# Network Simulator 2 (NS2)

Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel
Dipl.-Ing. Ali Diab

Integrated Hard- and Software Systems

# Outline

- History of NS2
- Getting Started
- NS2 Basics
- Example
- Mobility Management in ns2
- References

# History of NS2

# History of ns2

- Start 1989 as a variant of REAL (network simulator for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks)

- After 1995, Funding from DARPA through many projects (VINT project at LBL, Xerox PARC, UCB, USC/ISI. SAMAN and NSF with CONSER)

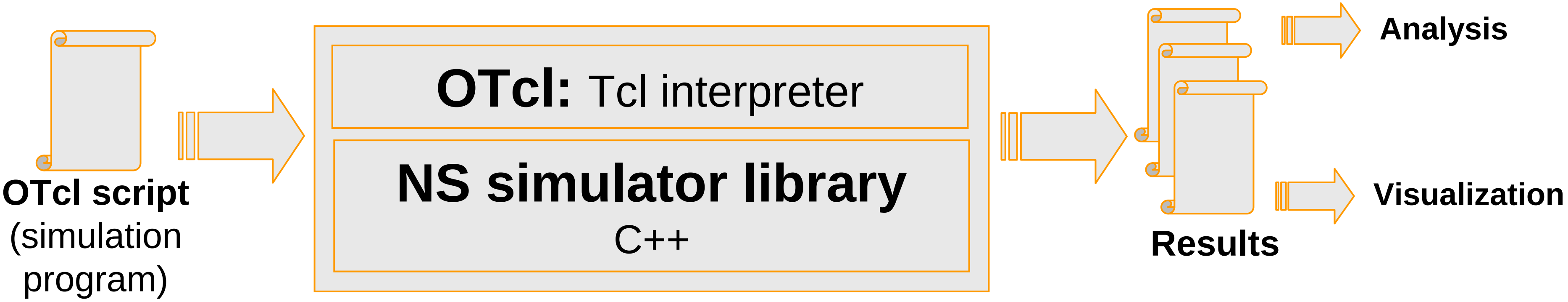
- NS2 includes many Contributions, e.g. from other researchers, wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems

# Getting Started

# NS2 Properties

- A discrete event simulator (timing of events is maintained in a scheduler)

- Two languages, why?
  - **System language**: C++, fast and robust language, widely used, compiled, typed to manage complexity, high efficiency.
  - **Scripting language**: OTCL, high level programming, fast changeable applications, Interpreted, less efficient.



