# INTRODUCTION TO NS-2

Networks Laboratory

# ns2- Network Simulator

- One of the most popular simulator among networking researchers
  - Open source, free
- Discrete event, Packet level simulator
  - Events like 'received an *ack* packet', 'enqueued a data packet'
- Network protocol stack written in C++
- Tcl (Tool Command Language) used for specifying scenarios and events.
  - You can think that Tcl is used to write the high-level programming, while C++ code is doing the actual simulation for speed consideration
- Simulates both wired and wireless networks.

# Simulation Network

- Wired Network
  - Routing: Distance Vector, Link State
  - Transportation: TCP and UDP
  - Queuing disciplines: drop-tail, RED, FQ, SFQ, DRR, RR
  - QoS: IntServ and DiffServ
- Wireless
  - Ad-hoc routing and mobile IP: AODV
  - Sensor-MAC, WiMAX (new)
  - Power control in wireless networks
- Tracing, Visualization, Analysis, Other utilities

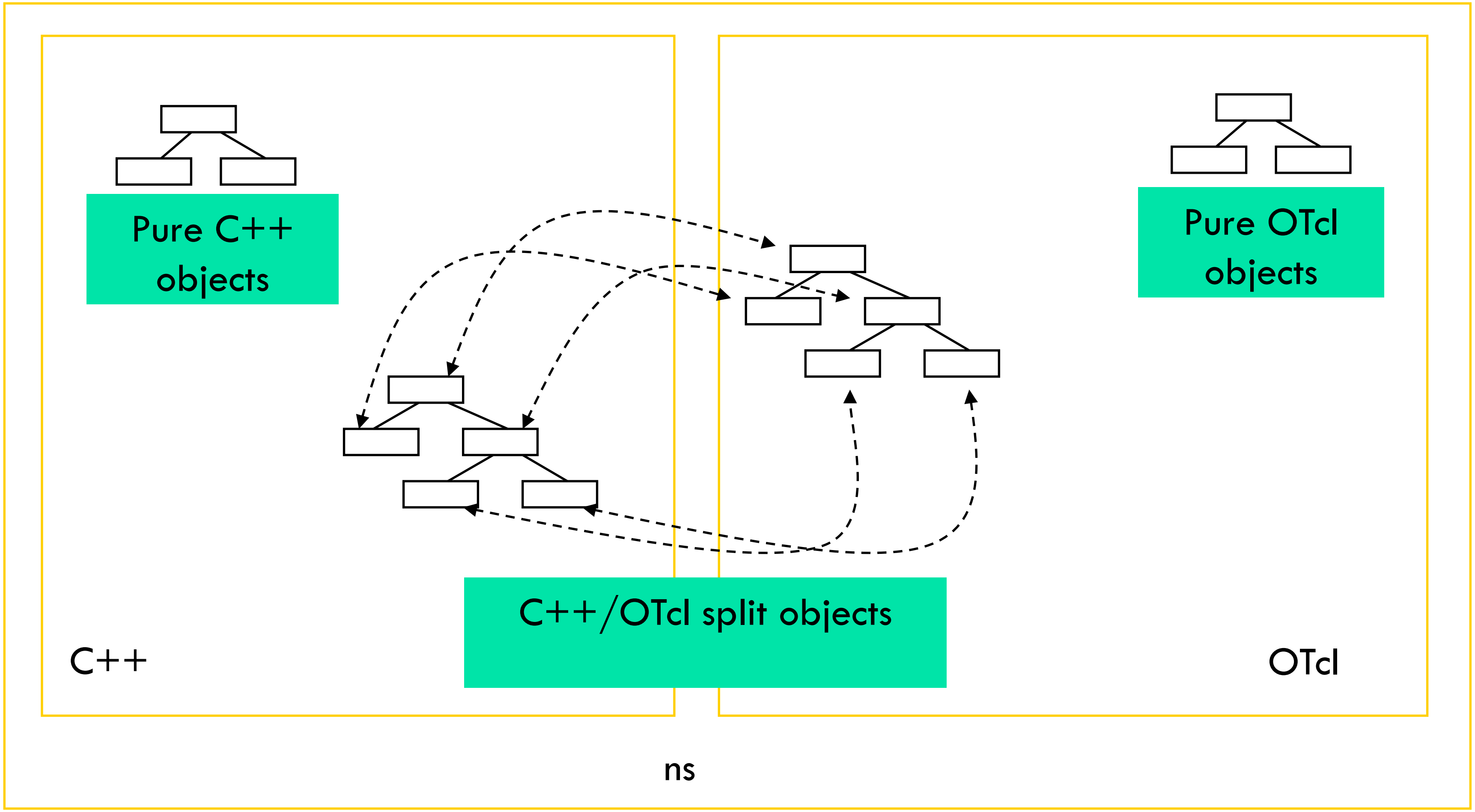
# NS2 Functionalities

- Traffic models and applications
  - Web, FTP, Telnet, CBR, real time traffic
- Transport protocols
  - Unicast: TCP (Reno, New-Reno, Vegas, etc.), UDP
  - Multicast: SRM
- Routing and queuing
  - Wired and ad-hoc routing and directed diffusion
  - Queuing protocols: RED, drop-tail, etc
- Physical media
  - Wired (point-to-point, LANs), wireless (multiple propagation models), error models, satellite

# Platforms & Architecture

- Platforms
  - Most UNIX and UNIX-like systems (FreeBSD, Linux, Sun Solaris)
  - Window 95/98/NT with Cygwin
  - (Emulation only for FreeBSD for now)
- Architecture: Object-Oriented
- C++ for “data”
  - Per packet action
- OTcl for control
  - Periodic or triggered action
- Modularity (+), re-usability(+), scalability(+)
- Speed(-), memory(-)

# OTcl and C++: The Duality



# “Ns” Components

- Ns, the simulator itself
- Nam, the network animator
  - Visualize *ns* (or other) output
  - Nam editor: GUI interface to generate ns scripts
    - Since we only run ns2 in remote Unix server, we will not introduce Nam usage in this class
  - It is not essential to simulation and analysis
- Pre-processing:
  - Traffic and topology generators (use Tcl to write)
- Post-processing:
  - Simple trace analysis, often in Awk, Perl, or Tcl
