Ubiquitous Computing

NETWORK SIMULATOR-2

Shankaraiah.
Protocol Engineering & Technology (PET) Lab
Electrical Communication Engineering,
Indian Institute of Science,
Bangalore – 560012, India
Outline

• Introduction
• Fundamental Skills - Tcl and OTcl
• Network Simulator - NS-2
• An examples
Introduction

• NS2 history
  - Modified from REAL network simulator
  - Developed through VINT project at UCB
  - NS1 vs. NS2

• NS version 2 is a discrete-event driven and object-oriented network simulator
  - Type of simulation: continuous, discrete event, and combined
Introduction

• Why to simulate?
  – Performance Evaluation

• Main categories for performance evaluation of computer systems, protocols, etc:
  – Analytic/mathematical models:
• Queuing theory, markov chains, etc
• Advantages: where possible, just plug numbers to give answers
• Disadvantages: real system is too complex to model (e.g. traffic pattern), so simplify too much??
Introduction

- **Measurement:**
  - Build actual network or system
  - Stress test under various scenarios and measure
  - Advantage: captures real system attributes
  - Disadvantages: expensive and takes time to build

- **Third option is simulation**
  - Write computer program that abstracts most relevant aspects of protocol
  - Maintain event queue (insert, remove, action, etc) of activities
  - More subtle events we add, more realistic results
  - Combines more accuracy of real system with speed of evaluation
  - Advantages: more accurate than analytic model
  - Disadvantages: Less accurate than real system measurement
Introduction

- More than 90% of networking papers use simulation
- Researchers used to write all their simulations from scratch
- NOW, 3 main simulators for networking research:
  - Network Simulator 2 (NS2) (ISI at USC) (free)
  - GlomoSim (UCLA) (free)
  - OPNET (commercial)

- Majority of all published wireless networking papers use one of these 3 simulators
- NS2: learning curve, may start going through tutorials.
- Support networking research and education
  - Protocol design, traffic studies, etc
  - Protocol comparison
Overview of NS

- Provide a collaborative environment
  - Freely distributed, open source
- Share code, protocols, models, etc
  - Allow easy comparison of similar protocols
  - Increase confidence in results
- More people look at models in more situations
- Experts develop models
- Multiple levels of detail in one simulator
NS functionalities

• Wired world
  – Routing DV, LS, PIM-SM
  – Transportation: TCP and UDP
  – Traffic sources: web, ftp, telnet, cbr, stochastic
  – Queuing disciplines: droptail, RED, FQ, SFQ, DRR
  – QoS: IntServ and Diffserv

• Wireless
  – Infrastructure Wireless (Wired & Wireless) Network
  – Ad Hoc Wireless Networks
  – Sensor Networks
  – Mobile Ipv4 and IPv6, UMTS, GPRS

• Satellite
  Tracing, visualization, various utilities
NS-2, the Network Simulator

- A discrete event simulator
  - Simulator has list of events
  - Process: take next one, run it, until done.
  - Each event happens in an instant of virtual (simulated) time,
  - but takes an arbitrary amount of real time

- Focused on modeling network protocols
  - Wired, wireless, satellite
  - TCP, UDP, multicast, unicast
  - Web, telnet, ftp
  - Ad hoc routing, sensor networks
  - Infrastructure: stats, tracing, error models, etc
**NS Components**

- NS, the simulator itself
- Nam, the network animator
  - Visualize NS (or other) output
  - Nam editor: GUI interface to generate ns scripts
- Preprocessing:
  - Traffic and topology generators
- Post processing:
  - Simple trace analysis, often in Awk, Perl, or Tcl
NS 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 queuing:
  – Wired routing, ad hoc routing and directed diffusion
  – queuing protocols: RED, droptail, etc
• Physical media:
  – Wired (point-to-point, LANs), wireless (multiple propagation models), satellite, etc.
NS Architecture
• Object-oriented (C++, Otcl)
  – C++ for Data:
  – Per packet processing, core of NS
  – Fast to run, detailed, complete control
  – Otcl for Control:
• Simulation scenario configurations
  – Modular approach
+ Reusability
+ Maintenance
  – Performance (speed and memory)
Using NS

1. **Problem**
2. **Simulation model**
3. **Modify ns**
4. **Setup/run simulation with ns**
5. **Result analysis**
How NS works

- OTcl: Tcl interpreter with OO extension
- NS Simulator Library
  - Event Scheduler Objects
  - Network Component Objects
  - Network Setup Helping Modules (Plumbing Modules)

Simulation Results → Analysis

NAM
Network Animator
An example: Skeleton

• A NS-2 Simulation script generally includes
  • Create the event scheduler
  • Turn on tracing, if needed
  • Create network topology
  • Setup routing
  • Create transport agent
  • Create traffic source/sink
  • Transmit application-level data
An example: how to start

• Create a event sheduler
  - set ns [new Simulator]

• Open a file for trace data
  - set nf [open out.nam w]
  - $ns namtrace-all $nf
An example: How to start