**OTcl script** (simulation program)

**OTcl:** Tcl interpreter

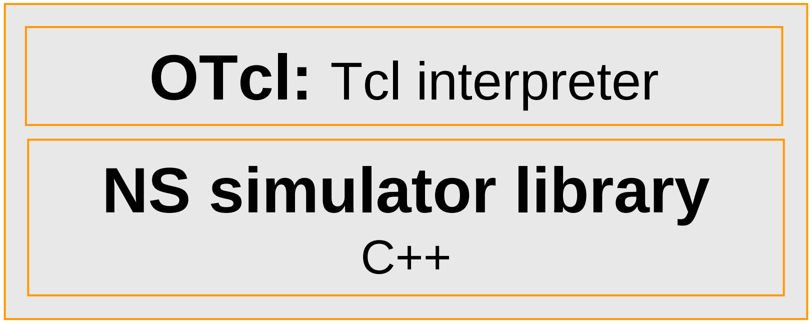
**NS simulator library** C++

**Results**

**Analysis**

**Visualization**

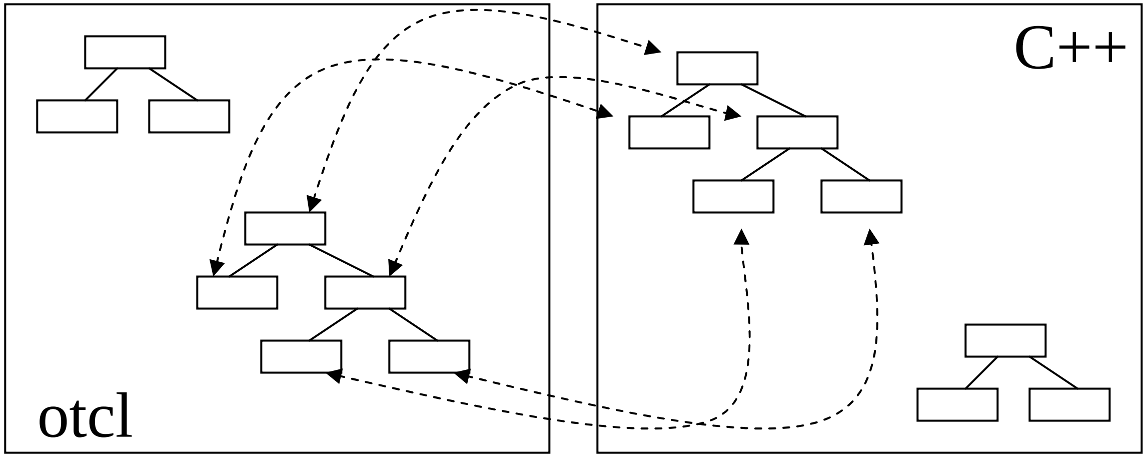
# NS2 Properties



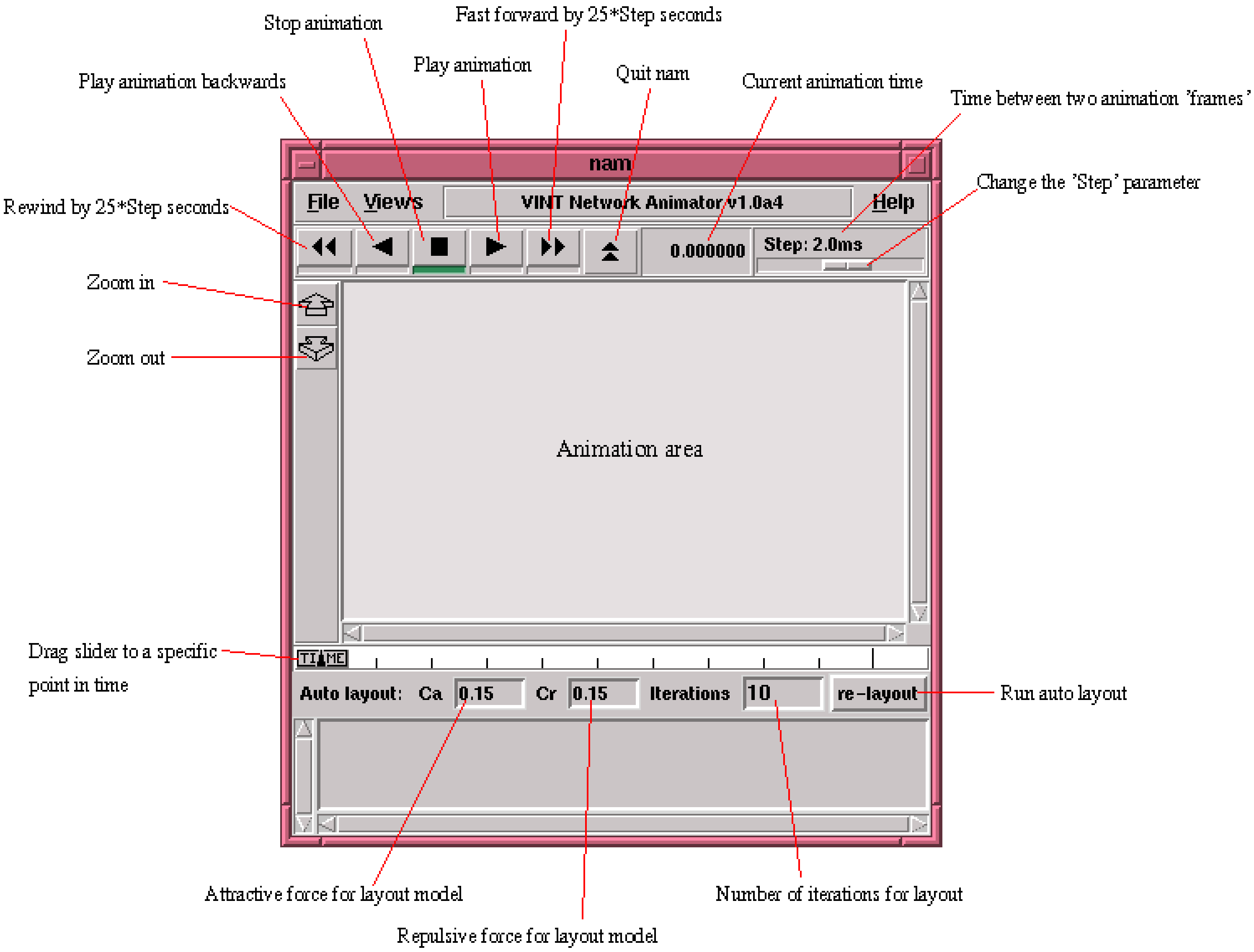
**OTcl:** Tcl interpreter

**NS simulator library**

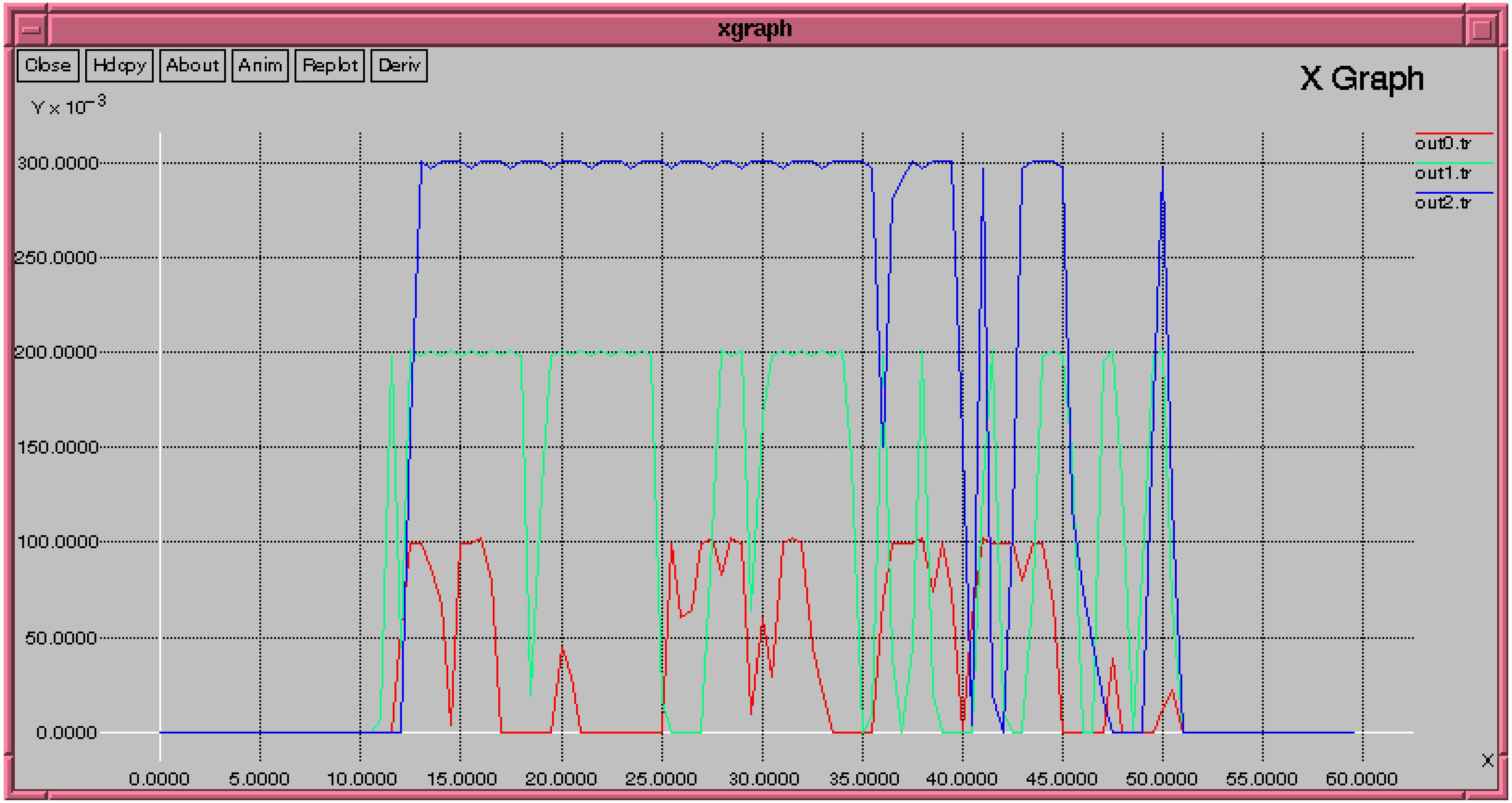
C++



# NS2 Visualization Tool (Nam)


Stop animation
Fast forward by 25*Step seconds
Play animation
Quit nam
Play animation backwards
Current animation time
Time between two animation 'frames'
Change the 'Step' parameter
Rewind by 25*Step seconds
Zoom in
Zoom out
Animation area
Drag slider to a specific point in time
Run auto layout
Attractive force for layout model
Repulsive force for layout model
Number of iterations for layout
nam
File
Views
VINT Network Animator v1.0a4
Help
0.000000
Step: 2.0ms
TIME
Auto layout:
Ca
0.15
Cr
0.15
Iterations
10
re-layout

# NS2 Analysis Tool (Xgraph)



# Tcl Overview

- Set a 0 → declare a variable named „a“ with a value „0“
- Set b $a → declare a variable named „b“ with a value equal to the value of the variable „a“
- Set x [expr $a + $b] → declare a variable named „x“ with a value equal to the sum of „a“ and „b“
- # → write a comment
- Set file1 [open out1.tr w] → define a file named „file1“ and assign it to „out1.tr“
- Puts “text” → print out the word “text”

# Tcl Overview

- Puts “The value of x is $x” → print out “The value of x is 0”

- exec xgraph data.tr & → execute the program “xgraph”, which takes the file “data.tr” as an input

- If { expression }

  { some commands } else { some commands }

- For { set i 0 } { $i < 5 } { incr i }

  { some commands }

- Proc example {x1 x1 ….}

  { some commands …

  return $something}

# NS2 Basics

# Creation of Event Scheduler

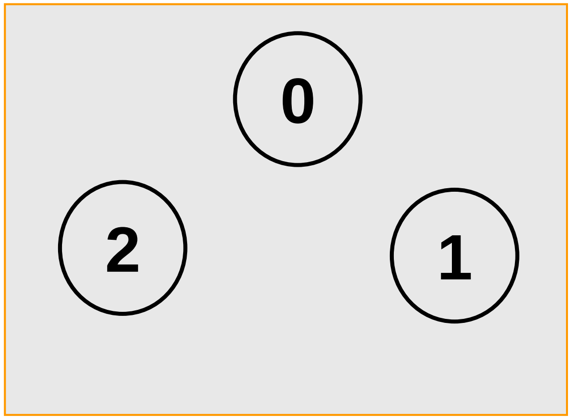
- Create a scheduler
  - set ns [new Simulator]