  - You can also use grep (under linux), or C/java

# Basic TCL

```
□ puts "Hello, world!“
□ puts [expr 1 + 6 + 9]
□ set a 3
   puts $a
□ set myVariable 18
  puts [expr $myVariable + 6 + 9]
□ set myVariable {red green blue}
  puts [lindex $myVariable 2]
  set myVariable "red green blue"
  puts [lindex $myVariable 1]
```

# Simulation with NS2

- Create a New Event Scheduler (simulator env.)
- Turn on Tracing
  - Can use *nam* also
- Topology Creation
  - Create Nodes, Network, Queuing, etc.
  - Setup Routing
  - Send Data
    - Create Transport Connection, Create Traffic, Start Applications
  - Insert Errors
- Analyze the Trace File

# Event Scheduler

- Event
  - Generation of a packet, start/finish of transmission
- Create a New Event Scheduler

  set ns [new Simulator]
- Schedule Events

  $ns at <time> <event>
  - <event>: any legitimate ns/tcl command
  - $ns at 10.0 “finish”
- Start Scheduler

  $ns run

# Tracing and Analyzing

- Packet Tracing
  - On all links
    - $ns trace-all [open cwnd.tr  w]
  - On one specific link
    - $ns trace-queue $n0 $n1 $tr

```
<Event> <time> <from> <to> <pkt> <size> -- <fid> <src> <dst> <seq> <attr>
    + 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
```

- Event Tracing
  - Record “event” in trace file
    - $ns eventtrace-all

```
    E 2.267203 0 4 TCP slow_start 0 210 1
```

# Setup Routing

- 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
- Other Types of Routing Supported
  - Source routing, Hierarchical routing

# Sending Data

- Create UDP Agent  and Attach
  - set udp0 [new Agent/UDP]
  - $ns attach-agent $n0 $udp0
- Create CBR Traffic
  - set src [new Application/Traffic/CBR]
    - set cbr0 [new Application/Traffic/CBR]
    - $cbr0 set packetSize_ 500
    - $cbr0 set interval_ 0.005
    - $cbr0 attachagent $udp0
- Create Traffic Sink and Attach
  - set null [new Agent/Null]
  - $ns attach-agent $n1 $null

# Sending Data

- Create Exponential or Pareto on-off
  - set src [new Application/Traffic/Exponential]
  - set src [new Application/Traffic/Pareto
- Connect two Agents
  - $ns connect $udp0 $null
- Start and Stop of Data
  - $ns at 0.5 “$cbr0 start”
  - $ns at 4.5 “$cbr0 stop”
- Create TCP Agent and Attach
  - set tcp0 [new Agent/TCP]
  - $ns attach-agent $n0 $tcp0

