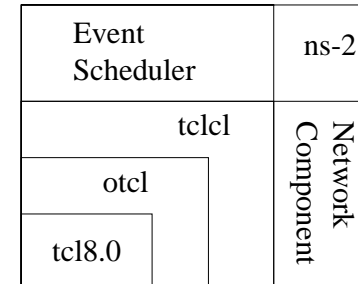


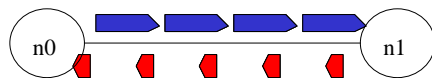
ns-2 201

Polly Huang
USC/ISI
huang@isi.edu

tcl Interpreter With Extensions



Example Script



```
set ns [new Simulator]          set ftp [new Application/FTP]
set n0 [$ns node]              $ftp attach-agent $tcp
set n1 [$ns node]              $ns at 0.2 "$ftp start"
$ns duplex-link $n0 $n1 1.5Mb  $ns at 1.2 "exit"
    10ms DropTail              $ns run
set tcp [$ns create-connection]
    TCP $n0 TCPSink $n1 0]
```

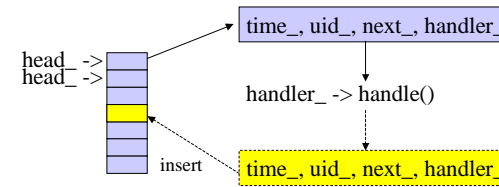
Outline

- **Internals**
- Making changes
- Creating new components

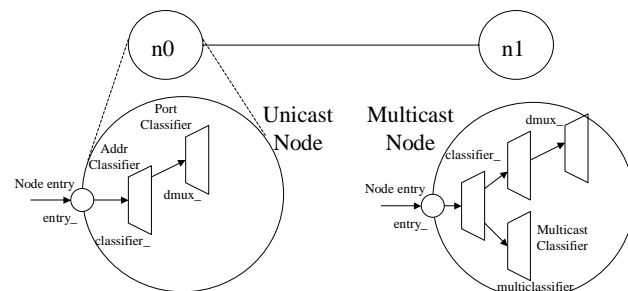
Internals

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

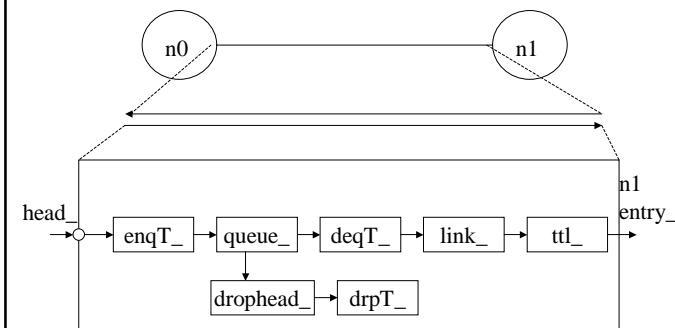
Discrete Event Scheduler



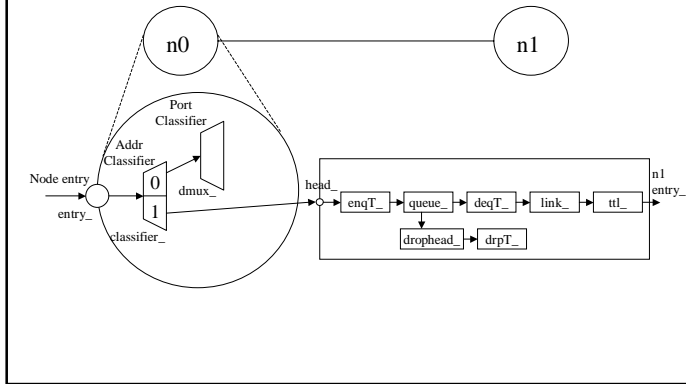
Network Topology - Node



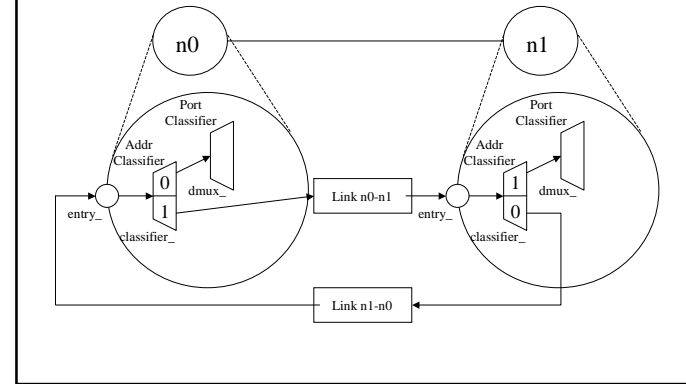
Network Topology - Link