- Schedule an event
  - $ns at <time> <event>
  - Example: $ns at 10.0 “record_data”

- Start the scheduler
  - $ns run

# Creation of Network Topology

- Create Nodes
  - set n_0 [$ns node]
  - set n_1 [$ns node]
  - set n_2 [$ns node]



- Create Nodes (using a loop)
  - For { set i 0 } { $i < 3 } { incr i }

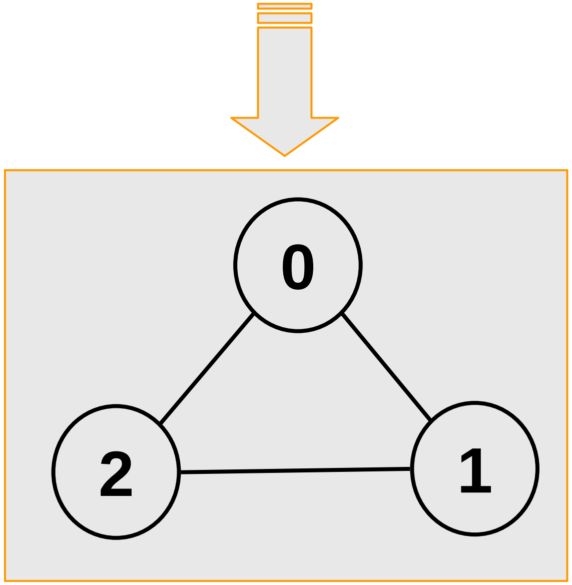
    ```
    {
    set n_$i [$ns node]
    }
    ```

# Creation of Network Topology

- Create links between the nodes
  - $ns <link type> $n_0 $n_1 <bandwidth> <delay> <queue type>
    - <link type>: duplex-link, simplex-link
    - <bandwidth>: in Mb
    - <delay>: in ms
    - <queue type>: DropTail, RED, CBQ, FQ, SFQ, DRR

# Creation of Network Topology

- Create links between the nodes of our example
  - $ns simplex-link $n_0 $n_1 1Mb 5ms DropTail
  - $ns simplex-link $n_0 $n_2 1Mb 5ms DropTail
  - $ns duplex-link $n_1 $n_2 10Mb 25ms DropTail



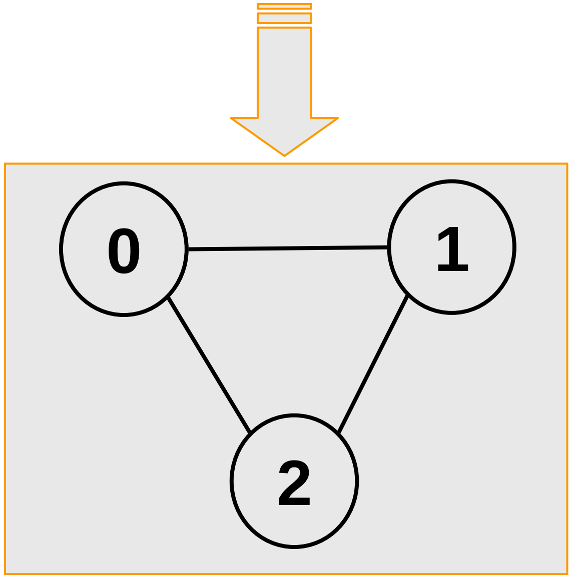
# Creation of Network Topology

- Define the properties of the links between the nodes
  - $ns duplex-link-op $n_0 $n_1 <attribute> <value>

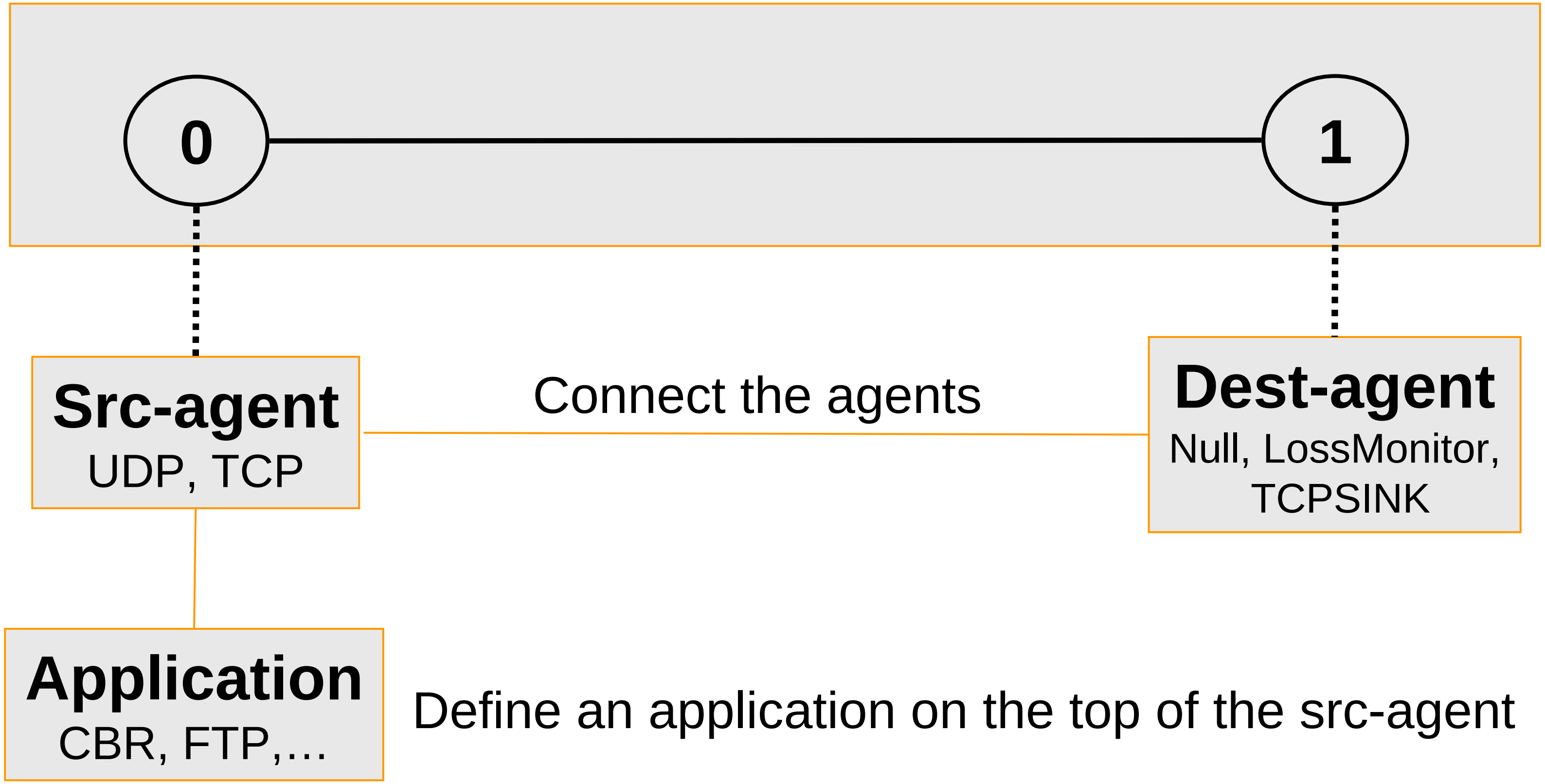
  - <attribute>: orient, color, queuePos, label
    - orient: the orientation of a link (up, down, right, left, right-up, right-down, left-up, left-down)
    - color: the color of the link (black, green, red,…etc)
    - queuePos: angle of the queue line with horizontal (default 0.5)
    - Label: label of the link

