EEC 687 Mobile Computing (Spring, 2009)

Ns-2 Laboratory

Prof. Chansu Yu

http://academic.csuohio.edu/yuc/
c.yu91@csuohio.edu

Contents

ns-2 Overview

- What is ns-2?
  - Abbreviation of Network Simulator
  - Discrete event simulator targeted at networking (wired and wireless) research
  - Basically, a TCL interpreter
- Where to get?
  - Free and open source
  - ns website http://www.isi.edu/nsnam/ns/
- Working platforms
  - Most UNIX or UNIX-like systems; e.g. Linux
  - Windows (using cygwin)

Linux

- Genuine Linux
  - Either Fedora or Ubuntu
- Cygwin
  - Linux interface on top of MS Windows
  - http://nsnam.isi.edu/nsnam/index.php/Running_Ns_and_Nam_Under_Windows_9x/2000/XP_Using_Cygwin

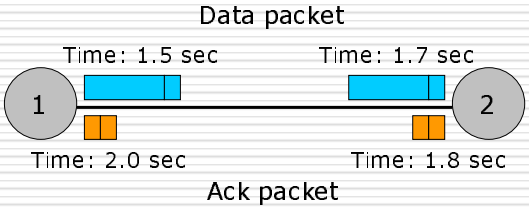
* Class homepage:
http://mobicomclass.wordpress.com/

Download and Install ns-2

- Reading
  - Wireless and Mobility Extensions to ns-2 (http://www.isi.edu/nsnam/ns/tutorial/nsindex.html).
  - Ns2 manual, "Mobile networking in ns," Ch. 16
- Download the latest ns-2 (version 2.30) with additional components source from http://www.isi.edu/nsnam/dist/ns-allinone-2.30.tar.gz and put it to your desirable folder, say /home/student.
- Run the following commands at /home/student :
  - % gunzip ns-allinone-2.30.tar.gz
  - % tar –xvf ns-allinone-2.30.tar
- Run the script at /home/student/ns-allinone-2.30 : "% ./install"
  - This installation script will check your Linux environment, compile and install your ns-2 system.

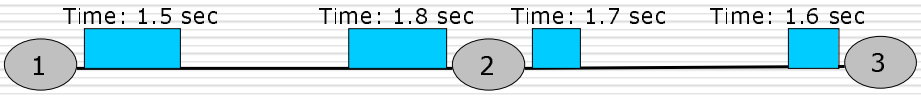
ns-2: Discrete Event Simulator

- ns-2 is an discrete event driven simulation
  - Physical activities are translated to events
  - Events are queued and processed in the order of their scheduled occurrences
  - Time progresses as the events are processed



ns-2: Discrete Event Simulator

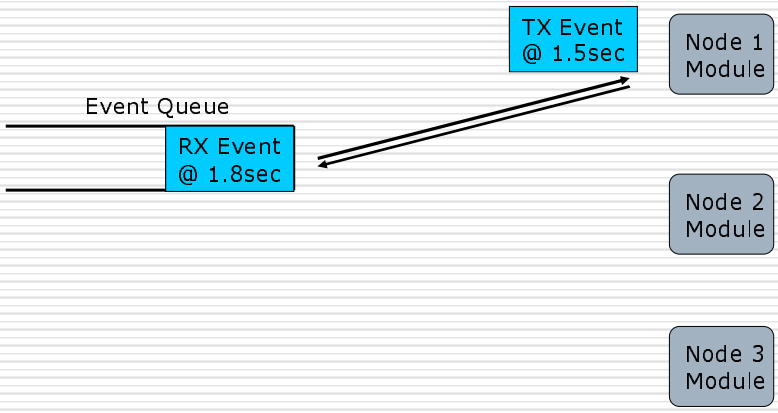
- ns-2 is an discrete event driven simulation
  - Physical activities are translated to events
  - Events are queued and processed in the order of their scheduled occurrences
  - Time progresses as the events are processed



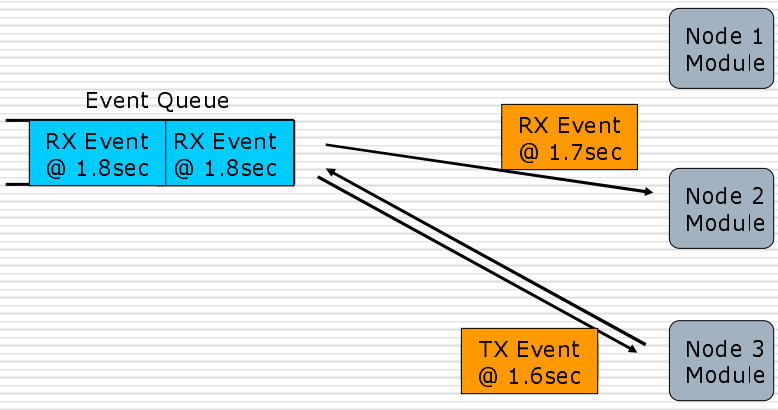
ns-2: Discrete Event Simulator

- T=1.5, node 1 tx a packet to node 2 (packet generation event)
  - T=1.8, node 2 will receive (packet reception event)
- T=1.6, node 3 tx a packet to node 2 (packet generation event)
  - T=1.7, node 2 will receive (packet reception event)

