# The Network Simulator ns-2

Joerg Widmer

University of Mannheim

widmer@informatik.uni-mannheim.de

# Outline

- Introduction
- Simulator architecture
- Example simulation
- Visualization and analysis
- Features
- Resources

# Introduction

Why network simulations?

- Controlled environment necessary
- But: capturing all the details of real-world scenarios is impossible

Real-world experiments and simulations

- Standard platform for protocol development
  - Users from ca. 600 institutes, 50 countries

# ns Architecture

Discrete event simulator

- ○ Object-oriented
- ○ Modular
- ○ Extensible framework

- ○ Developed by UCB, LBNL, ISI/USC, CMU, ...
- ○ About 100K lines of C++, 70K lines of OTcl code, and 50K lines of examples and documentation

# Platforms

- Most Unix systems
  - Linux
  - FreeBSD, NetBSD, ...
  - Sun Solaris
  - HP UX, SGI
- Windows 95/98/NT

# Words of Caution

- Not a finished product
  - Bugs
  - Changes of the architecture

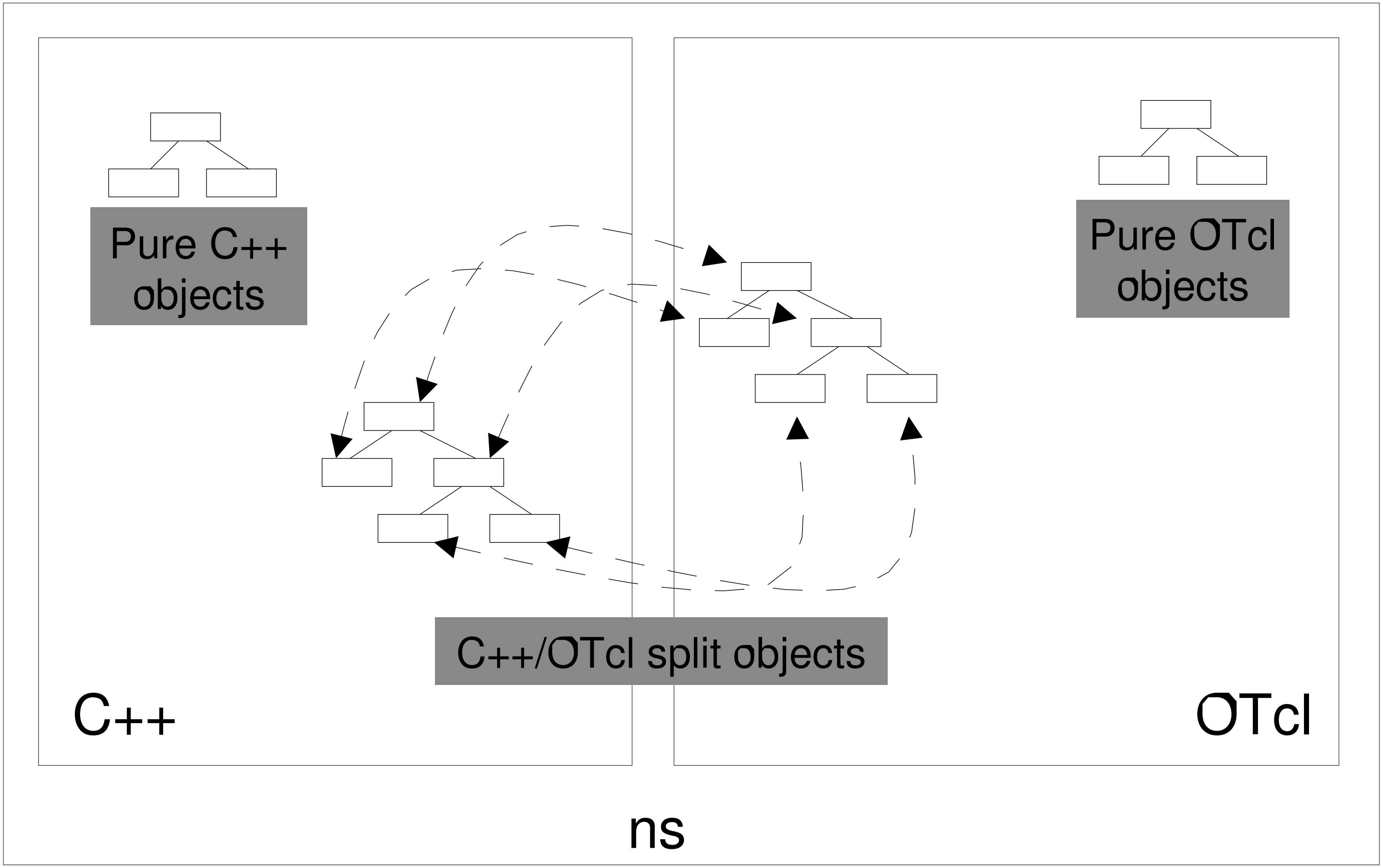
- Users need to verify that
  - their simulations are not invalidated by bugs
  - the model implemented in ns conforms to what they expect

