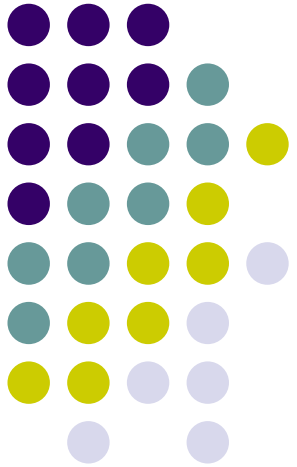
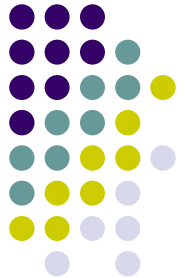
Network Simulator 2

鄭瑞恆 (rhc@hcu.edu.tw)
玄奘人文社會學院 資訊管理學系副教授
2003.08.26

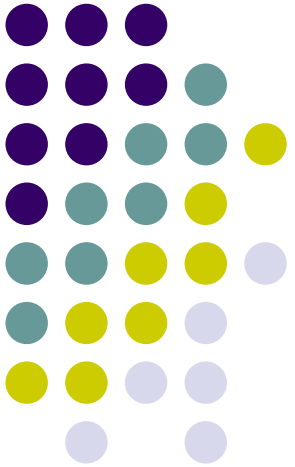


Outline

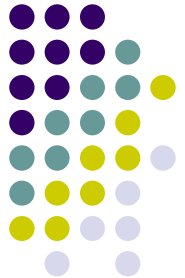


- Ns overview
- Ns programming: A Quick start
- Case study I: A simple Wireless network
- Case study II: Create a new agent in Ns

Ns overview

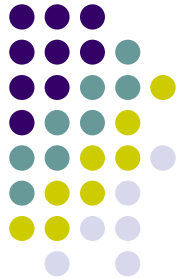


Ns overview



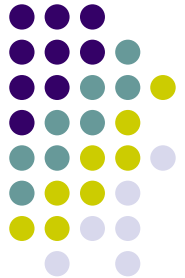
- Ns Status
  - Periodical release (ns-2.26, Feb 2003)
  - Platform support
    - FreeBSD, Linux, Solaris, Windows and Mac
- Ns functionalities
  - Wired
    - Routing, Transportation, Traffic sources,Queuing disciplines, QoS
  - Wireless
    - Ad hoc routing, mobile IP, sensor-MAC
  - Tracing, visualization and various utilities

Ns overview



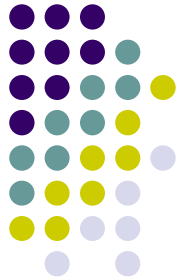
- Componets
  - Ns: a simulator
  - Nam: a network animator
    - Visualize *ns* (or other) output
  - Scenario generator
    - Traffic and topology generators
  - Post-processing:
    - Tracing results analysis

Ns overview



- 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 queueing:
    - Wired routing, ad hoc routing
    - queueing protocols: RED, drop-tail, etc
  - Physical media:
    - Wired (point-to-point, LANs), wireless (multiple propagation models) ...