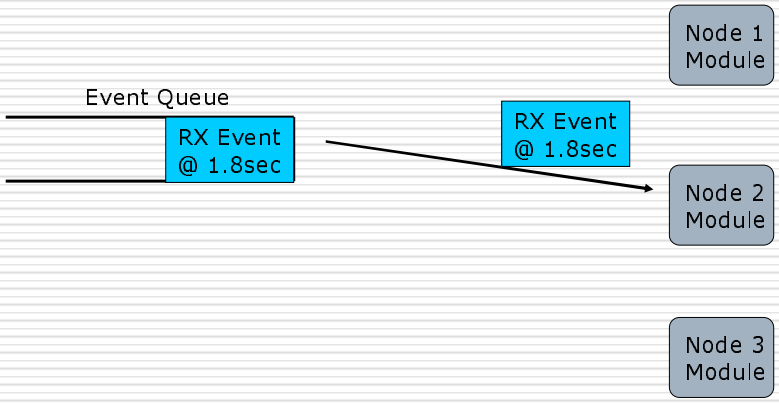
Event Driven Simulation



Event Driven Simulation



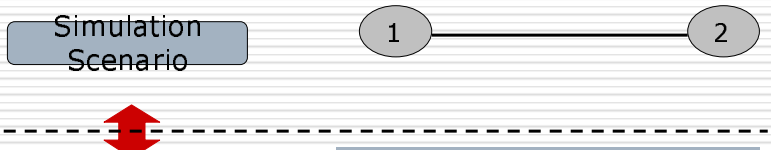
Event Driven Simulation



ns-2 uses two languages? (Tcl & C++)

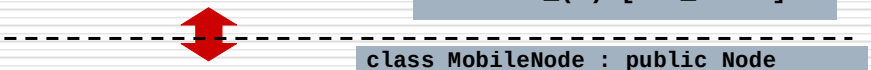
- C++: Detailed protocol simulations require systems programming language
  - byte manipulation, packet processing, algorithm implementation
  - Run time speed is important
  - Turn around time (run simulation, find bug, fix bug, recompile, re-run) is slower
- Tcl: Simulation of slightly varying parameters or configurations
  - quickly exploring a number of scenarios
  - iteration time (change the model and re-run) is more important

ns-2 Environment



Tcl Script

```
set ns_ [new Simulator]
set node_(0) [$ns_ node]
set node_(1) [$ns_ node]
```



C++ Implementation

```
class MobileNode : public Node
{
        friend class PositionHandler;
public:
         MobileNode();
                 .
                 .
}
```

Basic TCL

(http://tmml.sourceforge.net/doc/tcl/index.html)

```
set a 43
set b 27
set c [expr $a + $b]
set d [expr [expr $a - $b] * $c]
for {set k 0} {$k < 10} {incr k} {
    if {$k < 5} {
            puts "k < 5, pow= [expr pow($d, $k)]"
    } else {
            puts "k >= 5, mod= [expr $d % $k]"
    }
}
```

Hello World - Interactive Mode

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

Hello World - Passive Mode

simple.tcl

```
set ns [new Simulator]
$ns at 1 “puts \“Hello World!\””
$ns at 1.5 “exit”
$ns run
```

% **ns simple.tcl**
Hello World!

%

Simulation with ns-2

- Creating the event scheduler
- Creating network: nodes, links & queue
- Computing routes
- Creating connection
- Creating traffic
- Inserting errors
- Tracing
- Wireless Support

Protocols or Controls Implemented in ns2

- Transport layer (traffic agent)
  TCP; UDP
- Network layer (routing agent)
  - Wired
    Distance vector; Link state (patch needed)
  - Wireless
    AODV; DSR; DSDV; TORA
- Interface queue
  FIFO queue; DropTail queue; Priority queue; etc.
- Logic link control layer
  IEEE 802.2; ARP

Protocols or Controls Implemented in ns2 (cont.)

- MAC layer
  - Wired
    - IEEE 802.3 (CSMA/CD)
  - Wireless
    - IEEE 802.11 (CSMA/CA)
      - DCF
      - PCF (partially implemented)
- Physical layer
  - Wired
    - IEEE 802.3
  - Wireless
    - IEEE 802.11
      - DSSS (Direct Sequence Spread Spectrum)
      - FHSS (Frequency-Hopping Spread Spectrum); not implemented
      - IR (Infrared); not implemented

Protocols or Controls Implemented in ns2 (cont.)

- Wireless channel
  - Friss-space model
  - Two-ray ground model
  - Shadowing model
  - Fading model (patch needed)
  - Omni directional antenna

How to Use ns-2?

- Design simulation
  Determine simulation scenario, parameters.
- Build ns-2 script using tcl
  If necessary implement algorithm using C++.
- Run simulation
  For convenience use shell batch file.
- Analyze simulation results
  Use shell command or programming languages.

Visualization is Possible in ns-2

- Example
  - nam-1.12/edu/C2-sliding-color.nam
  - nam-1.12/tcl/test-wireless-2.nam

ns Tutorials

- NS website http://www.isi.edu/nsnam/ns/
- NS Manual
  http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf
- Marc Greis's Tutorial
  http://www.isi.edu/nsnam/ns/tutorial
- http://www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2-tutorial-exercise1.pdf
- http://nile.wpi.edu/NS/
- http://nesl.ee.ucla.edu/courses/ee206a/2002s/guest_presentations/GP02_Park_ns2.ppt
- http://www.ece.ubc.ca/~elec565/ns2_tutorial.ppt