# Creation of Network Topology

- Define the orientation of the links between the nodes of our example
    - $ns duplex-link-op $n_0 $n_1 orient right
    - $ns duplex-link-op $n_0 $n_2 orient right-down
    - $ns duplex-link-op $n_1 $n_2 orient left-down



# Connection and Traffic



**Src-agent**
UDP, TCP

Connect the agents

**Dest-agent**
Null, LossMonitor, TCPSINK

**Application**
CBR, FTP,…

Define an application on the top of the src-agent

# UDP agent

- set Src-agent [new Agent/UDP]

- $ns attach-agent $n_0 $Src-agent

- set Dest-agent [new Agent/NULL]

- $ns attach-agent $n_1 $Dest-agent

- $ns connect $Src-agent $Dest-agent

# TCP agent

- set Src-agent [new Agent/TCP]
- $ns attach-agent $n_0 $Src-agent
- set Dest-agent [new Agent/TCPSink]
- $ns attach-agent $n_1 $Dest-agent
- $ns connect $Src-agent $Dest-agent

# Creation of Traffic

- FTP
  - set src [new Application/FTP]
  - $src attach-agent $Src-agent

- Telnet
  - set src [new Application/Telnet]
  - $src attach-agent $Src-agent

# Creation of Traffic

- CBR
  - set src [new Application/Traffic/CBR]
  - $src attach-agent $Src-agent

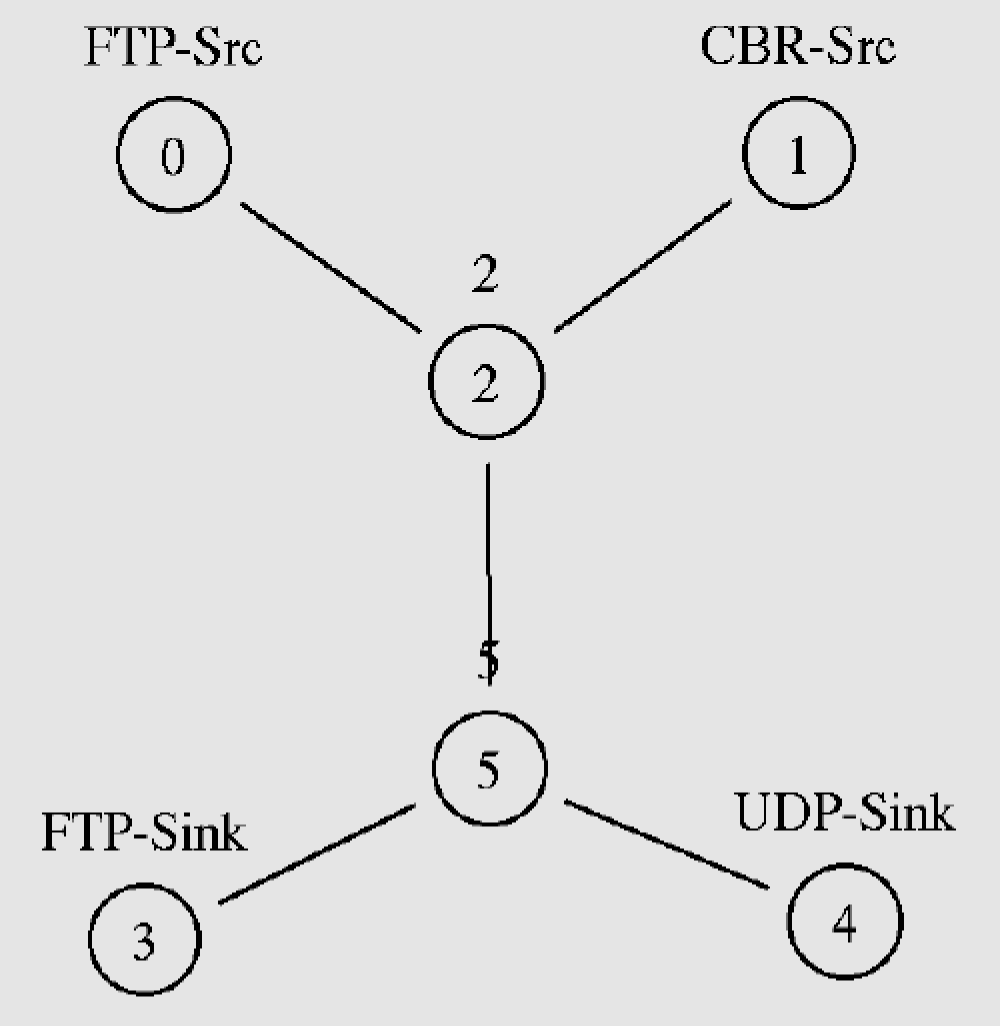
- Exponential or Pareto on-off
  - set src [new Application/Traffic/Exponential]
  - set src [new Application/Traffic/Pareto]
  - $src attach-agent $Src-agent

# Parameterize, Start and Stop a Traffic Source

- CBR
  - set src [new Application/Traffic/CBR]
  - $src attach-agent $Src-agent
  - $src set interval_ 40ms
  - $src set packetSize_ 500
  - $ns at 10.0 “$src start”
  - $ns at 100.0 “$src stop”

# Example

# Example



# Example

```
set ns [new Simulator]


# To be able to use nam, we should Create a nam trace datafile.
set namfile [open results/versuch1.nam w]
$ns namtrace-all $namfile

# After that, we should create the nodes
For { set i 0 } { $i < 6 } { incr i }
{ set node($i) [$ns node] }

$ns run
```

# Example

```
set ns [new Simulator]


# After that, we should connect the nodes with each other
$ns duplex-link $node(0) $node(2) 1.0Mb 20.0ms DropTail
$ns duplex-link $node(1) $node(2) 1.0Mb 20.0ms DropTail
$ns duplex-link $node(2) $node(5) 1.0Mb 20.0ms DropTail
$ns simplex-link $node(5) $node(2) 0.125Mb 20.0ms DropTail
$ns duplex-link $node(3) $node(5) 1.0Mb 20.0ms DropTail
$ns duplex-link $node(4) $node(5) 1.0Mb 20.0ms DropTail


$ns run
```

# Example

```
set ns [new Simulator]



# After that, we have to create the agents
set agent(0) [new Agent/UDP]
$ns attach-agent $node(1) $agent(0)
$agent(0) set fid_ 6
$ns color 6 "red„
set sink(0) [new Agent/Null]
$ns attach-agent $node(4) $sink(0)
$ns connect $agent(0) $sink(0)
$ns run
```

# Example

```
set ns [new Simulator]

# After that, we have to create traffic source and add it to the agent
set traffic_source(0) [new Application/Traffic/CBR]
$traffic_source(0) set interval_ 0.001950
$traffic_source(0) set paketSize_ 230
$traffic_source(0) attach-agent $agent(0)

$ns run
```

# Example

```
set ns [new Simulator]

# Now, we have to schedule starting and stopping the  traffic source
$ns at 3.0 "$traffic_source(0) start„
$ns at 100.0 "$traffic_source(0) stop„

$ns run
```