# Sending Data

- Create Traffic Sink and Attach
  - set null0 [new Agent/TCPSink]
  - $ns attach-agent $n1 $null0
- Connect the Agents
  - $ns connect $tcp0 $null0
- Traffic on Top of TCP
  - FTP
    - set ftp [new Application/FTP]
    - $ftp attach-agent $tcp0
  - Telnet
    - set telnet [new Application/Telnet]
    - $telnet attach-agent $tcp0

# 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

# Analyze the Trace File

- Trace files are huge in size
  - Only redirect the parameters you want to measure
  - Traces begin with a single character or abbreviation
  - It indicates the type of trace, followed by a fixed or variable trace format
- Perl scripts are available to analyze trace files
- Refer for the details
  - http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats

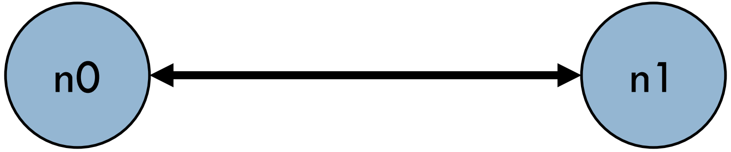
# Script Structure for Wired Scenario

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

# Script Structure: Wireless

```
# parameters and options
set ns [new Simulator]
# [Turn on tracing]
# create MobileNode object (PHY to layer 3
  configured)
# Create topology
# create mobility
# Create Layer 4 and above
#   - UDP/TCP agents
#   - application and/or setup traffic sources
# Post-processing procedures
# Start simulation
```

# Simple two node wired network



Step 1:

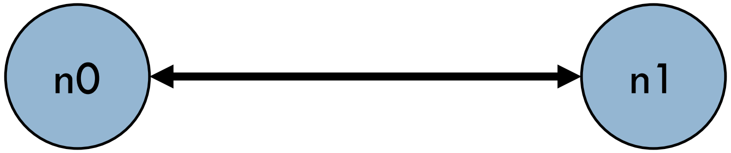
```
#Create a simulator object
# (Create event scheduler)
        set ns [new Simulator]
```

Name of scheduler

Step 2:

```
#Open trace files
set f [open out.tr w]
$ns trace-all $f
```

# Simple two node wired network



Step 3:

```
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]
```

Step 4:

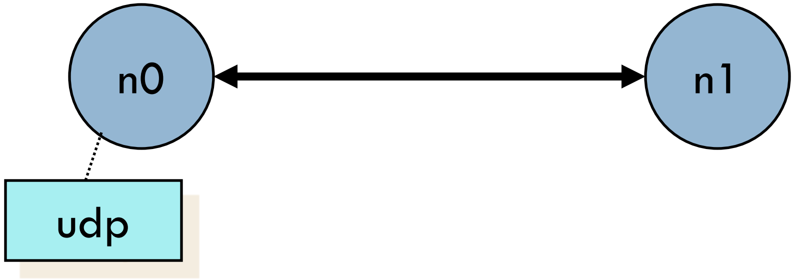
```
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

# Simple two node wired network

```
#Create a simulator object
set ns [new Simulator]
#Open trace files
set f [open out.tr w]
$ns trace-all $f
#Define a 'finish' procedure
proc finish {} {
        global ns f
        $ns flush-trace
        close $f
        exit 0
}
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

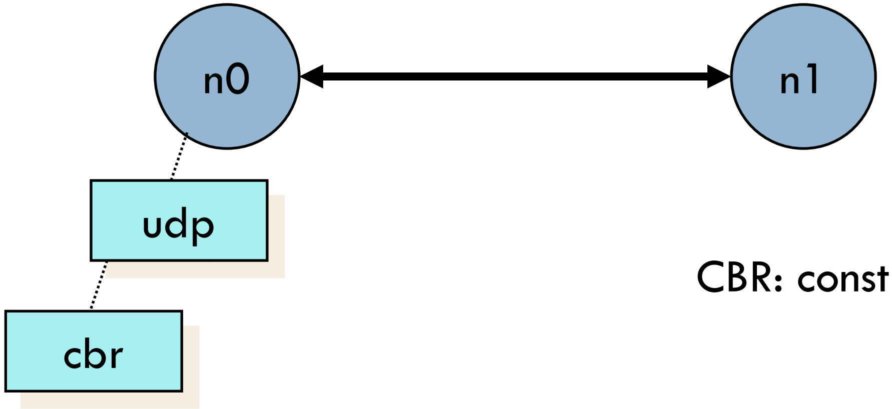
But we have no traffic!