# Split-language Programming

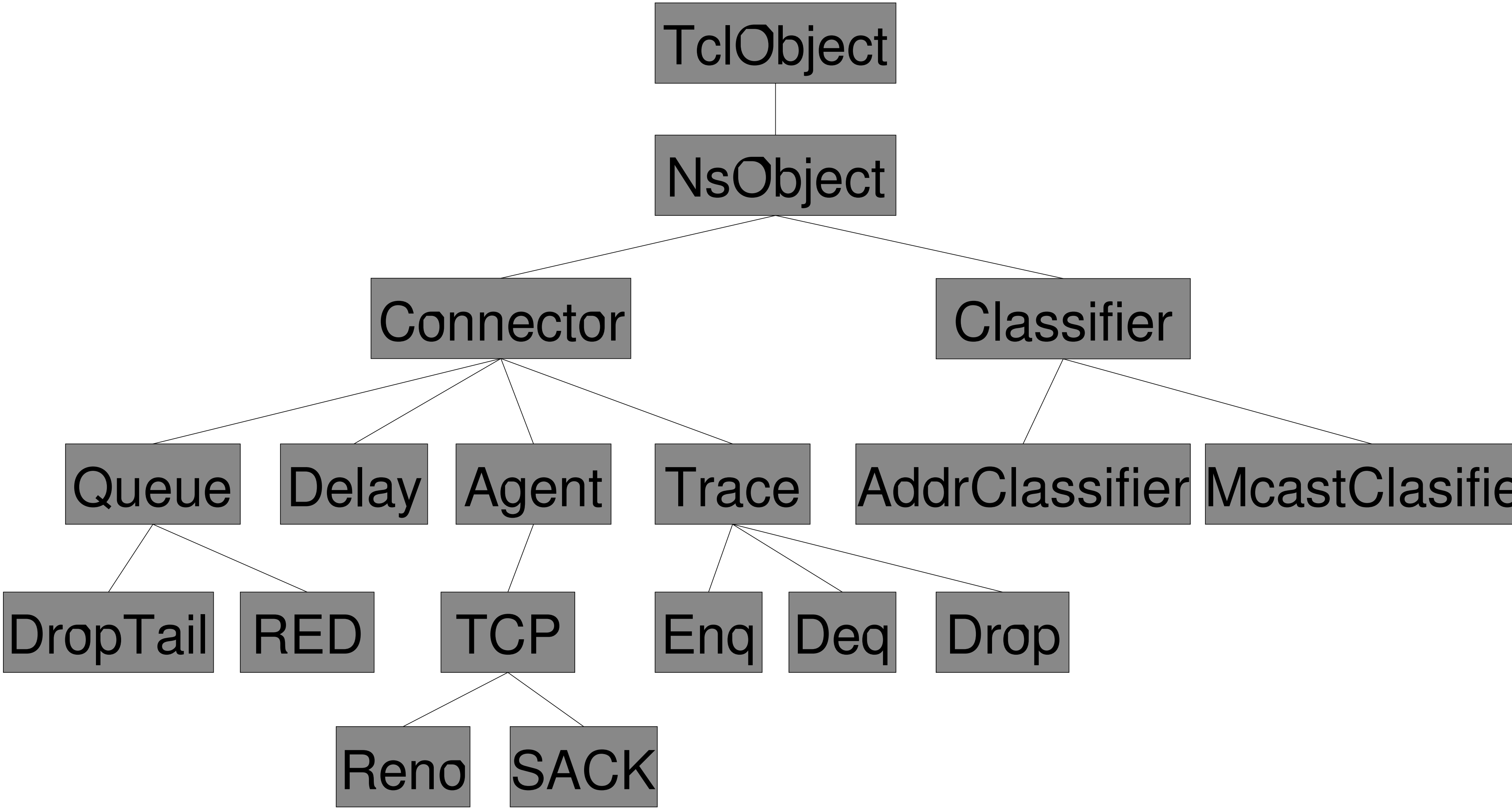
- C++ for the core components
  - (low level event processing, packet forwarding, etc.)
- OTcl for control operations
  - (to build the simulation scenario, model dynamic configurations, etc.)
- TclCL as link between C++ and OTcl