# Example

```
set ns [new Simulator]
................................................................................

# Now, we have to start the finish procedure
proc finish {} {
global ns namfile
$ns flush-trace
close $namfile
exec nam results/versuch1.nam &
exit 0
}
$ns run
```

# Example

```
set ns [new Simulator]
.................................................................
.................................................................
# After that, we have to schedule the stop procedure
$ns at 110.000000 "finish"

$ns run
```

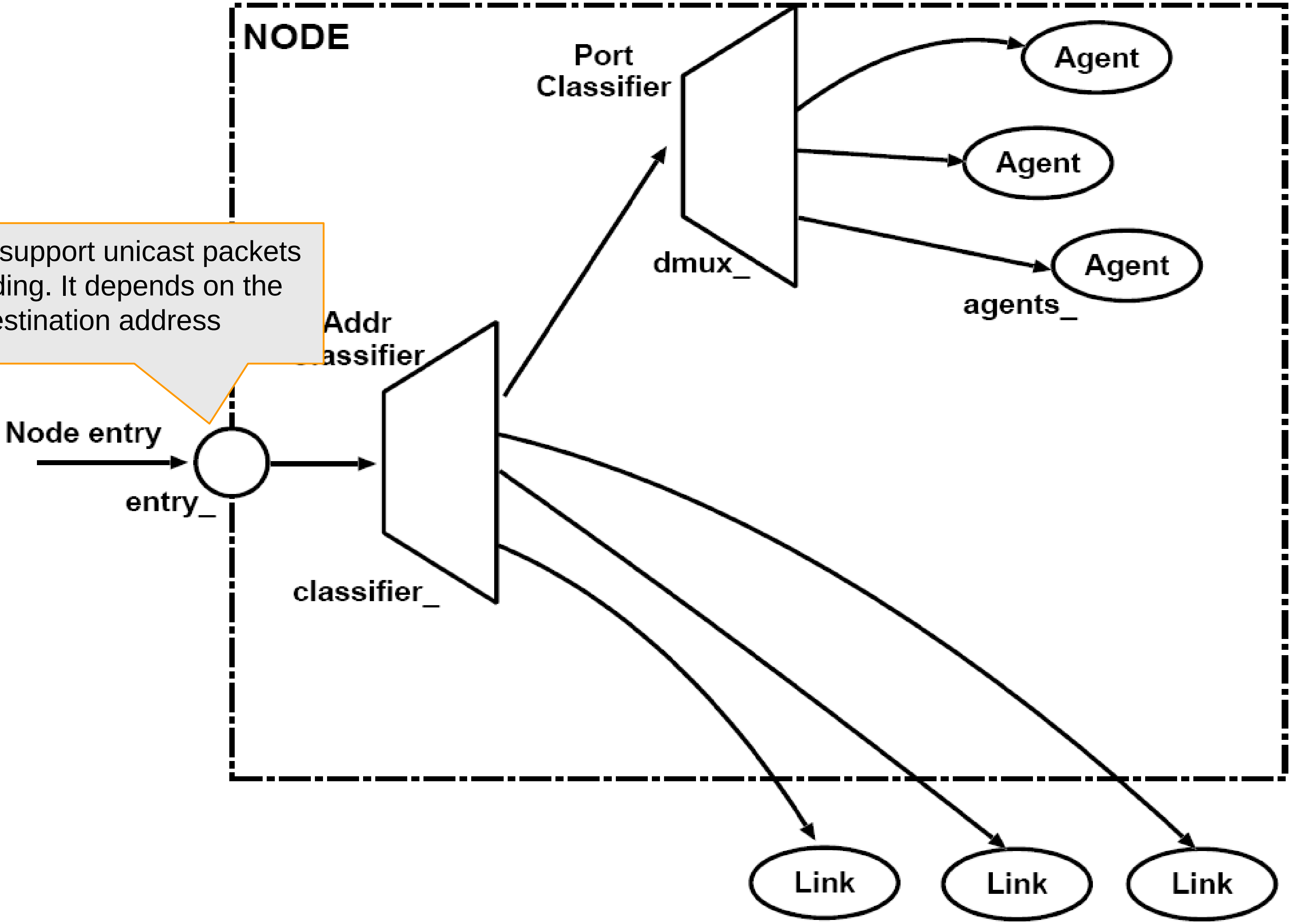
# Example

- After that the file should be saved “filename.tcl”

- The executing of the example is through writing: “ns filename.tcl” in linux commands window (Console)

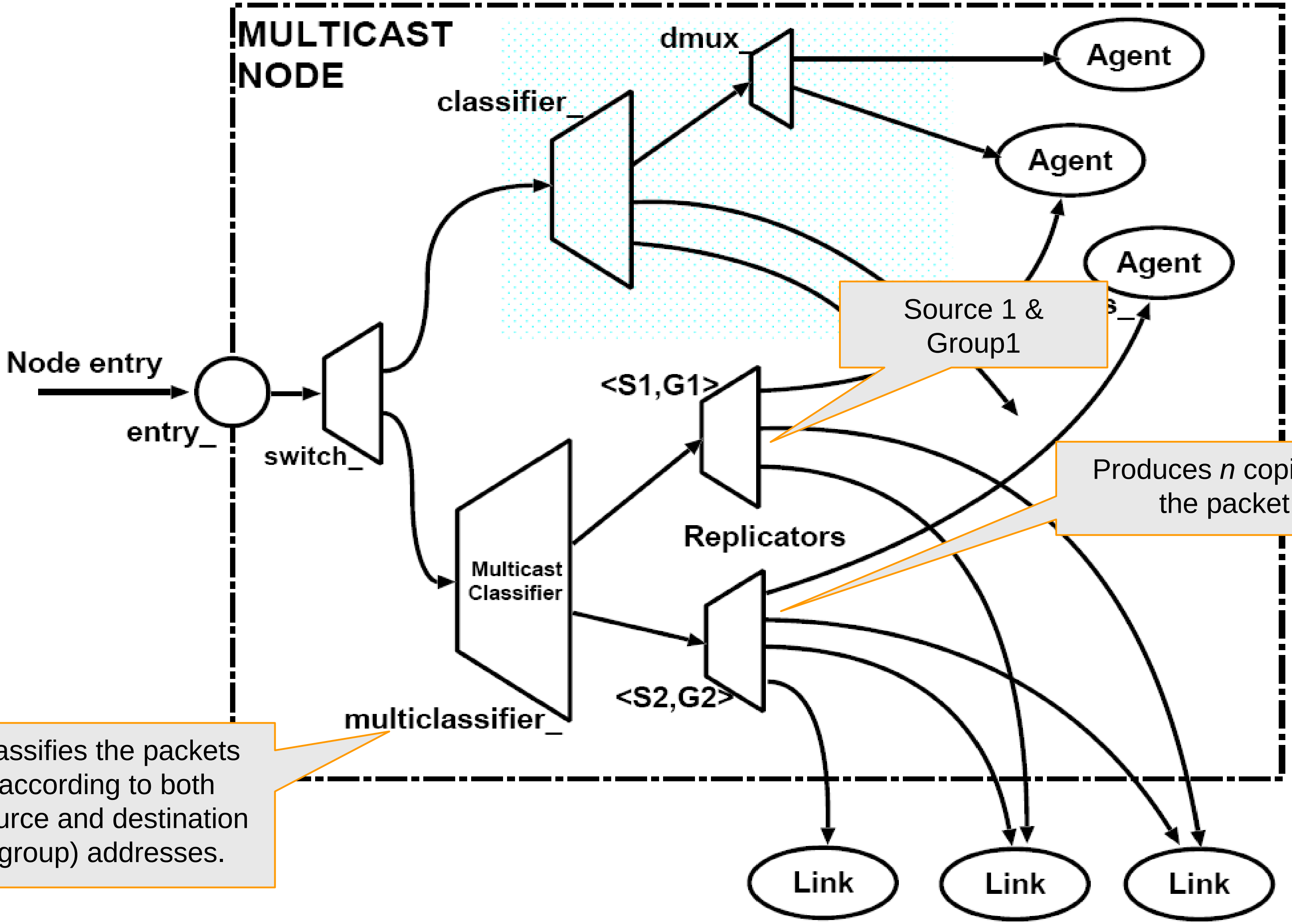
# Mobility Management in ns2

# Nodes in NS2 – Normal Node



Used to support unicast packets forwarding. It depends on the destination address