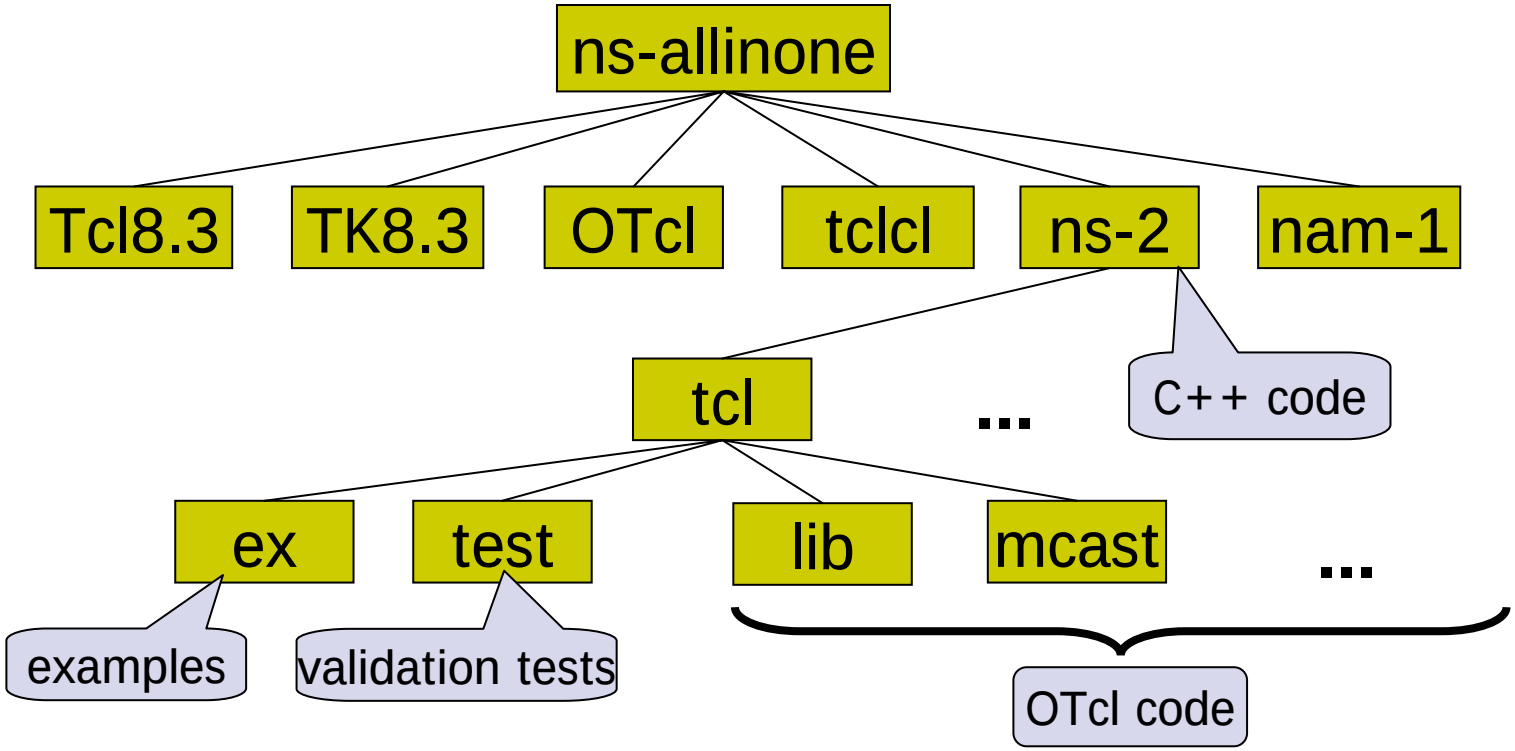
Ns overview



- Installation
  - Getting all-at-once
    - http://www.isi.edu/nsnam
- Help and Resources
  - Ns and nam build questions
  - Ns manual and tutorial

Ns overview

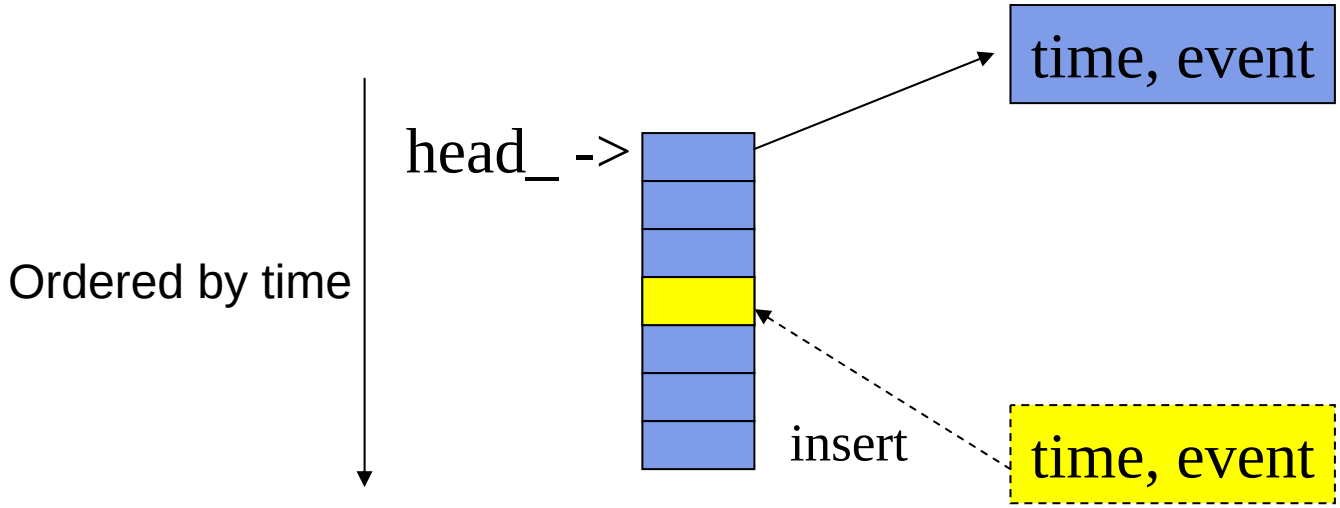
- Ns Directory structure



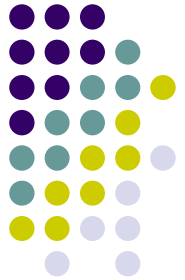
How does the simulator run

- A *discrete event simulator*
  - Model world as *events*
  - Simulator has list of events
  - Simple model: single thread of control
  - Process: take next one, run it, until done
  - Each event happens in an instant of *virtual time*
- C++ and OTcl Separation
  - C++ for “data” and OTcl for control

Discrete Event Scheduler



Demo: Creating Event Scheduler



- Create event scheduler

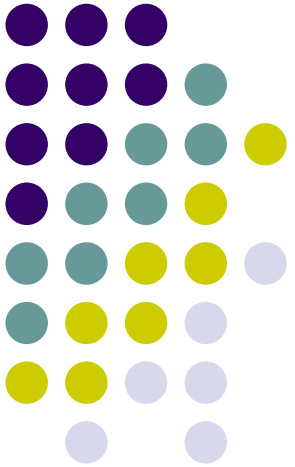
  Example: set ns [new Simulator]
- Create scheduled events

  Format: $ns at <time> <event>

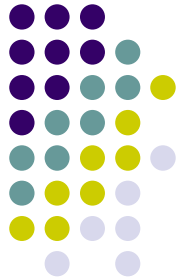
  Example: $ns at 5.0 “exit”
- Start scheduler