- Necessary to know both languages
- Difficult to debug

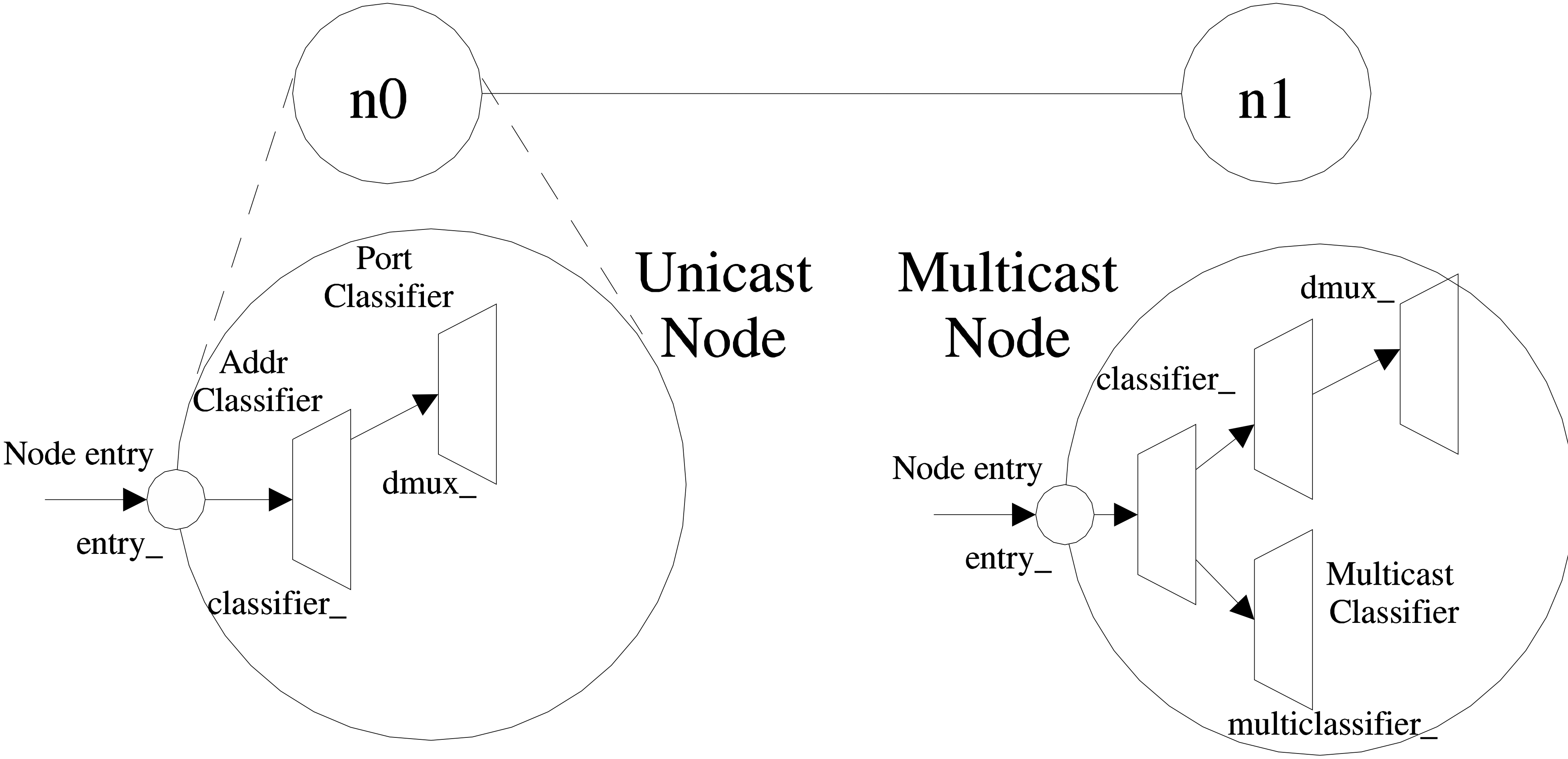
# OTcl and C++


Pure C++ objects
Pure OTcl objects
C++/OTcl split objects
C++
OTcl
ns

# (Partial) Class Hierarchy



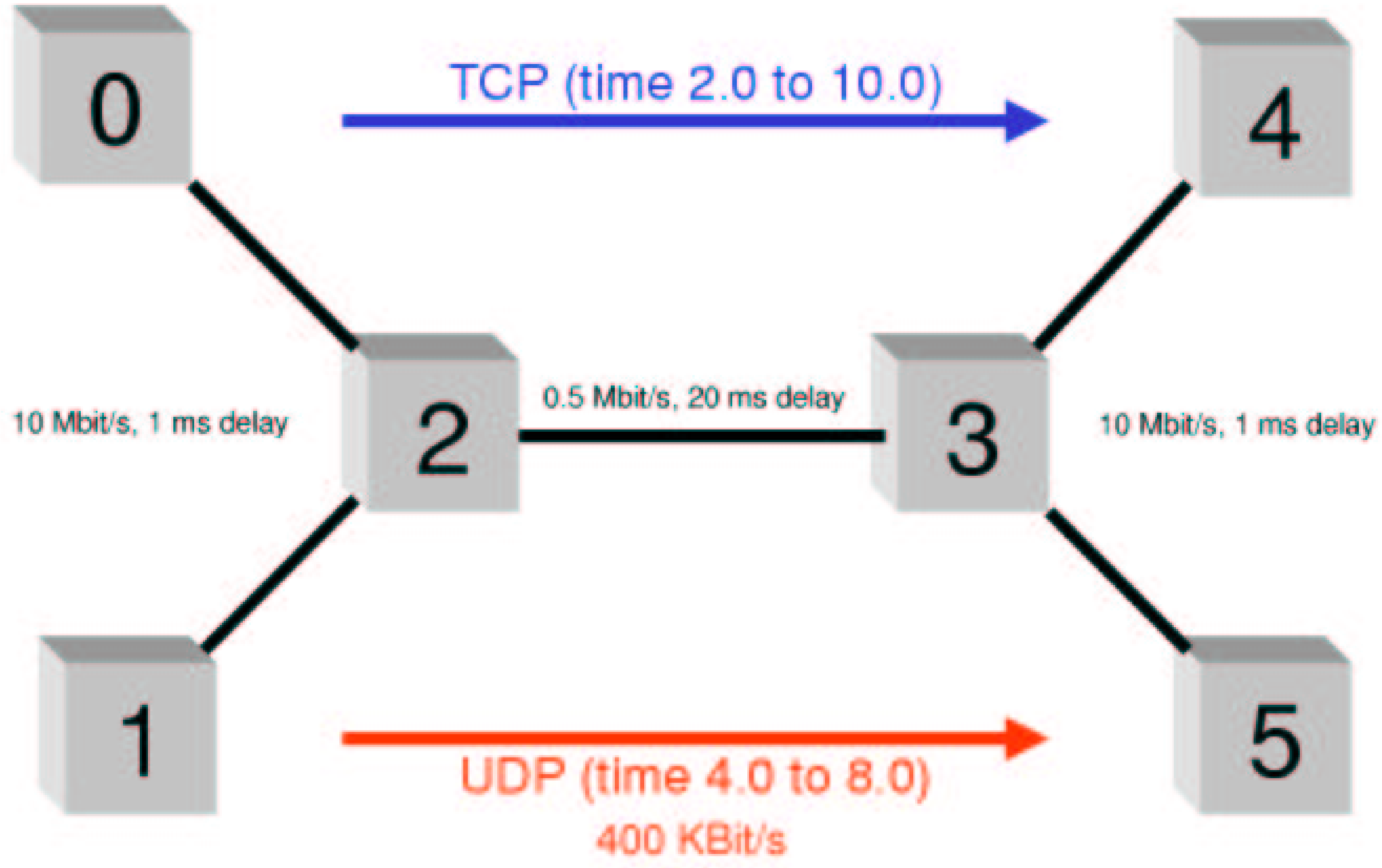
# Node Structure



# Creating a Simulation

- Create the event scheduler
- Create the network topology
- Specify traffic patterns
- Insert errors, modify network conditions, ...
- Tracing
- Visualization and analysis

# Example Setup



# Simulator Object and Tracing

```
set ns [new Simulator]

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

# Network Topology

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n2  10Mb  1ms DropTail
$ns duplex-link $n1 $n2  10Mb  1ms DropTail
$ns duplex-link $n2 $n3 500Kb 20ms DropTail
$ns duplex-link $n3 $n4  10Mb  1ms DropTail
$ns duplex-link $n3 $n5  10Mb  1ms DropTail
```

# Traffic Agents

- TCP

```
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n4 $sink
$ns connect $tcp $sink

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

# Traffic Agents

- UDP

```
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n1 $udp
$ns attach-agent $n5 $null
$ns connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetSize_ 1000
$cbr set rate_ 400000
```

# ... and start the simulation

```
$ns at 2.0 "$ftp start"
$ns at 4.0 "$cbr start"
$ns at 8.0 "$cbr stop"
$ns at 10.0 "$ns flush-trace; close $f;
            close $nf; exit 0"

