Network Simulator 2 (NS2)

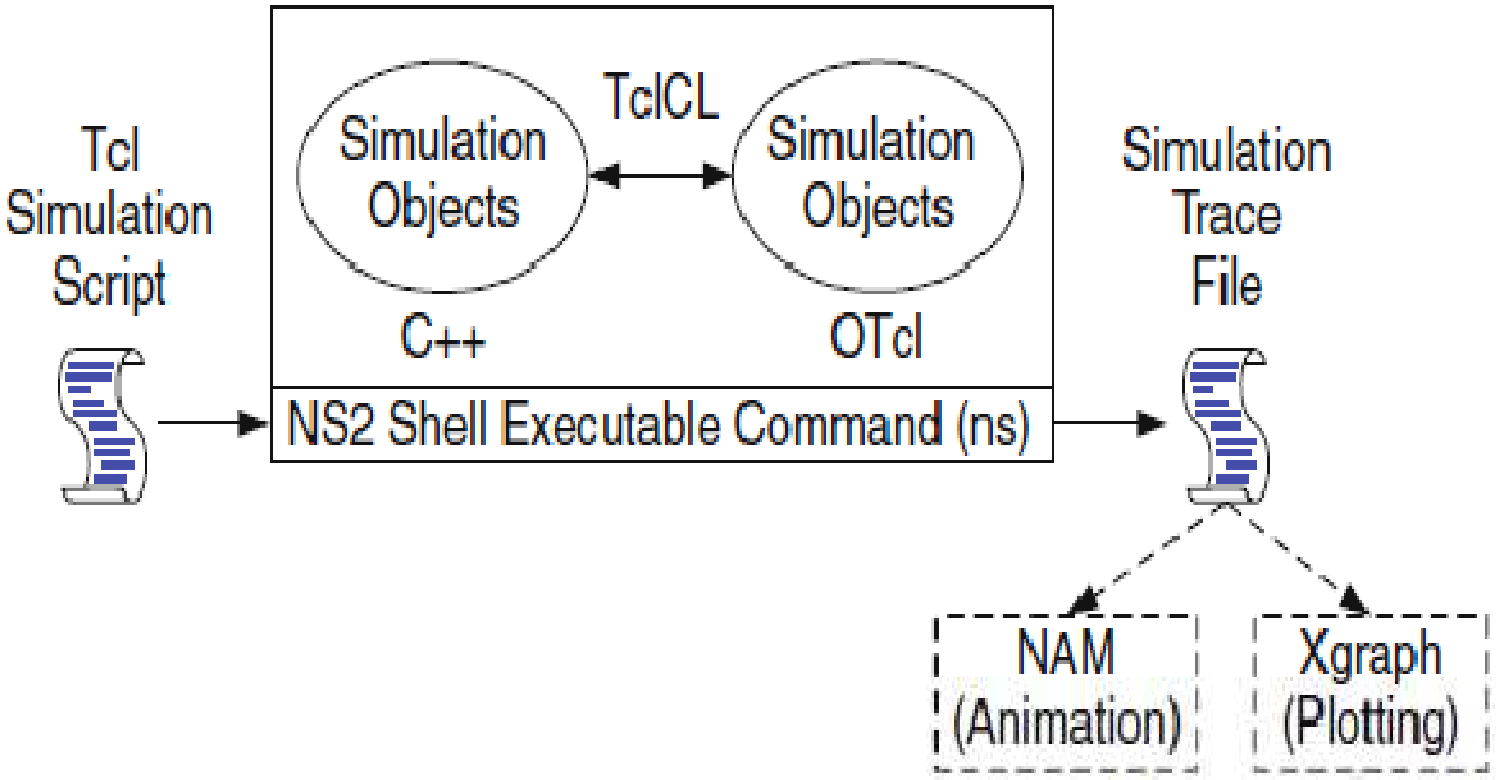
Introduction

- Simply an event driven simulation tool
- Proved useful in studying the dynamic nature of communication networks
- Simulation of wired as well as wireless network functions and protocols