# Adding traffic to the link



```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
```

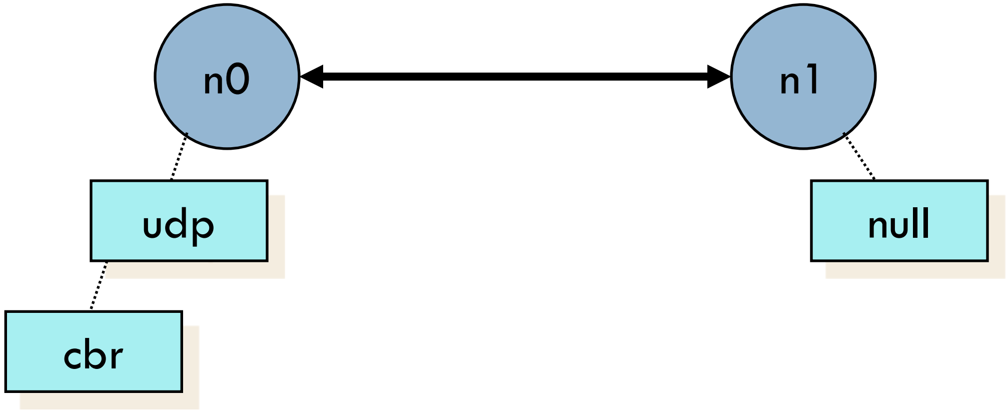
# Adding traffic to the link



CBR: constant bit rate

```
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

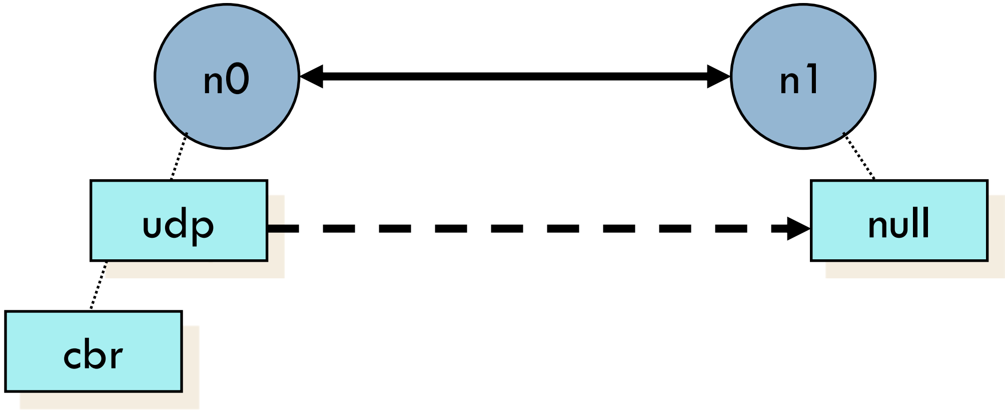
# Adding traffic to the link



#Create a Null agent (a traffic sink) and attach it to node n1

```
set null0 [new Agent/Null]
$ns attach-agent $n1  $null0
```

# Adding traffic to the link



```
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop”
$ns at 5.0 "finish"
$ns run
```

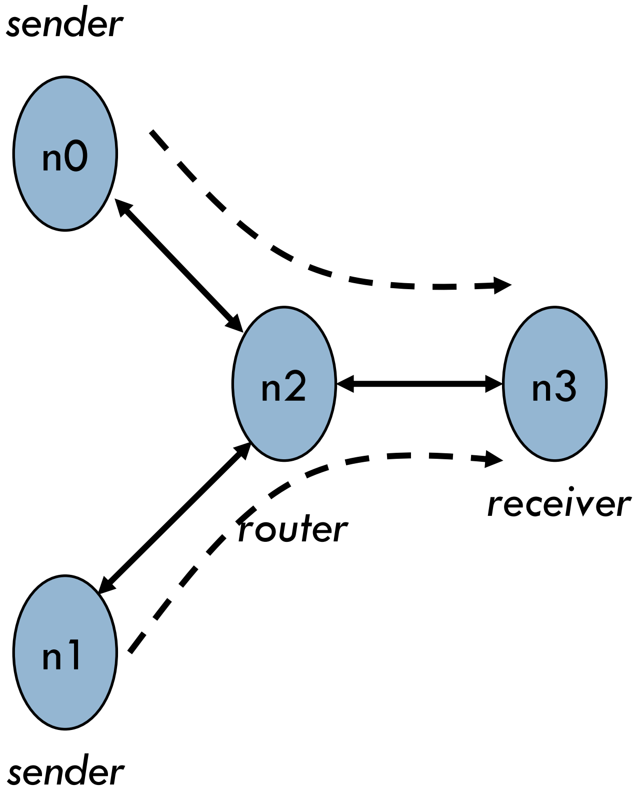
# Record Simulation Trace

- Packet tracing:
  - On all links: $ns trace-all [open out.tr w]
  - On one specific link: $ns trace-queue $n0 $n1$tr