# Nodes in NS2 – Multicast Node



Source 1 & Group1

Produces *n* copies of the packet

classifies the packets according to both source and destination (group) addresses.

# Nodes in NS2 – Mobile Node

- Extended structure than other normal nodes
- There is no links between nodes
- They can move inside a certain topology
- They should be configured by many parameters to define the physical, MAC, routing, etc.
- Routing could be wireless / Wireless-wired (HA & FAs)

# Mobile Node in NS2 - Configuring a Mobile Node

## The following parameters should be defined

| | |
|---|---|
| adhocRouting | : Routing protocol → AODV, DSDV, TORA, DSR,.. |
| llType | : The link layer → LL, LL/Sat |
| macType | : The MAC layer → MAC/802_11, MAC/Sat, MAC/Sat/UnslottedAloha, MAC/Tdma |
| ifqType | : Type of Queue → Queue/DropTail, Queue/DropTail/priQueue |
| ifqLen | : Length of the Queue |
| antType | : Type of Antenna → Antenna/OmniAntenna |
| propInstance | : Wireless propagation model → Propagation/TwoRayGround, Propagation/Shadowing |

# Mobile Node in NS2 - Configuring a Mobile Node

phyType : Type of physical interfaces → Phy/WirelessPhy, Phy/Sat

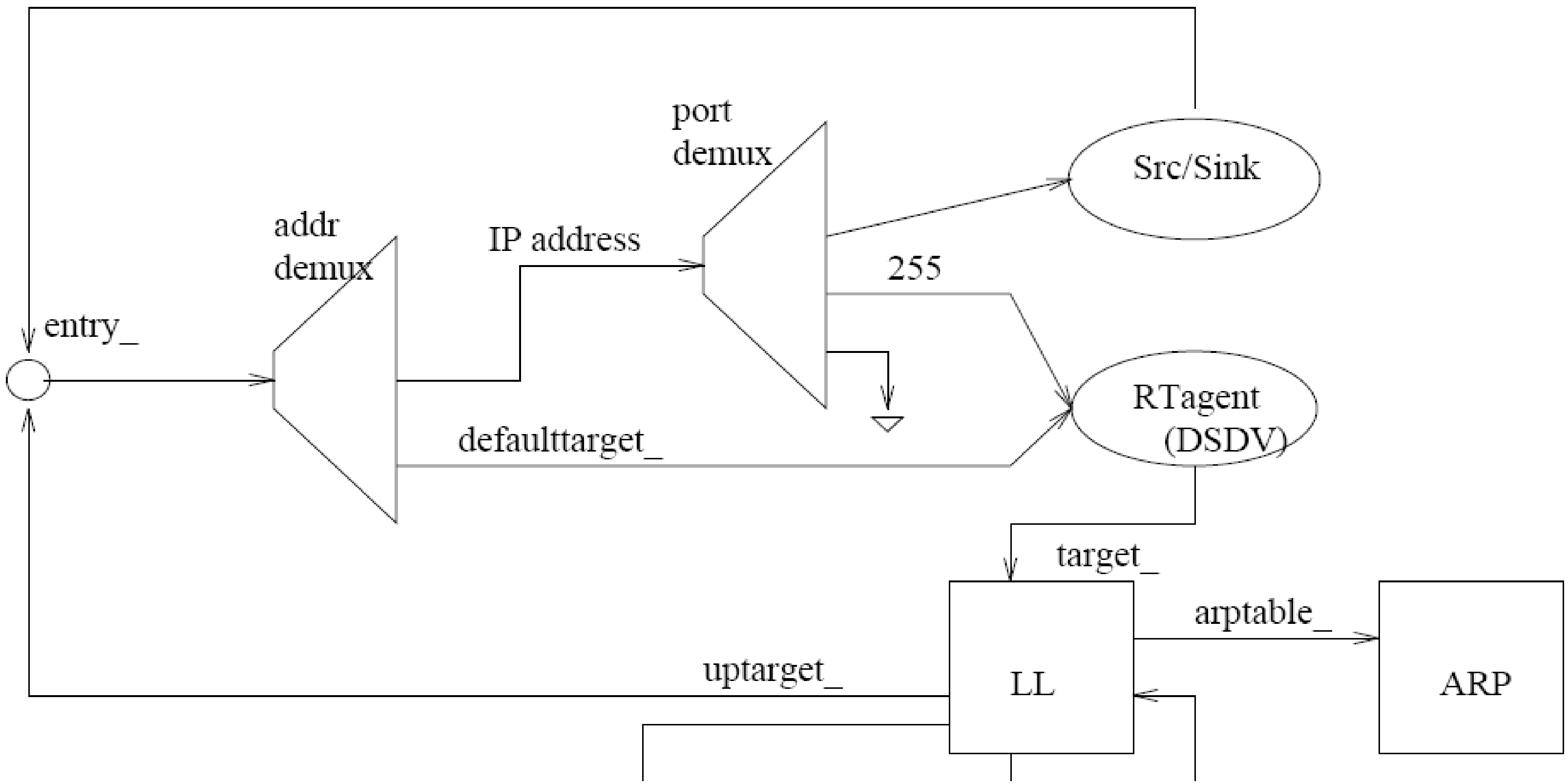
Channel : Type of wireless channel → Channel/WirelessChannel, Channel/Sat