Basic Architecture

- ns executable command
- Takes on input argument, the name of a Tcl (Tool Command Language) simulation scripting file.
- A simulation trace file is created, and is used to plot graph and/or to create animation

Basic Architecture Cont…



Basic Architecture Cont…

- NS2 consists of two key languages:
  - C++ which defines the internal mechanism (i.e., a backend) of the simulation objects
  - Object-oriented Tool Command Language (OTcl) which sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a front end)
- The C++ and the OTcl are linked together using TclCL(Tcl With Classes)

Why Two Languages?

- OTcl to create and configure a network
- C++ to run simulation
- Use OTcl
  - For configuration, setup, or one time simulation
  - To run simulation with existing NS2 modules.
- Use C++
  - When dealing with a *packet*
  - When modifying existing NS2 modules

The first Tcl script

- Develop a Tcl script for ns which simulates a simple topology
- How to set up nodes and links
- How to send data from one node to another
- How to monitor a queue and how to start nam from Tcl script to visualize the simulation.

Template

- Can write Tcl scripts in any text editor like joe or emacs
- Name the first example 'example1.tcl'.
- First step is to create a simulator object

  set ns [new Simulator]
- Open a file for writing the nam trace data

  set nf [open out.nam w]

  $ns namtrace-all $nf
- The first line opens the file 'out.nam' for writing and returns file handle 'nf'.
- In The second line, the simulator object ns writes all simulation data relevant for nam into the file out.nam.

Template

- The next step is to add a 'finish' procedure that closes the trace file and starts nam

```
proc finish { } {
        global ns nf
#Close the trace file
        $ns flush-trace
        close $nf
#Execute nam on the trace file
        exec nam out.nam &
        exit 0
}
```

Template

- The next line tells the simulator object ns to execute the 'finish' procedure after 5.0 seconds of simulation time.

  $ns at 5.0 "finish"

- The last line finally starts the simulation.

  $ns run

Network Links and Nodes

- The way to define a node is

  set n0 [$ns node]

- Referring to the above node is written as $n0
- Links connecting nodes can be done as

  $ns duplex-link $n0 $n1 1Mb 10ms DropTail

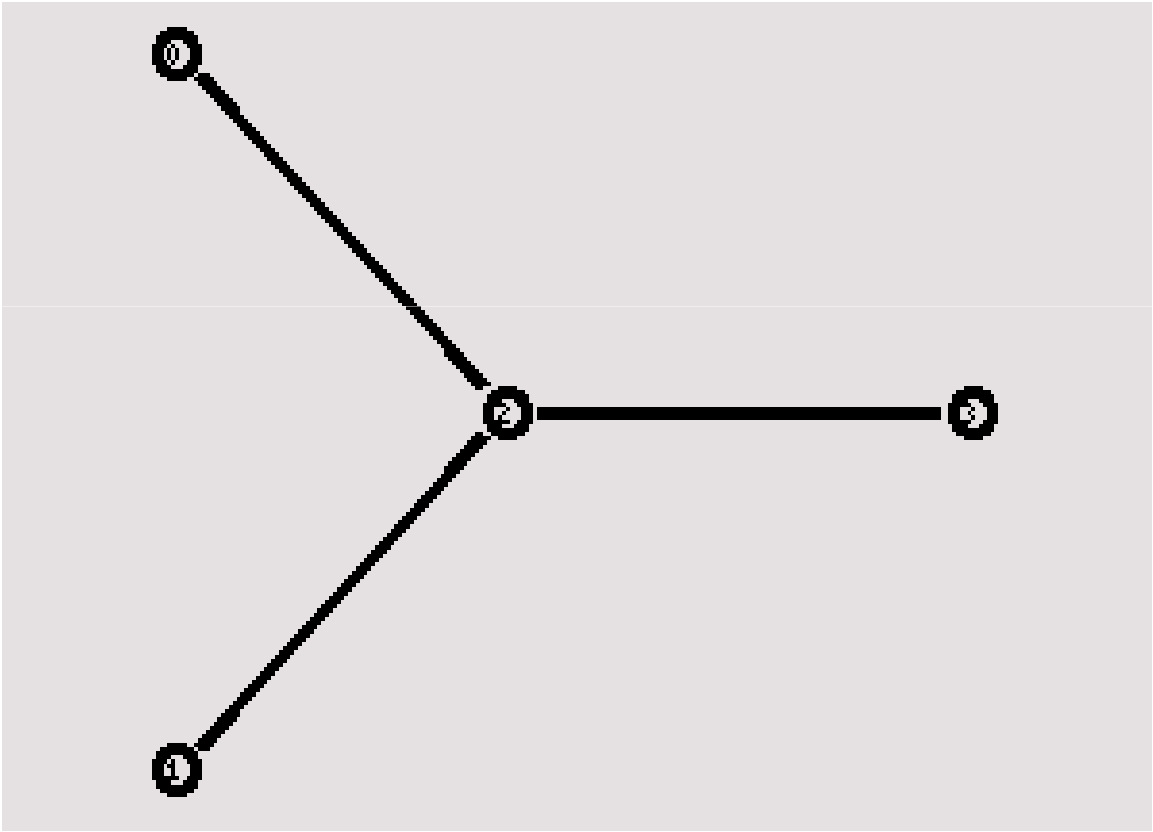
- Drop Tail, is a simple queue management algorithm used by Internet routers to decide when to drop packets