```
<Event> <time> <from> <to> <pkt> <size> -- <fid> <src> <dst> <seq> <attr>
   + 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
```

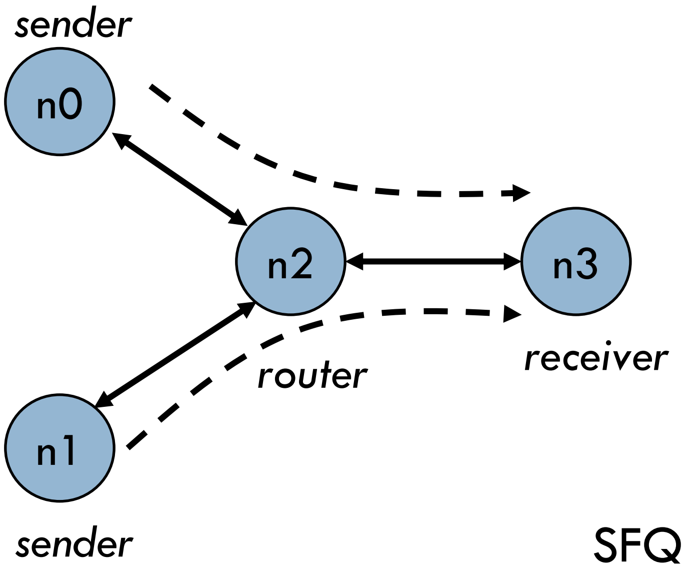
  - Event "+": enqueue, "-": dequeue; "r": received

# Simulate a simple topology – UDP Traffic



```
#Create a simulator object
set ns [new Simulator]
#Open trace files
set f [open out.tr w]
$ns trace-all $f
#Define a 'finish' procedure
proc finish {} {
      global ns
      $ns flush-trace
      exit 0
}
#Create four nodes
set n0 [$ns node]
set n1  [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