  Example: $ns run

Ns programming: A Quick start

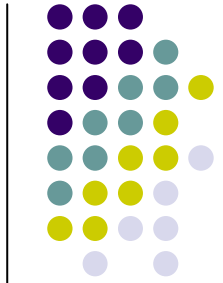


Ns programming



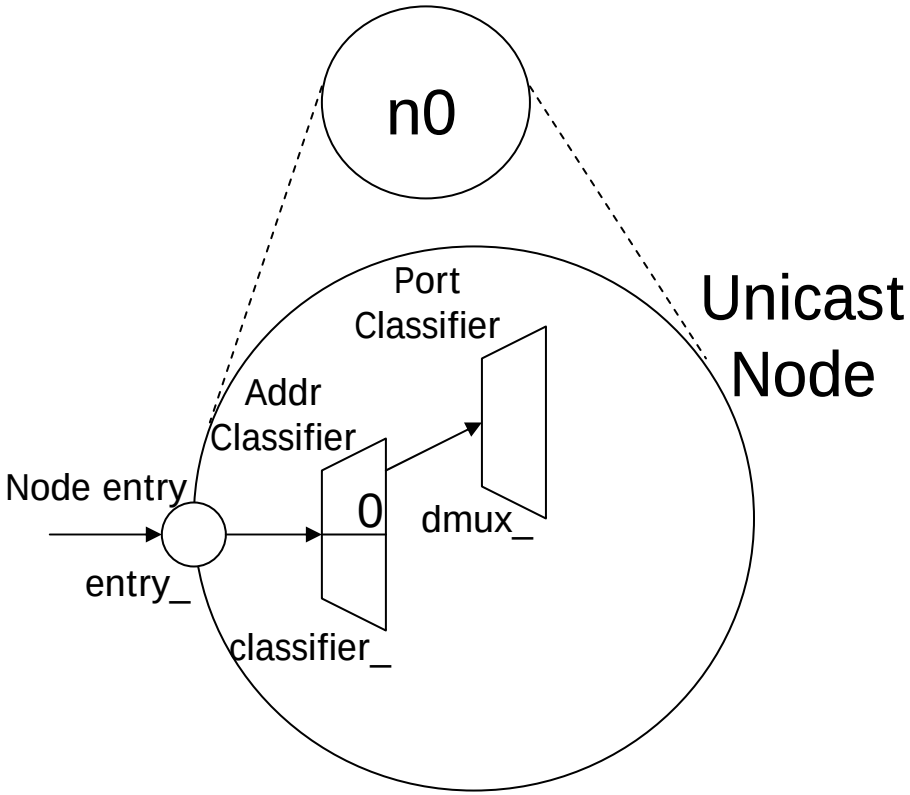
- Create a simulator
- Turn on tracing
- Create network: Nodes and Links
- Select routing algorithm
- Create transport connection
- Create traffic, application ...

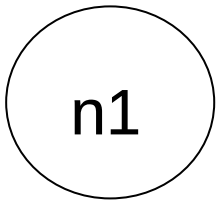
Hello World – Create a simulator



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

Create network: Nodes

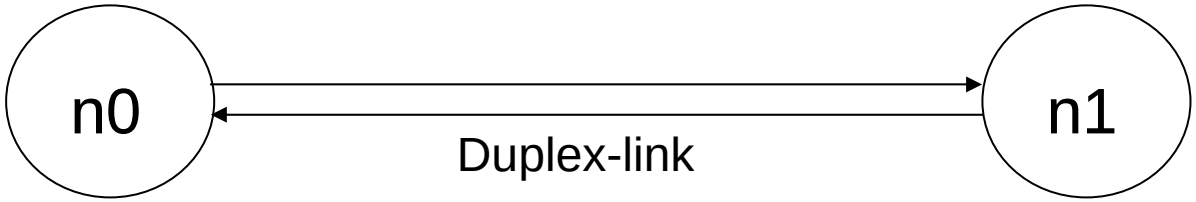




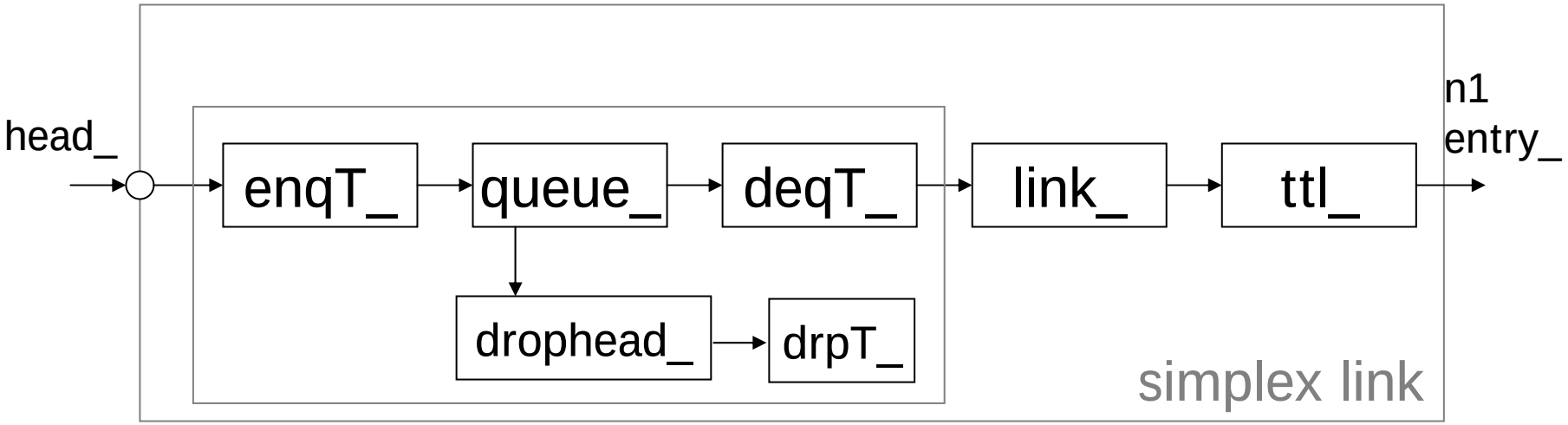
set n0 [$ns node]