Example 1

```
#Create a simulator object
        set ns [new Simulator]
#Open the nam trace file
        set nf [open out.nam w]
        $ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the trace file
        close $nf
        #Execute nam on the trace file
        exec nam out.nam &
        exit 0
}
        set n0 [$ns node]
        set n1 [$ns node]
        set n2 [$ns node]
        set n3 [$ns node]
        $ns duplex-link $n0 $n2 1Mb 10ms DropTail
        $ns duplex-link $n1 $n2 1Mb 10ms DropTail
        $ns duplex-link $n3 $n2 1Mb 10ms DropTail

      $ns duplex-link-op $n0 $n2 orient right-down
      $ns duplex-link-op $n1 $n2 orient right-up
      $ns duplex-link-op $n2 $n3 orient right
#Call the finish procedure after 5 seconds simulation time
      $ns at 5.0 "finish"
#Run the simulation
      $ns run
```

Example 1



Agents

- After designing the topology make traffic flow between them
- Therefore define routing, agents (protocols), and applications with respect to the topology.
- Define a TCP agent as

  set tcp [new Agent/TCP]
- Attach a node with the agent
- **attach-agent** → used to connect different layer components

  $ns attach-agent $n0 $tcp

Connect

- **connect** → Used to connect same layer components
- Connect command used for a connection between source and destination

  $ns connect $tcp1 $tcp2

Application

- **attach-agent** → used to connect different layer components
- Attaching an application

  set ftp [new Application/FTP]

  $ftp attach-agent $tcp1

Scheduling Events

- The Tcl script defines when an event should occur

  $ns at <time> <event>

  $ns at 0.5 "$ftp start"

Example 2

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
```

```
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Execute nam on the trace file
exec nam out.nam &
exit 0
}
```

Example 2

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

Example 2

```
# Create a CBR traffic source and attach it to udp0
    set cbr0 [new Application/Traffic/CBR]
    $cbr0 set packetSize_ 512
    $cbr0 set interval_ 0.005
    $cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to
    node n1
    set null0 [new Agent/Null]
    $ns attach-agent $n1 $null0
```

Example 2

```
#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"
#Call the finish procedure after 5 seconds of simulation time
    $ns at 5.0 "finish"
#Run the simulation
    $ns run
```

Example 2