# Simulate a simple topology – UDP Traffic

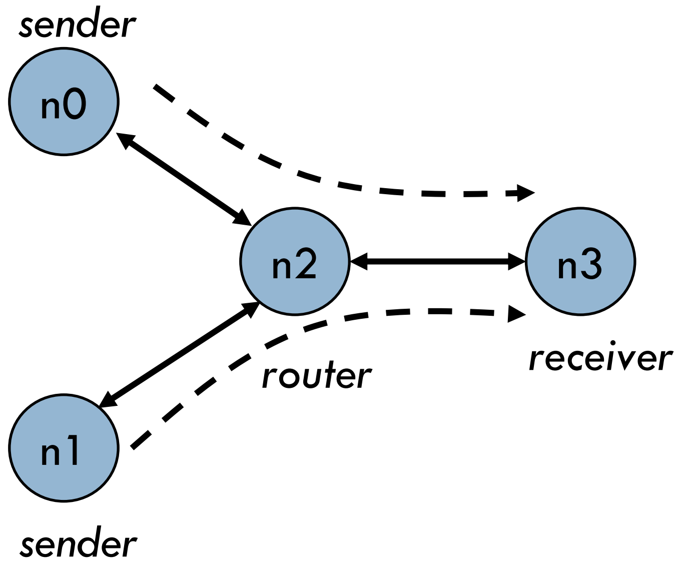


SFQ: Stochastic Fair queuing

#Create links between the nodes

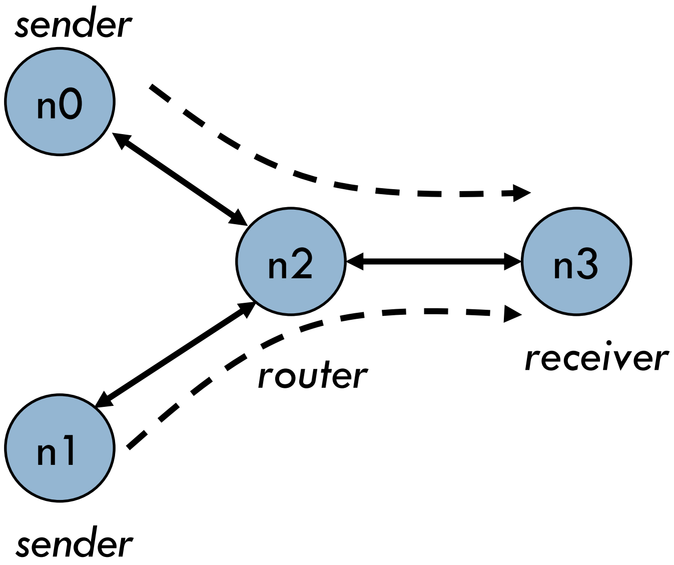
```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ
```

# Simulate a simple topology – UDP Traffic



```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set class_ 1  # fid in trace file
$ns attach-agent $n0 $udp0
```

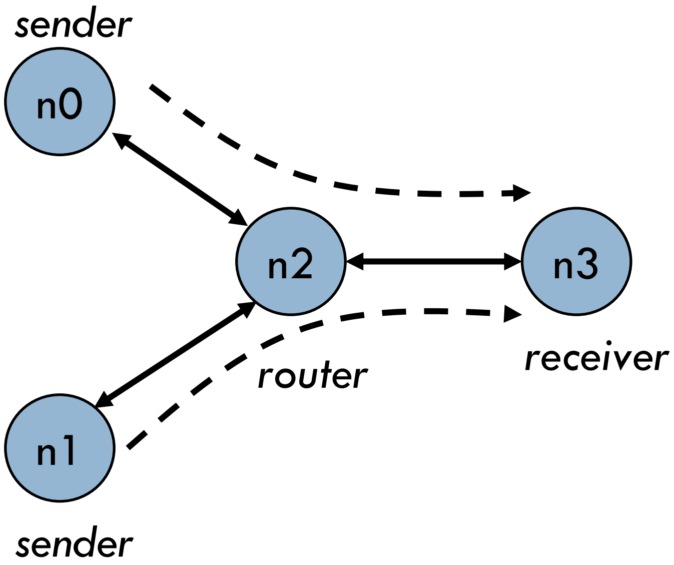
# Simulate a simple topology – UDP Traffic



# Create a CBR traffic source and attach it to udp0

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

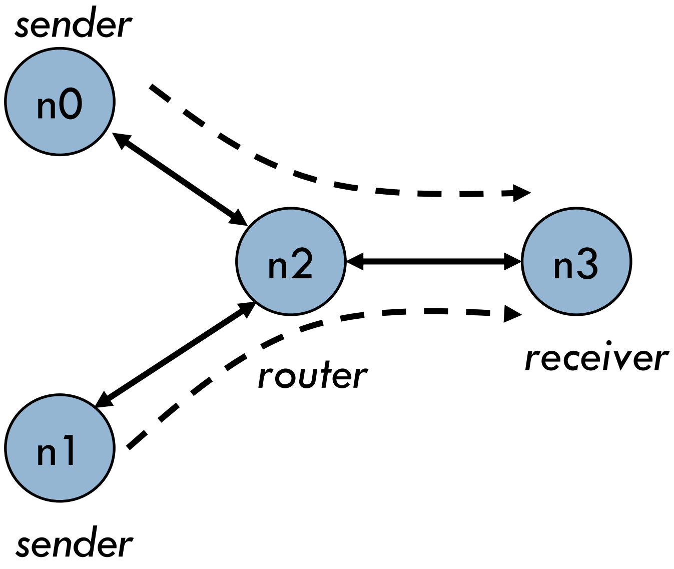
# Simulate a simple topology – UDP Traffic



#Create a UDP agent and attach it to node n1

```
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1
```