Set n1 [$ns node]

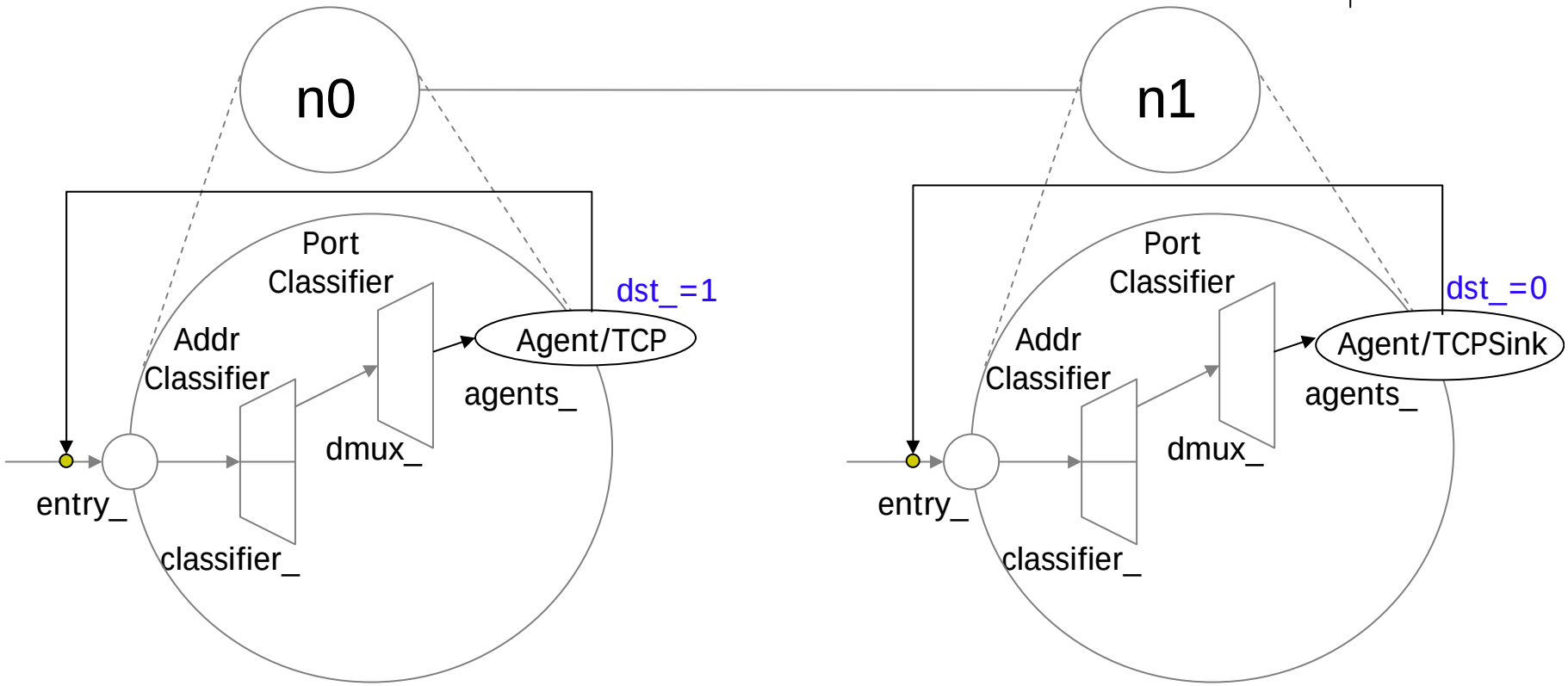
Create network: Links



$ns duplex-link $n0 $n1 1Mb 10ms drop-tail



Create transport connection

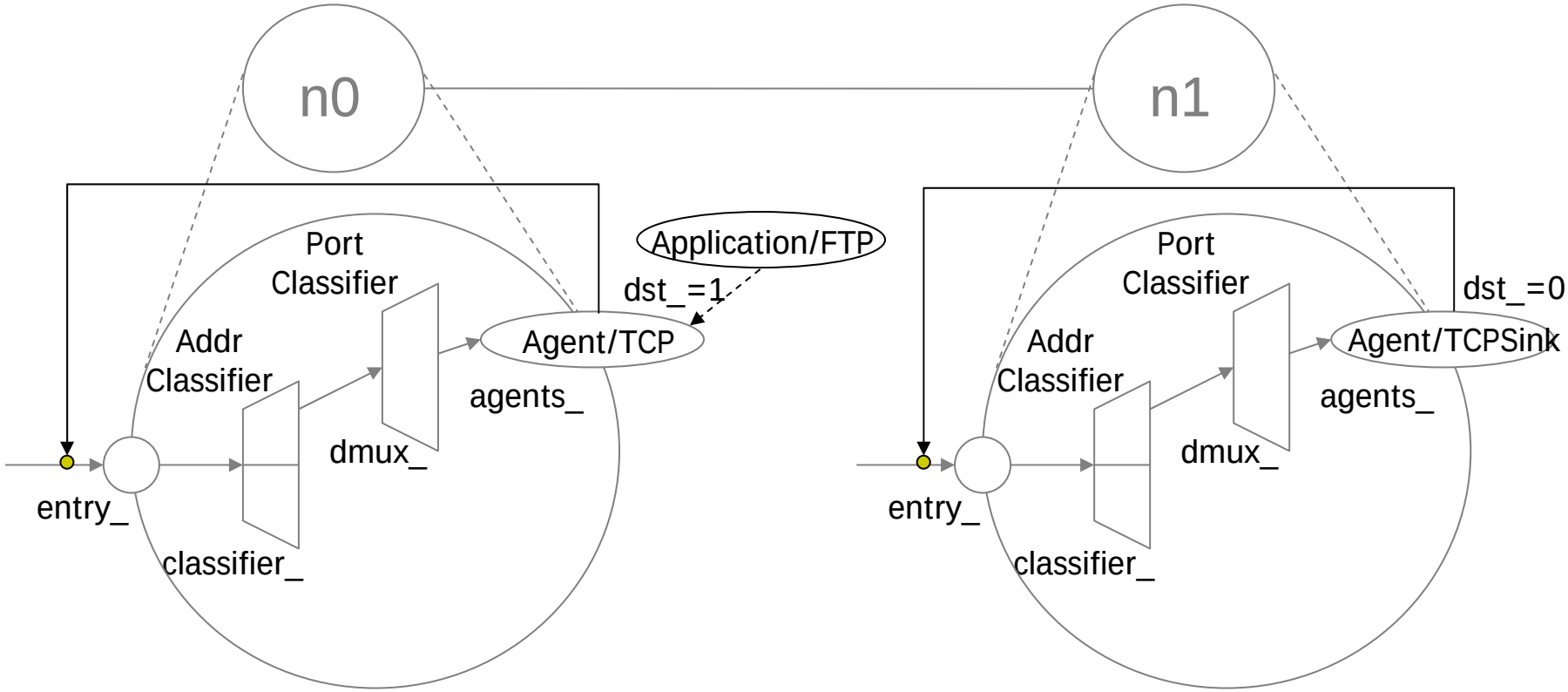


set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp

set tcpsink [new Agent/TCPSink]
$ns attach-agent $n1 $tcpsink

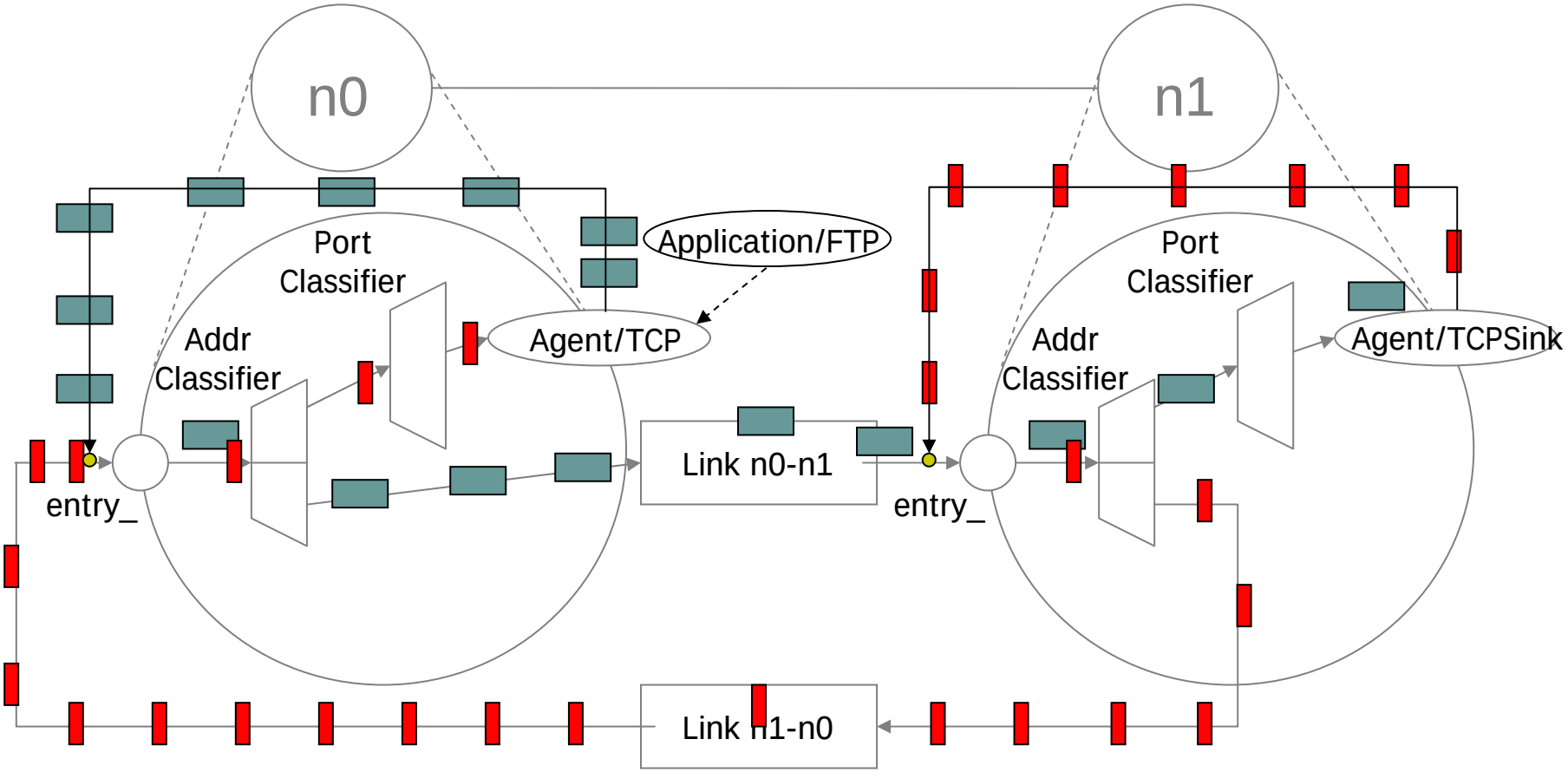
$ns connect $tcp $tcpsink

Application: Traffic Generator

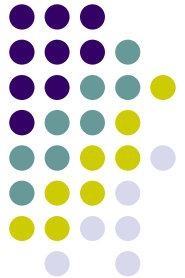


set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.2 “$ftp start”

Simulation result: Packet Flow



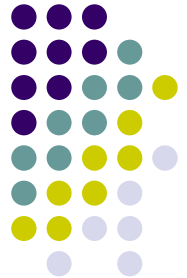
Examples from NS2 Documentation



- Example 1: Setup a network containing only two nodes, and run an FTP application on it.