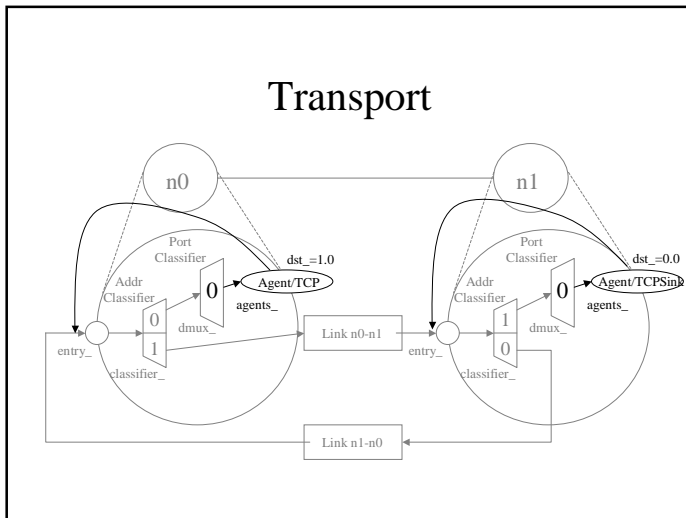
Routing



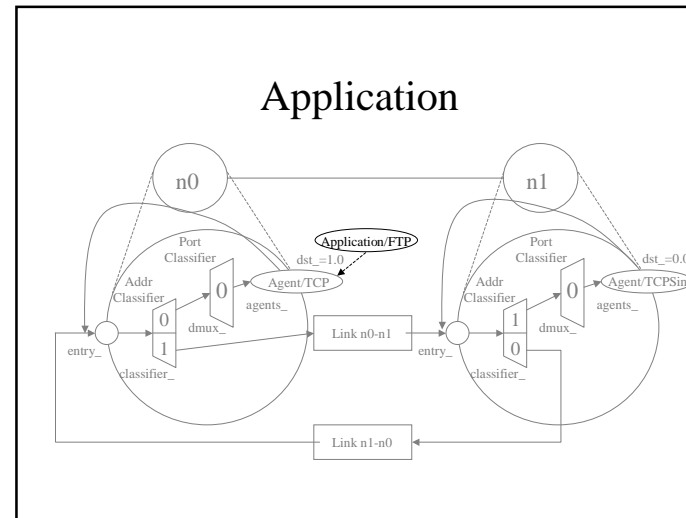
Routing (cont.)



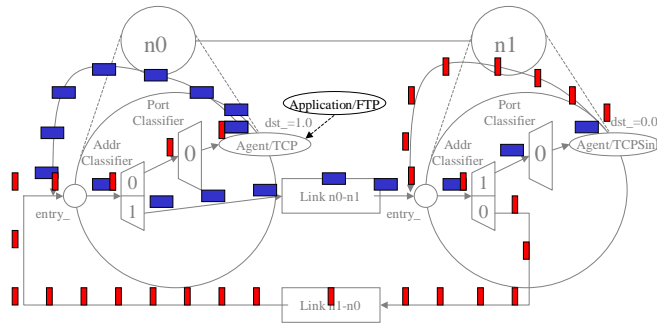
Transport



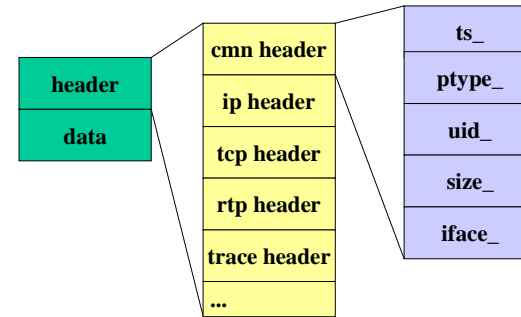
Application



Packet Flow



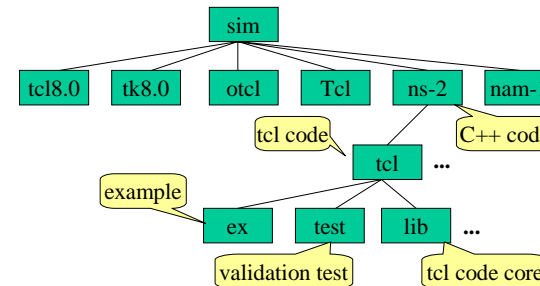
Packet Format



Outline

- Internals
- **Making changes**
- Creating new components

ns-2 Directory Structure



Making Changes in C++ Space

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

Making Changes in otcl Space

- Existing code
 - recompile
 - source
- Addition
 - source
 - change Makefile (NS_TCL_LIB), tcl/ns-lib.tcl (source) and recompile

Outline

- Internals
- Making changes
- **Creating new components**

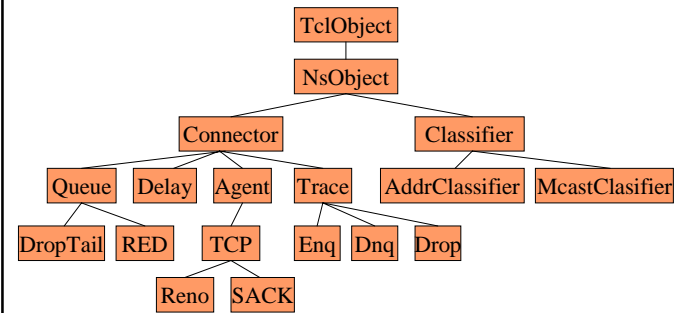
Creating New Components

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

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

Class Hierarchy (Partial)