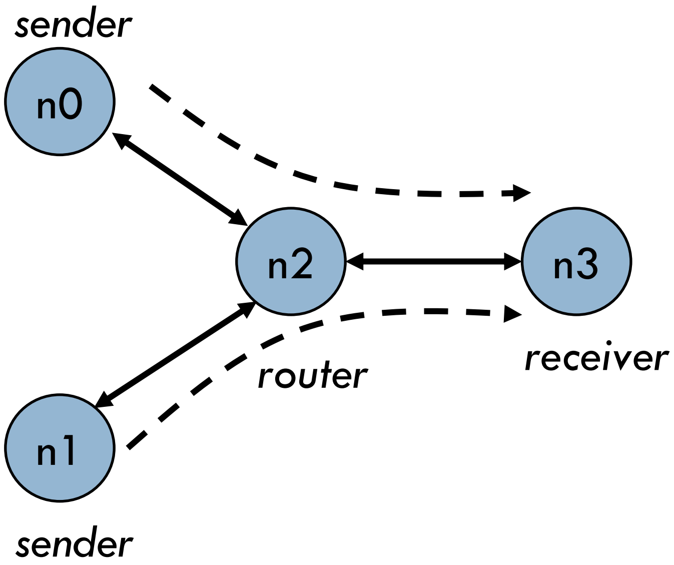
# Simulate a simple topology – UDP Traffic



# Create a CBR traffic source and attach it to udp1

```
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
```

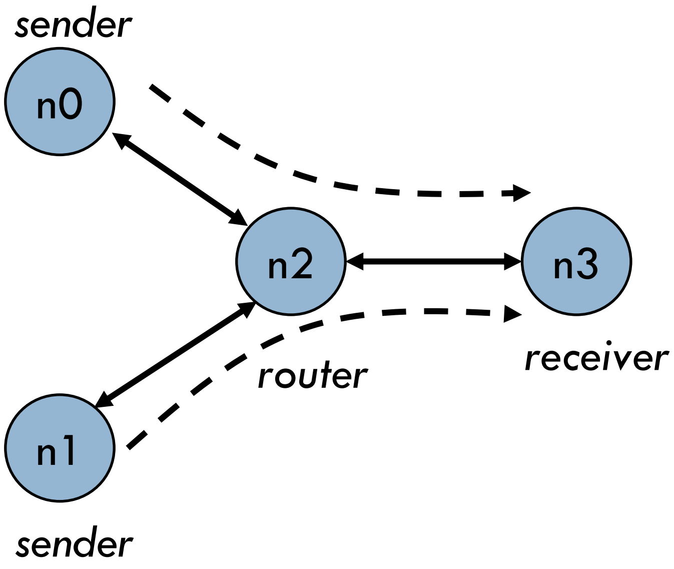
# Simulate a simple topology – UDP Traffic



#Create a Null agent (a traffic sink) and attach it to node n3

```
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

# Simulate a simple topology – UDP Traffic



#Connect the traffic sources with the traffic sink

```
$ns connect $udp0 $null0
$ns connect $udp1 $null0
```

# Simulate a simple topology – UDP Traffic

```
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

# Trace Analysis

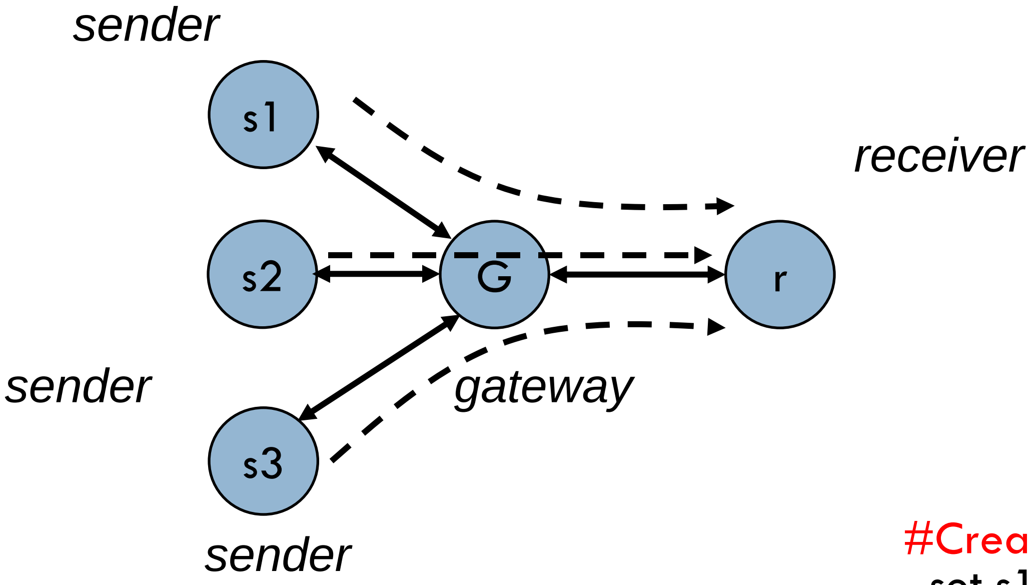
| event | time | from node | to node | pkt type | pkt size | flags | fid | src addr | dst addr | seq num | pkt id |
|---|---|---|---|---|---|---|---|---|---|---|---|

```
r : receive (at to_node)
+ : enqueue (at queue)
- : dequeue (at queue)
d : drop    (at queue)
```

```
src_addr : node.port (3.0)
dst_addr : node.port (0.0)
```

```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

# TCP Traffic



- 0, 1, 2 are senders
- 3 is a Gateway
- 4 receiver