- A procedure to close file and start NAM
  ```
  proc finish{
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
  }
  ```

- Schedule the procedure
  ```
  $ns at 5.0 "finish"
  ```

- Start simulation
  ```
  $ns run
  ```
An example: Topology

• **Node**
  - set n0 [$ns node]
  - set n1 [$ns node]
  - set n2 [$ns node]

• **Link**
  - $ns duplex-link $no $n1 1mb 5ms Drop Tail
  - $ns duplex-link $n1 $n2 400kb 10ms Drop Tail
An example: Agent/application

• Create a UDP agent and attach it to node no
  - set udp [new Agent/UDP]
  - $ns attach-agent $n0 $udp

• Create a CBR traffic source and attach it to udp0
  - set cbr [new application/Traffic/CBR]
  -$cbr attach-agent $udp

• Create a null agent to be traffic sink
  - set null[new Agent/Null]
  -$ns attach-agent $ns2 $null
An example: Agent/application

• Connection them
  - $ns connect $udp $null

• Schedule the event
  -$ns at 0.5 “$cbr start”
  -$ns at 4.5 “$cbr stop”
An example: Agent/Application

• Create a TCP agent and attach it to node no
  - set tcp [new Agent/TCP]
  - $ns attach-agent $n0 $tcp

• Create a FTP traffic source and attach it to udp0
  - set ftp [ new Application/FTP]
  - $ftp attach-agent $tcp

• Create a TCPSink agent to be traffic sink
  - set sink [new Agent/TCPSink]
  - $ns attach-agent $n2 $sink
An example: Agent/application

• Schedule the event
  - $ns$ at 0.5 “ftp start”
  - $ns$ at 4.5 “ftp stop”
Script Example

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail
set tf [open trace.nam w]
$ns namtrace-all $tf
```

```
set tcp [new Agent/TCP]
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n1 $tcpsink
$ns connect $tcp $tcpsink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.2 "$ftp start"
$ns at 1.2 "exit 0"
$ns run
```
Tracing: dump everything into a file

- Trace packets on individual link
- Tracefile format:

  `<event> <time> <from> <to> <pkt> <size>--<flowid> <src> <dst> <seqno> <aseqno>`

  ```
  + 1       0 2 tcp 900 ------- 1 0.0 3.1 7 15
  - 1       0 2 tcp 900 ------- 1 0.0 3.1 7 15
  r 1.00234 0 2 tcp 900 ------- 1 0.0 3.1 7 15
  ```

+ enqueue
- dequeue
r receive
d drop

**Nodes involved in this event:**

- **Node 0:** enqueue, drop, receive
- **Node 1:** dequeue

**Time:**

- time
Tracing: dump everything into a file

- Trace packets on individual links
- Tracefile format:

```
<event> <time> <from> <to> <pkt> <size>--<flowid> <src> <dst> <seqno> <aseqno>
+ 1       0 2 tcp 900 ------- 1 0.0 3.1 7 15
- 1       0 2 tcp 900 ------- 1 0.0 3.1 7 15
r 1.00234 0 2 tcp 900 ------- 1 0.0 3.1 7 15
```

+ enqueue
- dequeue
r receive
d drop

- time
- packet type
- packet flags
- packet length
- flow ID
- source
- dest addresses
- seq number
- packet ID

nodes involved in this event
Visualization Tools

- nam (Network AniMator)
  - Packet-level animation
  - Well supported by NS
- xgraph
  - Simulation results
Ubiquitous Computing
Wired Example

- We will consider following network topology.
  - With four nodes, 0,1,2 and 3.
  - Links 0-2, 1-2 have bandwidth 5Mb and 2ms delay
  - Links 2-3 has bandwidth 1.5Mb and 10ms delay
- TCP Connection from 0-3, with FTP Traffic, starts at 0.5ms, and look the simulation
- Now, limit queue length of link 2-3 to 10 and look the simulation.
- Now, Add UDP Connection from 1-3, with CBR Traffic with packet size 1000Byte, inter-arrival time: 8ms, start at time 1.0, and look the simulation.
Example for Throughput Analysis

• Create a new scenario using NAM with 2 nodes and do the following with NAM default parameter,

• Create a Link between the nodes.

• Create TCP Connection form one node to another.

• Use FTP Traffic.

• Run simulation for 10 seconds and get the throughput graph.

• Run same simulation with 30 second, 140 seconds and get the graph.
Wireless Network

- Wireless network
  - Nodes can move
  - No explicit “links” used to connect nodes
- Wireless network extension to NS2
  - Mobile node
  - Packet headers
  - Wireless channel and propagation model
  - Topology and movement
  - Routing and forwarding
Wireless Channel

- Duplicate packets to all mobile nodes attached to the channel except the sender
  - Propagation delay is included
  - Use of multiple channels are possible
- It is the receiver’s responsibility (PHY) to decide if it will accept the packet
  - Decision is based on received signal power
    - Each packet will have the transmission power stamped
    - Currently interference from other transmissions is not included in reception decision
    - Collision is handled at individual receiver
Example: Ad Hoc Network

• Scenario
  − 3 mobile nodes
  − Move within 500m*400m flat topology
  − AODV ad hoc routing protocol
  − Random way point mobility model
  − TCP & FTP Traffic
An Example – Step 1

# Create simulator
set ns [new Simulator]

# Create a topology in a 670m x 670m area
set topo [new Topography]
$topo load_flatgrid 670 670

# ns trace and nam trace
$ns trace-all [open ns.tr w]
$ns namtrace-all-wireless [open ns.nam w] 670 670
An Example – Step 2