- Example 2: Setup a network containing five nodes, and run an FTP application on it.
  (using dynamic routing algorithm)

Nam: Network Animator (Node)



- Color

```
$node color red
```

- Shape

```
$node shape box    ;# circle, box, hexagon
```

- Marks

```
$ns at 1.0 “$n0 add-mark m0 blue box”
$ns at 2.0 “$n0 delete-mark m0”
```

- Label

```
$ns at 1.1 “$n0 label \”web cache 0\””
```

Nam: Network Animator (Link)

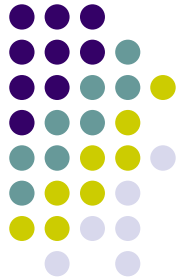
- Color

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

- Label

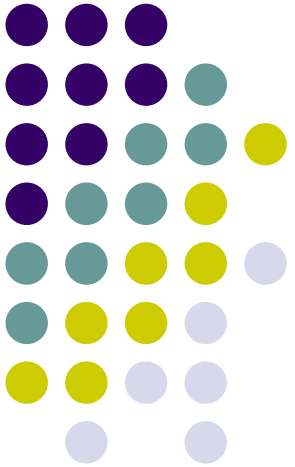
```
$ns duplex-link-op $n0 $n1 label "abced"
```

An example of using nam control command

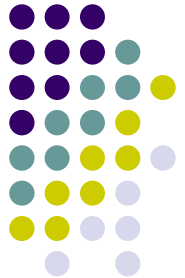


- Add some nam control commands in Exp 2 mentioned before

Case study I: A simple wireless network

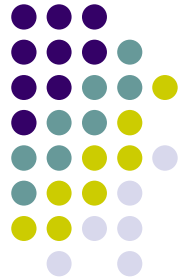


Wireless model



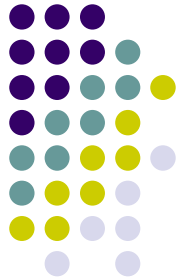
- Mobilenodes can move in a given topology and receive/transmit signals
- Wireless network stack consists of LL, ARP, MAC, IFQ …
- Allow simulations of multi-hop ad hoc networks, wireless LANs, sensor networks …

Scenario



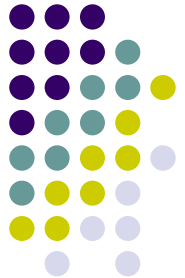
- 5 mobile nodes
- moving within 1000mX1000m flat topology
- using AODV ad hoc routing protocol
- Random mobility model
- CBR traffic

Step 1: Define Global Variables



```
# create simulator
set ns_ [new Simulator]

# create a flat topology in a 1000m x
1000m area
set topo [new Topography]
$topo load_flatgrid 1000 1000
```

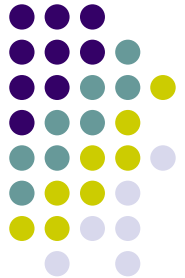
Step 2: Define ns/nam trace



```
# ns trace
set tracefd  [open out.tr w]
$ns trace-all $tracefd
# nam trace
set namtrace [open out.nam w]
$ns namtrace-all-wireless $namtrace 1000
1000
```

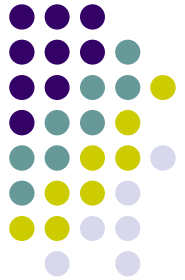
GOD

(General Operations Director)



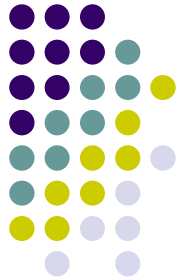
- Stores smallest number of hops from one node to another
- Optimal case to compare routing protocol performance
- set god [create-god <no of mnodes>]
- $god set-dist <from> <to> <#hops>

Step 3: Create God



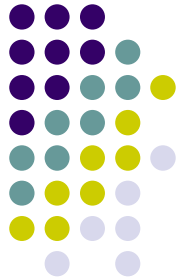
set god [create-god 5]

Step 4: Configure mobile nodes



```
set chan_1_ [new Channel/WirelessChannel]
$ns_ node-config -addressingType flat \
        -adhocRouting AODV \
        -llType LL \
        -macType Mac/802_11 \
        -ifqType Queue/DropTail/PriQueue \
        -ifqLen 50 \
        -antType Antenna/OmniAntenna \
        -propType Propagation/TwoRayGround \
        -phyType Phy/WirelessPhy \
        -channel $chan_1_ \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace ON \
        -movementTrace ON
```

Step 5: Create mobile nodes

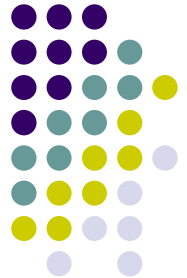


```
set node_(0) [$ns_ node]
# disable random motion
$node_(0) random-motion 0
```

Or

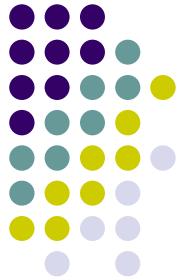
```
# Use "for" loop to create 3 nodes:
for {set i < 0} {$i < 5} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0
}
```