topoInstance : The used topology

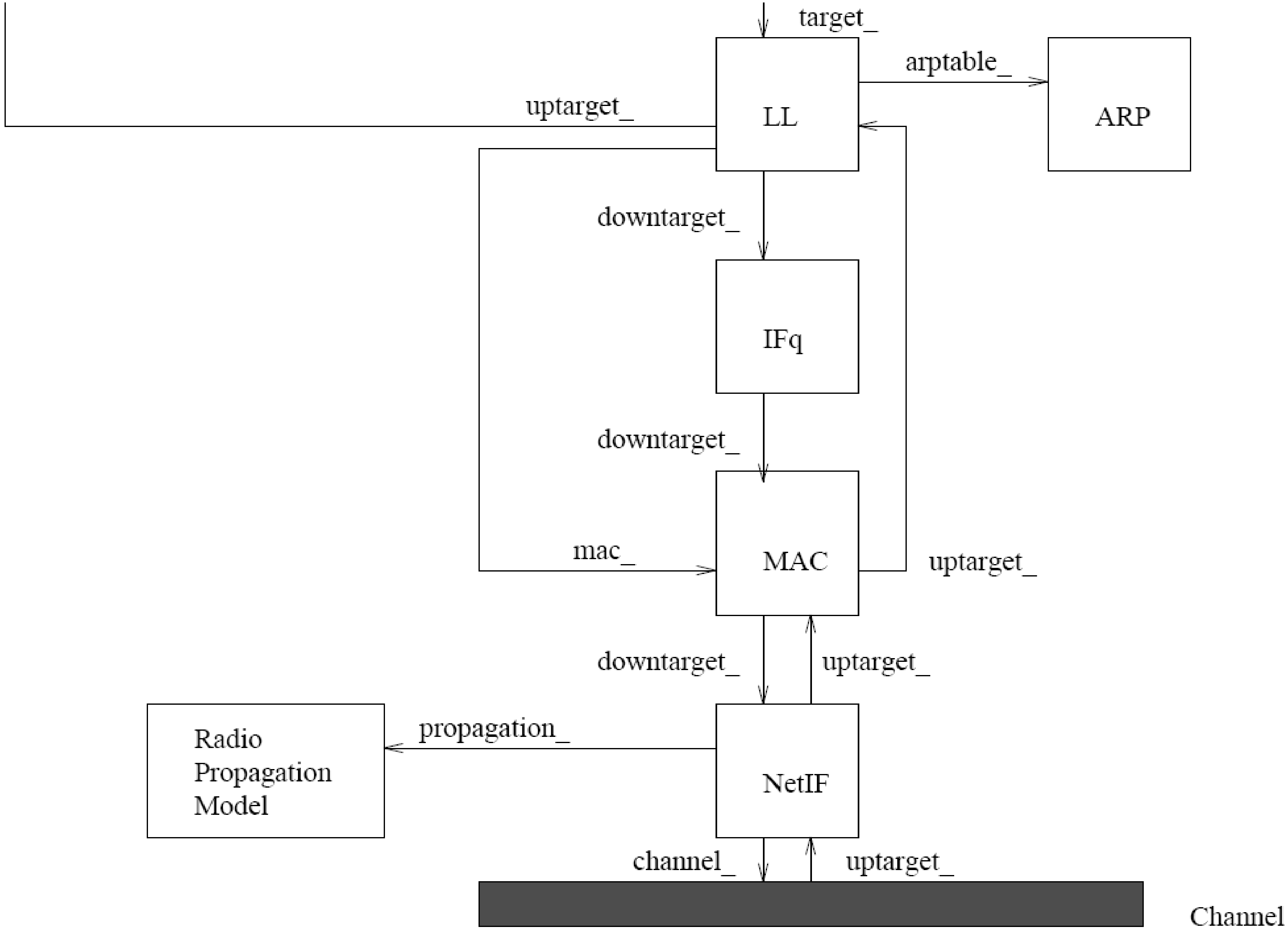
wiredRouting : Define if the node has a wired interface or not → ON, OFF

mobileIP : Define if mobile IP is used or not → ON, OFF

# Mobile Node in NS2 - Mobile Node Structure



# Mobile Node in NS2 - Mobile Node Structure



# Mobile Node in NS2 - Creating Node Movements

- Random movement

  $MN_(0) random-motion 1

  $MN_(0) start

- Determined movement

  $MN_(0) random-motion 0

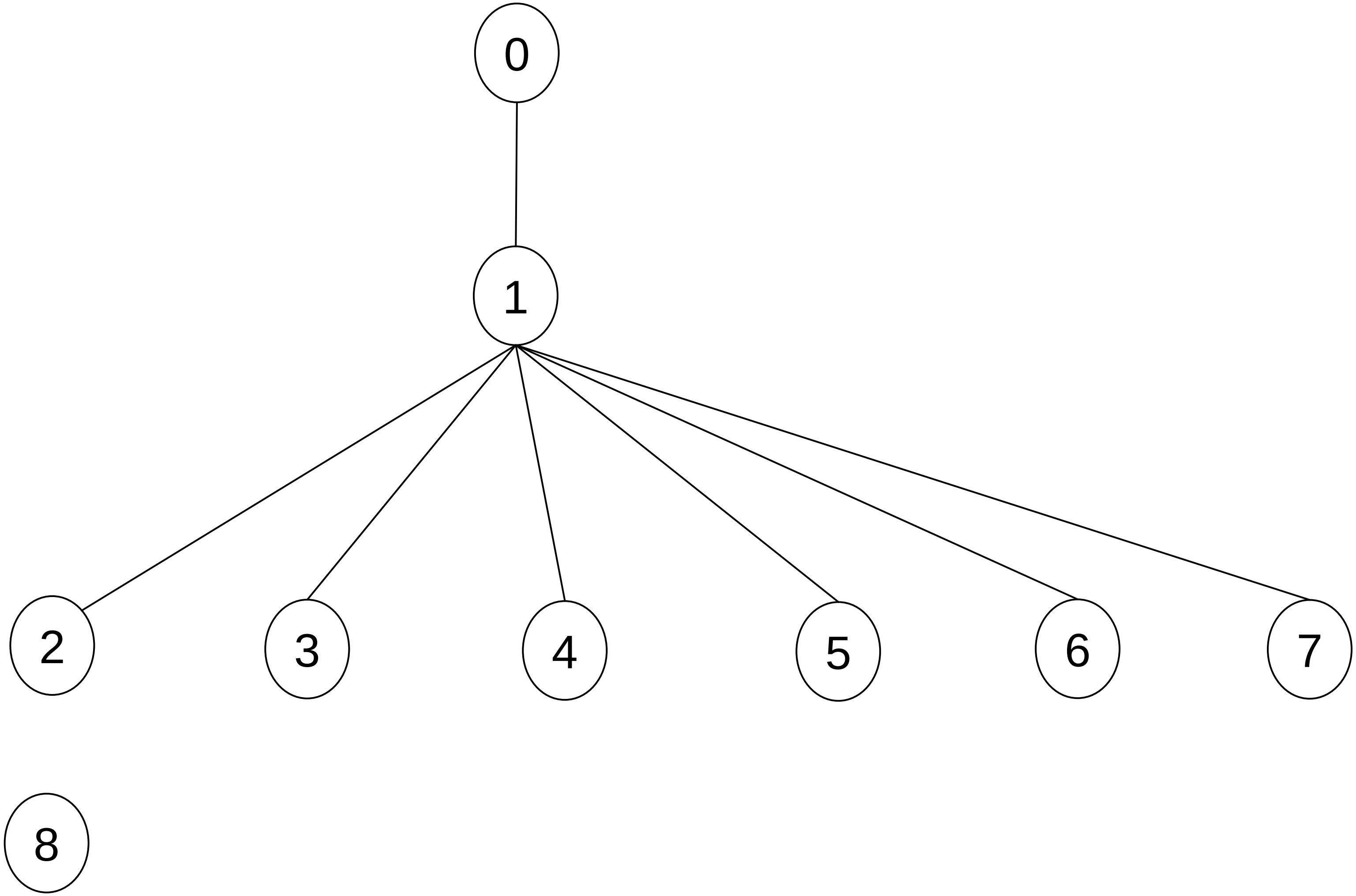
  $ MN_(0) set X_ <x1>

  $ MN_(0) set Y_ <y1>

  $ MN_(0) set Z_ <z1>

  $ns at <time (sec)> $MN_(0) setdest <x2> <y2> <speed (m/sec)>

# Example



# Example

```
# Firstly, we should define the wireless scenario options

set opt(chan)       Channel/WirelessChannel
set opt(prop)       Propagation/TwoRayGround
set opt(netif)      Phy/WirelessPhy
set opt(mac)        Mac/802_11
set opt(ifq)        Queue/DropTail/PriQueue
set opt(ll)         LL
set opt(ant)        Antenna/OmniAntenna
```