```plaintext
# Create God
set god [create-god 3]
```

- **God**: General Operations Director
  - Keep the number of nodes in the network
  - Called by 802.11 MAC to keep a sequence number cache of all nodes
  - Store an array of the smallest number of hops required to reach one node to another
  - Used for setdest operation
    
    ```
    $ns at 100.00 "$god set-dist 2 3 1"
    ```
An Example – Step 3

# Define how to create a mobile node

$ns node-config \\
  -adhocRouting DSR \\
  -llType LL \\
  -macType Mac/802_11 \\
  -ifqLen 50 \\
  -ifqType Queue/DropTail/PriQueue \\
  -phyType Phy/WirelessPhy \\
  -antType Antenna/OmniAntenna \\
  -propType Propagation/TwoRayGround \\
  -channel [new Channel/WirelessChannel] \\
  -topoInstance $topo \\
  -agentTrace ON \\
  -routerTrace OFF \\
  -macTrace OFF \"
Energy Parameters

$ns \ node-config \ 
  \ -energyModel \ EnergyModel \ 
  \ -initialEnergy 100.0 \ 
  \ -txPower 0.6 \ 
  \ -rxPower 0.2

• Node is energy-aware
  - Node status: on / off / sleep
• Pt_ and Pt_consume_
An Example – Step 4

# Create mobile nodes
for {set i 0} {i<3} {incr i} {
  set node($i) [$ns node]
  # disable random motion for static network
  $node($i) random-motion 0
}

# Define movement model (if applicable)
source movement-scenario-files

# Define traffic model (if applicable)
source traffic-scenario-files
A Movement File

$node_0$ set \texttt{X} \ 83.4
$node_0$ set \texttt{Y} \ 239.4
$node_0$ set \texttt{Z} \ 0.0
$node_1$ set \texttt{X} \ 257.1
$node_1$ set \texttt{Y} \ 345.4
$node_1$ set \texttt{Z} \ 0.0
$node_2$ set \texttt{X} \ 591.3
$node_2$ set \texttt{Y} \ 199.4
$node_2$ set \texttt{Z} \ 0.0
$ns$ at 33.0 "$node_0$ setdest 89.7 283.5 19.2"
$ns$ at 51.0 "$node_1$ setdest 221.8 80.9 14.9"
$ns$ at 50.0 "$node_2$ setdest 369.5 170.5 3.4"
A Traffic Scenario

```plaintext
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_ 1000
$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)
$ns_ at 20.0 "$cbr_(0) start"
```
An Example – Step 5

# Define node initial position in nam
for {set i 0} {$i < 3} {incr i} {
    $ns initial_node_position $node($i) 20
}

# Tell ns/nam the simulation stop time
$ns at 100.0 "$ns nam-end-wireless 100.0"
$ns at 100.0 "$ns halt"

# Start your simulation
$ns run
Traces

- Traces in NS format
  - `$ns trace-all [open tr.out w]`

    `<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`
    `d 1.04218 1 2 cbr 210 -------- 0 0.0 3.1 0 0`

- Traces in NAM format
  - `$ns namtrace-all [open tr.nam w]`

- Turn on tracing on specific links
  - `$ns trace-queue $n0 $n1`
  - `$ns namtrace-queue $n0 $n1`
More setting: Event and queueing

- **Schedule events**
  - `$ns at <time> <event>`
  - `<event>`: any legitimate ns/tcl commands

- **Links and queuing**
  - `$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>`
  - `<queue_type>`: DropTail, RED, CBQ, FQ, SFQ, DRR
More settings: Routing

- **Unicast**
  - `$ns rtproto <type>`
  - `<type>`: Static, Session, DV, cost, multi-path

- **Multicast**
  - `$ns multicast` (right after [new Simulator])
    - or set `ns [new Simulator –multicast on]`
  - `$ns mrtproto <type>`
  - `<type>`: CtrMcast, DM, ST, BST (centralized, dense mode, shared tree)
More settings: Traffic on Top of TCP

- **TCP**
  - set tcp [new Agent/TCP]
  - set tcpsink [new Agent/TCPSink]
  - $ns attach-agent $n0 $tcp
  - $ns attach-agent $n1 $tcpsink
  - $ns connect $tcp $tcpsink

- **FTP**
  - set ftp [new Application/FTP]
  - $ftp attach-agent $tcp

- **Telnet**
  - set telnet [new Application/Telnet]
  - $telnet attach-agent $tcp