$ns run
```

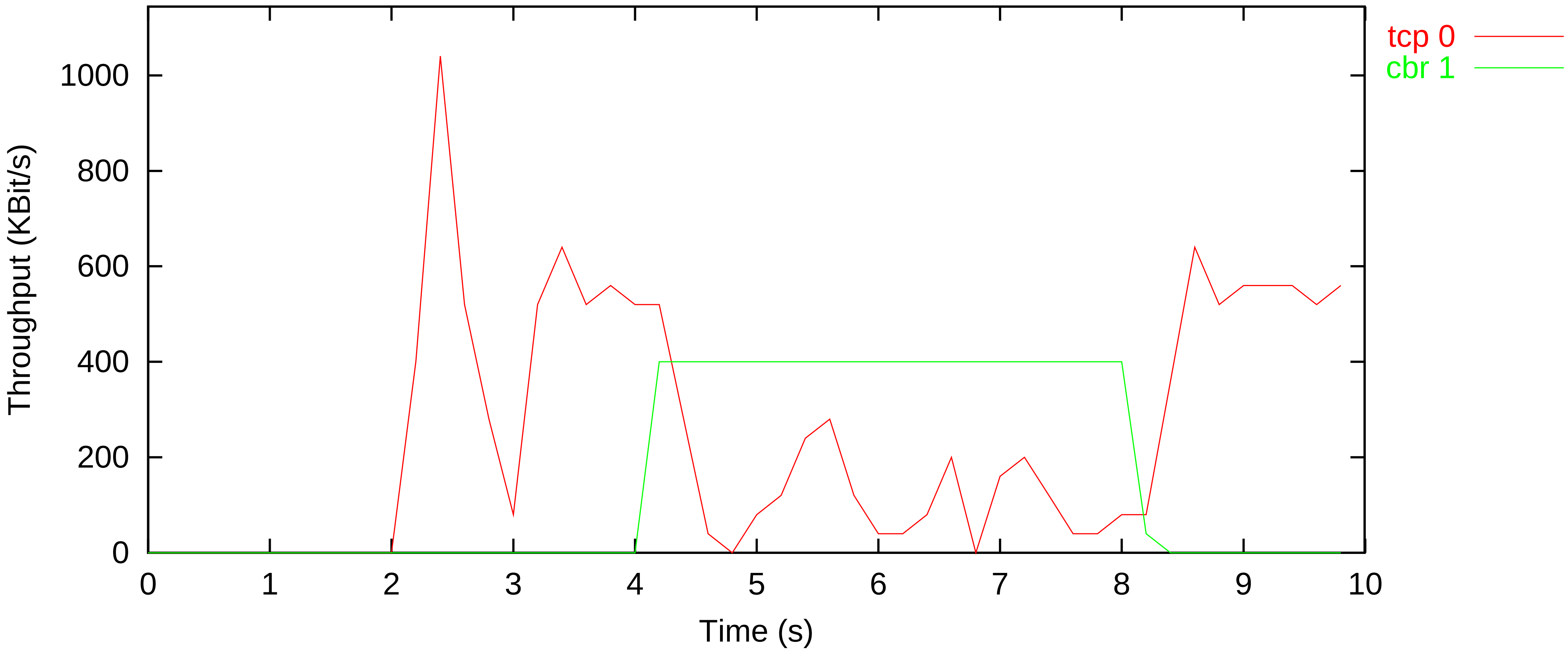
# Trace File Format

| | | nodes: | | packet: | | | | adress: | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| event | time | from | to | type | size | flags | flow ID | src | dest | seq-no | uid |

```
+ 4.053333 1 2 cbr 1000 ------- 1 1.0 5.0 6 415
- 4.053333 1 2 cbr 1000 ------- 1 1.0 5.0 6 415
r 4.054704 0 2 tcp 1000 ------- 0 0.0 4.0 209 411
+ 4.054704 2 3 tcp 1000 ------- 0 0.0 4.0 209 411
r 4.055244 1 2 cbr 1000 ------- 1 1.0 5.0 5 412
+ 4.055244 2 3 cbr 1000 ------- 1 1.0 5.0 5 412
r 4.05552 3 4 tcp 1000 ------- 0 0.0 4.0 198 384
+ 4.05552 4 3 ack 40 ------- 0 4.0 0.0 198 416
- 4.05552 4 3 ack 40 ------- 0 4.0 0.0 198 416
r 4.057552 4 3 ack 40 ------- 0 4.0 0.0 197 413
```

# ... Perl is your friend

## Throughput at Node 2



# Visualization with NAM

- Packet traces presented as graphical animation
- Additional NAM information in TCL trace files (node color, ...)
- Captures simulation dynamics
- "Intuitive" feel for what the protocol is doing

- Trace files, time sequence graphs, etc. are still necessary for in-depth analysis

# Creating your own components

- Look at existing components
- Try to reuse existing modules
- Decide about inheritance, fill in functions
- Linkage to OTcl, implement complementary OTcl classes/functions

- Interaction of C++ and OTcl is one of the most difficult design tasks

# Other Features

- Multicast Routing
- SRM
- RTP/RTCP
- Wireless Networks (WaveLan, Satellite, ...)
- Mobile IP
- QoS (IntServ, DiffServ)

# Other Features

- Automated scenario generation
- Test suites
- Abstraction
- Trace driven simulation
- Network emulation

# Resources

- Website: http://www.isi.edu/nsnam/ns/
  - ns documentation, tutorials, FAQ
  - CVS logs, class hierarchy, ...
- Mailing lists
  - ns-users@isi.edu
  - ns-announce@isi.edu

Much more can be found on the web...