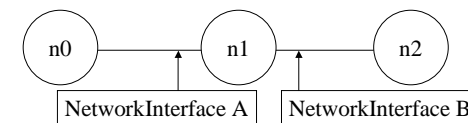


C++ and otcl Interface

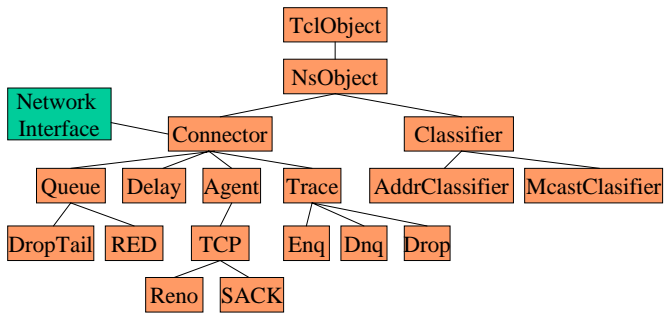
- Supported by tclcl (a.k.a., Tcl)
 - `Tcl& tcl = Tcl::instance();`
- Implementation is 100% in C++
 - otcl -> C++
 - `command(): <otcl command>`
 - `tcl.result()`
 - C++ -> otcl
 - `tcl.eval("<otcl action>")`
 - `bind()` or `[otcl->C++]`

Network Layer

- Network Interface - Packet Labeler



Class Hierarchy



Network Interface Labeler

- class NetworkInterface

- NetworkInterface()

- recv(pkt)

- pkt->iface() = label

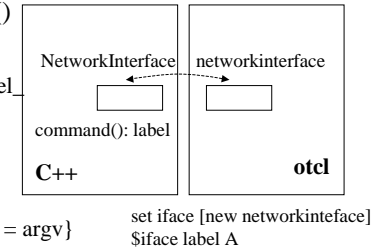
- send(pkt)

- target->recv(pkt)

- command()

- label argv {label_ = argv}

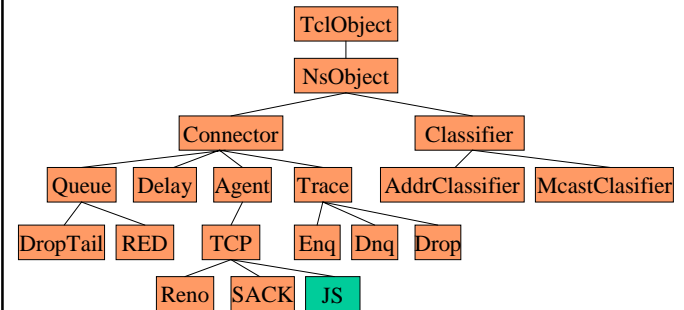
- TclClass("networkinterface")



Transport Layer

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

Class Hierarchy

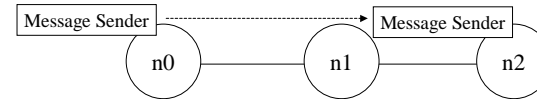


TCP Linear Slow Start

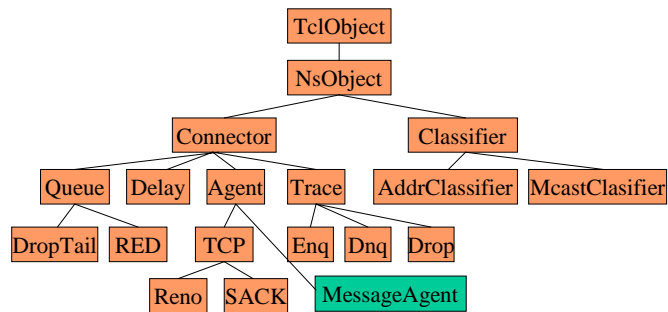
- class JSTcpAgent
 - openwin()
 - slowdown()
- TclClass(“Agent/TCP/JS”)

Application Layer

- Message sender



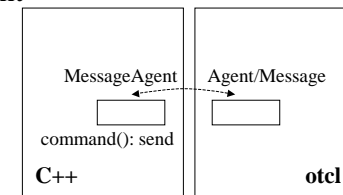
Class Hierarchy



Message Sender

- class MessageAgent

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



- TclClass(“Agent/Message”)

```

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


ns-2 301 Preview

- Generating representative web traffic
 - HttpSession internal - step by step code walk through
- Isolating flow control or error recovery mechanisms from TCP
 - TCP internal - step by step code walk through

Slides

- <http://www.research.att.com/~phuang>