```
……
#Create five nodes
set s1 [$ns node]
set s2 [$ns node]
set s3 [$ns node]
set G [$ns node]
set r [$ns node]
#Create links between the nodes
……
```

# TCP Traffic

- #Create a TCP agent and attach it to node s1

  ```
  set tcp1 [new Agent/TCP/Reno]
  $ns attach-agent $s1 $tcp1
  $tcp1 set window_ 8
  $tcp1 set fid_ 1
  ```

- "window_" is the upperbound of congestion window in a TCP. It is 20 by default.

# TCP Traffic

- #Create a TCP agent and attach it to node s2

```
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $s2 $tcp2
$tcp2 set window_ 8
$tcp2 set fid_ 2
```

- #Create a TCP agent and attach it to node s3

```
set tcp3 [new Agent/TCP/Reno]
$ns attach-agent $s3 $tcp3
$tcp3 set window_ 4
$tcp3 set fid_ 3
```

# TCP Traffic

- #Create TCP sink agents and attach them to node r

```
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
set sink3 [new Agent/TCPSink]

$ns attach-agent $r $sink1
$ns attach-agent $r $sink2
$ns attach-agent $r $sink3
```

For more TCP agents, see:
http://www.isi.edu/nsnam/ns/doc/node387.html

# TCP Traffic

- #Connect the traffic sources with the traffic sinks

```
$ns connect $tcp1  $sink1
$ns connect $tcp2 $sink2
$ns connect $tcp3 $sink3
```

- You cannot connect two TCP sources to the same TCP sink
  - You can do that for UDP traffic

# TCP Traffic

- #Create FTP applications and attach them to agents

```
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
```

For more Applications, see:
http://www.isi.edu/nsnam/ns/doc/node498.html

# TCP Traffic

```
#Define a 'finish' procedure
proc finish {} {
        global ns
        $ns flush-trace
        exit 0
}

$ns at 0.1 "$ftp1 start"
$ns at 0.1 "$ftp2 start"
$ns at 0.1 "$ftp3 start"
$ns at 5.0 "$ftp1 stop"
$ns at 5.0 "$ftp2 stop"
$ns at 5.0 "$ftp3 stop"
$ns at 5.25 "finish"
$ns run
```

# Trace Analysis

```
czou@eustis:~/ns2$ grep '^r' out.tr > 3TCP-receive-only.tr

r 0.1596 0 3 tcp 1040 ------- 1 0.0 4.0 1 6
r 0.15992 1 3 tcp 1040 ------- 2 1.0 4.1 1 8
r 0.16024 2 3 tcp 1040 ------- 3 2.0 4.2 1 10
r 0.16792 0 3 tcp 1040 ------- 1 0.0 4.0 2 7
r 0.16824 1 3 tcp 1040 ------- 2 1.0 4.1 2 9
r 0.16856 2 3 tcp 1040 ------- 3 2.0 4.2 2 11
r 0.17792 3 4 tcp 1040 ------- 1 0.0 4.0 1 6
r 0.18624 3 4 tcp 1040 ------- 2 1.0 4.1 1 8
r 0.18824 4 3 ack 40 ------- 1 4.0 0.0 1 12
r 0.19456 3 4 tcp 1040 ------- 3 2.0 4.2 1 10
r 0.19656 4 3 ack 40 ------- 2 4.1 1.0 1 13
r 0.19856 3 0 ack 40 ------- 1 4.0 0.0 1 12
r 0.20288 3 4 tcp 1040 ------- 1 0.0 4.0 2 7
r 0.20488 4 3 ack 40 ------- 3 4.2 2.0 1 14
r 0.20688 3 1 ack 40 ------- 2 4.1 1.0 1 13
r 0.2112 3 4 tcp 1040 ------- 2 1.0 4.1 2 9
r 0.2132 4 3 ack 40 ------- 1 4.0 0.0 2 17
r 0.2152 3 2 ack 40 ------- 3 4.2 2.0 1 14
```