns at 1.0 "\$mnode start"

78