Step 6: Define node movement and traffic model



```
# Define node movement model
source <movement-scenario-files>

# Define traffic model
source <traffic-scenario-files>
```

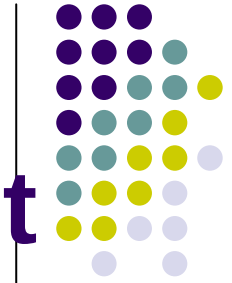
Mobilenode Movement



- Node position defined in a 3-D model
- However z axis is not used

```
$node set X_ <x1>
$node set Y_ <y1>
$node set Z_ <z1>
$node at $time setdest <x2> <y2>
  <speed>
```

Scenario Generator: Movement

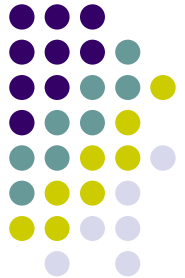


- Mobile Movement Generator

  `setdest -n <num_of_nodes> -p pausetime -s <maxspeed> -t <simtime> -x <maxx> -y <maxy>`

  Source: `ns-2/indep-utils/cmu-scen-gen/setdest/`

A Movement File



```
$node_(2) set Z_ 0.000000000000
$node_(2) set Y_ 199.373306816804
$node_(2) set X_ 591.256560093833
$node_(1) set Z_ 0.000000000000
$node_(1) set Y_ 345.357731779204
$node_(1) set X_ 257.046298323157
$node_(0) set Z_ 0.000000000000
$node_(0) set Y_ 239.438009831261
$node_(0) set X_ 83.364418416244
$ns_ at 50.000000000000 "$node_(2) setdest 369.463244915743 170.519203111152 3.371785899154"
$ns_ at 51.000000000000 "$node_(1) setdest 221.826585497093 80.855495003839 14.909259208114"
$ns_ at 33.000000000000 "$node_(0) setdest 89.663708107313 283.494644426442 19.153832288917"
```

Scenario Generator: Traffic

- Generating traffic pattern files
  - CBR/TCP traffic

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed
   seed] [-mc connections] [-rate rate]
```

  - CBR traffic

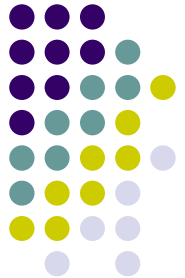
```
ns cbrgen.tcl –type cbr –nn 20 –seed 1 –mc 8  -
   rate 4
```

  - TCP traffic

```
ns cbrgen.tcl –type tcp -nn 15 -seed 0 –mc 6
```

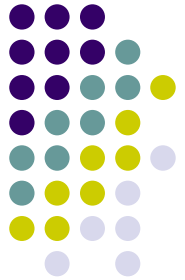
- Source: *ns-2/indep-utils/cmu-scen-gen/*

A Traffic Scenario



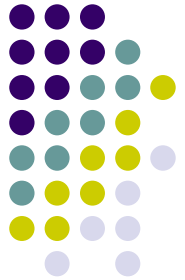
```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 4.0
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 127.93667922166023 "$cbr_(0) start"
…….
```

Step 7: End of simulation



```
# Define node initial position(size) in nam
for {set i 0} {$i < 5 } { incr i} {
     $ns initial_node_position $node_($i) 20
}

# Tell ns/nam the simulation stop time
$ns_ at 10.0 "puts \"NS EXITING...\" ; $ns_
halt"

# Start your simulation
$ns_ run
```

nam Visualization



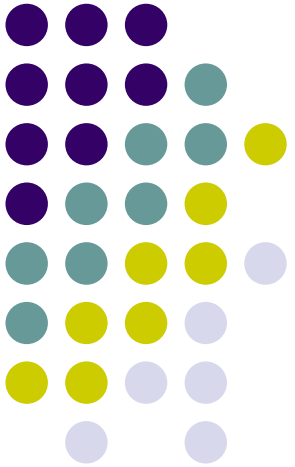
- Use nam to visualize:
  - Mobile node position
  - Mobile node moving direction and speed
  - Power ranges of nodes
- Replace

  $ns namtrace-all $fd

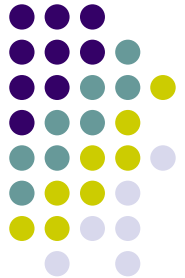
  with

  $ns namtrace-all-wireless $fd

Case study II: Create a new agent in Ns

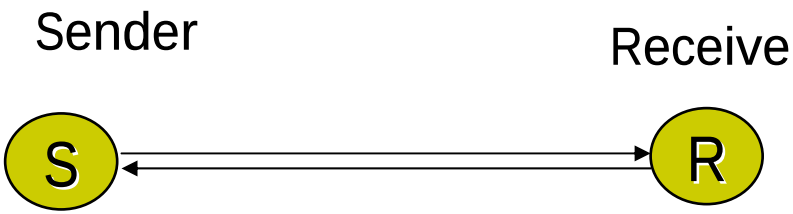


Extending ns in OTcl



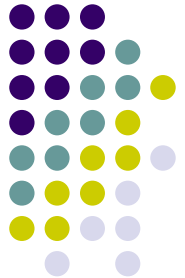
- Add your changes in your tcl scripts
- Or
  - Add new tcl files
  - Change Makefile.in (NS_TCL_LIB), tcl/lib/ns-lib.tcl
  - Recompile

New Agent: Agent/Message



- Sender sends message to the receiver
- Receiver receives the message and sends back a ack to the sender

Step 1: Define sender

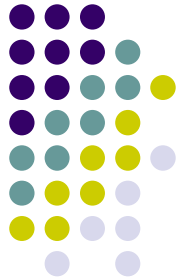