# Example

```
……….
……….
set opt(ifqlen)     32768
set opt(nn)         1
set opt(adhocRouting)    NOAH
set opt(x)          1000
set opt(y)          100
set opt(seed)       0.0
set opt(stop)       200.0
set opt(ftp-start)  0.0
set num_wired_nodes    2
```

# Example

```
# Create simulator instance
set ns_ [new Simulator]
```

```
# Create nam and trace files
set tracefd  [open out.tr w]
set namtrace [open out.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
```

```
# Create a file to record the lost packets for UDP
set LostPackets  [open UDPlost.tr w]
```

# Example

```
……
……
```

```
# Set up the hierarchical routing
$ns_ node-config -addressType hierarchical

AddrParams set domain_num_ 7
lappend cluster_num 1 1 1 1 1 1 1
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 2 2 1 1 1 1 1
AddrParams set nodes_num_ $eilastlevel
```

# Example

```
……
…….
```

```
# Create topography object
set topo   [new Topography]
```

```
# Define topology
$topo load_flatgrid $opt(x) $opt(y)
```

```
# Create God object
create-god [expr 6 + $opt(nn)]
```

# Example

```
#Create the wired nodes


set W(0) [$ns_ node 0.0.0]
set W(1) [$ns_ node 0.0.1]
```

```
#The above written code can be written as followed too
set temp {0.0.0 0.0.1}
for {set i 0} {$i < $num_wired_nodes} {incr i} {
set W($i) [$ns_ node [lindex $temp $i]]
}
# Note, this code is an alternative to the above written code. One of
# them is enough
```

# Example

```
# Configure for ForeignAgent and HomeAgent nodes
$ns_ node-config         -mobileIP ON \
                         -adhocRouting $opt(adhocRouting) \
                         -llType $opt(ll) \
                         -macType $opt(mac) \
                         -ifqType $opt(ifq) \
                         -ifqLen $opt(ifqlen) \
                         -antType $opt(ant) \
                         -propType $opt(prop)
                         -phyType $opt(netif) \
                         -channelType $opt(chan) \
                         -topoInstance $topo \
```

# Example

```
                    -wiredRouting ON \
                    -agentTrace ON \
                    -routerTrace OFF \
                    -macTrace ON
```

```
# Create HA and  five FAs
set HA [$ns_ node 1.0.0]
set FA [$ns_ node 2.0.0]
set FA1 [$ns_ node 3.0.0]
set FA2 [$ns_ node 4.0.0]
set FA3 [$ns_ node 5.0.0]
set FA4 [$ns_ node 6.0.0]
```

# Example

```
# Deactivate the random movement
$HA random-motion 0
$FA random-motion 0
$FA1 random-motion 0
$FA2 random-motion 0
$FA3 random-motion 0
$FA4 random-motion 0
```

```
# Define the coordinates of the base-station nodes (HA & FAs)
$HA set X_ 10.000000000000
$HA set Y_ 10.000000000000
$HA set Z_ 0.000000000000
```

# Example

```
$FA set X_ 150
$FA set Y_ 10.000000000000
$FA set Z_ 0.000000000000

$FA1 set X_ 290
$FA1 set Y_ 10.000000000000
$FA1 set Z_ 0.000000000000

$FA2 set X_ 330
$FA2 set Y_ 10.000000000000
$FA2 set Z_ 0.000000000000
```

# Example

```
$FA3 set X_ 470
$FA3 set Y_ 10.000000000000
$FA3 set Z_ 0.000000000000

$FA4 set X_ 600
$FA4 set Y_ 10.000000000000
$FA4 set Z_ 0.000000000000
```

# Example

```
# Create links between wired and wireless nodes
$ns_ duplex-link $W(0) $W(1) 100Mb 20ms DropTail
$ns_ duplex-link $W(1) $HA 100Mb 20ms DropTail
$ns_ duplex-link $W(1) $FA 100Mb 9ms DropTail
$ns_ duplex-link $W(1) $FA1 100Mb 9ms DropTail
$ns_ duplex-link $W(1) $FA2 100Mb 9ms DropTail
$ns_ duplex-link $W(1) $FA3 100Mb 9ms DropTail
$ns_ duplex-link $W(1) $FA4 100Mb 9ms DropTail
$ns_ duplex-link-op $W(0) $W(1) orient down
$ns_ duplex-link-op $W(1) $HA orient left-down
$ns_ duplex-link-op $W(1) $FA orient right-down
```

# Example

```
# create a mobile node that moves between the HA and the FAs.
# note address of MH indicates that its in the same domain as HA.
$ns_ node-config -wiredRouting OFF
set MH [$ns_ node 1.0.1]
set node_(0) $MH
set HAaddress [AddrParams addr2id [$HA node-addr]]
[$MH set regagent_] set home_agent_ $HAaddress
```

```
# Define the start position of the MN
$MH set X_ 10.000000000000
$MH set Y_ 20.000000000000
$MH set Z_ 0.000000000000
```

# Example

```
# Set up the movements of the MN
$ns_ at 20.00 "$MH setdest 150 20.00 20.00"
$ns_ at 40.00 "$MH setdest 290 20.00 20.00"
$ns_ at 60.00 "$MH setdest 330 20.00 11.11"
$ns_ at 80.00 "$MH setdest 470 20.00 16.00"
$ns_ at 100.00 "$MH setdest 600 20.00 20.00"
```

# Example

```
# Create a UDP agent. The traffic is a downlink traffic
set agent(0) [new Agent/UDP]
$ns attach-agent $W(0) $agent(0)
$agent(0) set fid_ 6
$ns color 6 "red„
set sink(0) [new Agent/LossMonitor]
$ns attach-agent $MH $sink(0)
$ns connect $agent(0) $sink(0)
```

```
# After that, we have to create traffic source and add it to the agent
set traffic_source(0) [new Application/Traffic/CBR]
$traffic_source(0) set interval_ 0.001950
$traffic_source(0) set paketSize_ 230
$traffic_source(0) attach-agent $agent(0)
```

# Example

```
# Write the number of lost packets in $LostPackets
proc record {} {
    global sink(0) LostPackets
    set ns [Simulator instance]
    set time 0.1
    set DP [$sink(0) set nlost_]
    set now [$ns now]
    puts $LostPackets "$now $DP"
    $sink(0) set nlost_ 0
    $ns at [expr $now+$time] "record"
    }
```

# Example

```
# Write the finish procedure
proc finish {} {
global ns namfile LostPackets
$ns flush-trace
close $namfile
close $LostPackets
exec nam out.nam &
exec xgraph UDPlost.tr
exit 0
}
```

# Example

```
# scheduling the start and the stop of the  traffic source
$ns_ at 3.0 "$traffic_source(0) start„
$ns_ at 100.0 "$traffic_source(0) stop„
```

```
# Schedule the finish procedure
$ns_ at 110.000000 "finish"
```

```
# Schedule the record procedure
$ns_ at 3.000000 „record"
```

```
# Start ns2
$ns_ run
```

# References

- Using ns and nam in Education : http://www.isi.edu/nsnam/ns/edu/
- The network simulator (ns2): http://www.isi.edu/nsnam/ns/
- Marc Greis's tutorial: http://www.isi.edu/nsnam/ns/tutorial/index.html
- SIMON - Simulation Environment for Mobile Networks: http://wcms1.rz.tu-ilmenau.de/fakia/index.php?id=1570