```
Class Sender –superclass Agent/Message

# Message format: “Addr Opcode SeqNo”
Sender instproc send-next {} {
    $self instvar seq_ agent_addr_
    $self send “$agent_addr_ send $seq_”
    incr seq_
    global ns
    $ns at [expr [$ns now]+0.1] "$self
send-next"
}
```

Step 2: Define sender's receive procedure

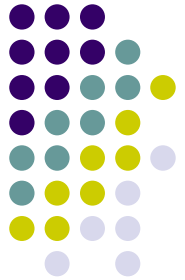
```
Sender instproc recv msg {
    $self instvar agent_addr_
    set saddr [lindex $msg 0]
    set SeqNo [lindex $msg 2]
    puts "Sender gets ack $SeqNo from
$saddr"
}
```

Step 3: Define receiver



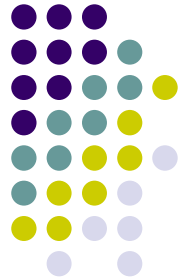
```
Class Receiver –superclass Agent/Message
Receiver instproc recv msg {
    $self instvar agent_addr_
    set saddr [lindex $msg 0]
    set SeqNo [lindex $msg 2]
    puts “Receiver gets seq $SeqNo from
$saddr”
    $self send “$agent_addr_ ack $SeqNo”
}
```

Step 4: Setup simulation scenario- create a scheduler



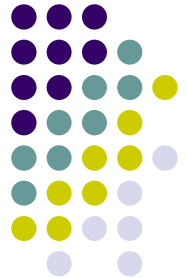
```
# Create scheduler
set ns [new Simulator]

# Turn on nam Tracing
set fd [new out.nam w]
$ns namtrace-all $fd
```

Step 5: Setup simulation scenario- create a network



```
for {set i 0} {$i < 6} {incr i} {
    set n($i) [$ns node]
}
$ns duplex-link $n(0) $n(1) 128kb 50ms DropTail
$ns duplex-link $n(1) $n(4) 10Mb 1ms DropTail
$ns duplex-link $n(1) $n(5) 10Mb 1ms DropTail
$ns duplex-link $n(0) $n(2) 10Mb 1ms DropTail
$ns duplex-link $n(0) $n(3) 10Mb 1ms DropTail

$ns queue-limit $n(0) $n(1) 5
$ns queue-limit $n(1) $n(0) 5
```

Step 6: Setup simulation scenario- select a routing protocol



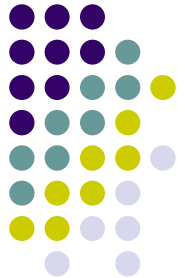
- Routing

```
# Packet loss produced by queueing

# Routing protocol: let’s run distance vector
$ns rtproto DV
```

Step 7: Setup simulation scenario- create traffic

```
set udp0 [new Agent/UDP]
$ns attach-agent $n(2) $udp0
set null0 [new Agent/NULL]
$ns attach-agent $n(4) $null0
$ns connect $udp0 $null0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
 $cbr0 set interval_ 0.5
$cbr0 attach-agent $udp0
$ns at 1.0 “$cbr0 start”
```

Step 8: Put new agent into simulation scenario

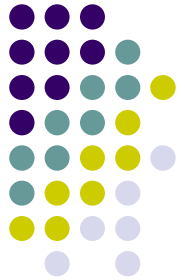


```
set sdr [new Sender]
$sdr set seq_ 0
$sdr set packetSize_ 1000

set rcvr [new Receiver]
$rcvr set packetSize_ 40

$ns attach-agent $n(3) $sdr
$ns attach-agent $n(5) $rcvr
$ns connect $sdr $rcvr
$ns at 1.1 “$sdr send-next”
```

Step 9: End of simulation

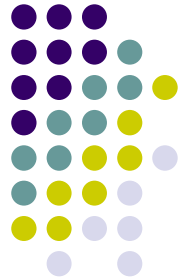
```
$ns at 5.0 finish
proc finish {} {
    global ns fd
    $ns flush-trace
    close $fd
    exit 0
}
$ns run
```

Example output



```
Receiver gets seq 0 from 3
Sender gets ack 0 from 5
Receiver gets seq 1 from 3
Sender gets ack 1 from 5
Receiver gets seq 2 from 3
Sender gets ack 2 from 5
Receiver gets seq 3 from 3
Sender gets ack 3 from 5
Receiver gets seq 4 from 3
Sender gets ack 4 from 5
Receiver gets seq 5 from 3
 …
```

Another way to add your change into ns



- tcl/lib/ns-lib.tcl

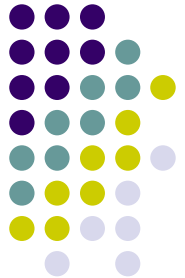
```
Class Simulator
…
source ../mysrc/msg.tcl
```

- Makefile

```
NS_TCL_LIB = \
    tcl/mysrc/msg.tcl \
    …
```

  - Or: change Makefile.in, `make distclean`, then `./configure --enable-debug` , `make depend` and `make`

References



- The VINT project, *The ns manual*
- Padmaparna Haldar and Xuan Chen, *Ns Tutorial 2002,* Network Simulator (ns) Tutorial 2002.
- Padma Haldar, *NS Fundamentals,* Network Simulator (ns) Tutorial 2002.
- Padma Haldar, *Extending Ns,* Network Simulator (ns) Tutorial 2002.
- Padma Haldar, *Wireless world in NS,* Network Simulator (ns) Tutorial 2002.