Network Simulator 2: Introduction

Presented by Ke Liu
Dept. Of Computer Science
SUNY Binghamton

Spring, 2006
NS-2 Overview
NS-2

- Developed by UC Berkeley
- Maintained by USC
- Popular simulator in scientific environment
- Other popular network simulators
  - QualNet: based on GloMoSim
  - Others: GloMoSim, OPNET, etc

© Ke Liu, SUNY-Binghamton, 2006
NS2 Goals

• To support networking research and education
  – Protocol design, traffic studies, etc.
  – Protocol comparison;
  – New architecture designs are also supported.

• To provide *collaborative* environment
  – Freely distributed, open source;
  – *Increase confidence* in result

© Ke Liu, SUNY-Binghamton, 2006
Two Languages: C++, OTcl

OTcl: short for MIT Object Tcl, an extension to Tcl/Tk for object-oriented programming.

- Used to build the network structure and topology which is just the surface of your simulation;

- Easily to configure your network parameters;

- Not enough for research schemes and protocol architecture adaption.
Two Languages (Con’t)

C++: Most important and kernel part of the NS2

- To implement the kernel of the architecture of the protocol designs;

- From the packet flow view, the processes run on a single node;

- To change or “comment out” the existing protocols running in NS2;

- Details of your research scheme.
Why 2 Languages?

- 2 requirements of the simulator
  - Detailed simulation of Protocol: Run-time speed;
  - Varying parameters or configuration: easy to use.

- C++ is fast to run but slower to code and change;

- OTcl is easy to code but runs slowly.

© Ke Liu, SUNY-Binghamton, 2006
Protocols/Models supported by NS2

- **Wired Networking**
  - Routing: Unicast, Multicast, and Hierarchical Routing, etc.
  - Transportation: TCP, UDP, others;
  - Traffic sources: web, ftp, telnet, cbr, etc.
  - Queuing disciplines: drop-tail, RED, etc.
  - QoS: IntServ and Diffserv

- **Wireless Networking**
  - Ad hoc routing and mobile IP
    - Routing Protocol: AODV, DSDV, DSR, etc.
    - MAC layer Protocol: TDMA, CDMA(?), IEEE Mac 802.x, etc.
    - Physical layers: different channels(?), directional attena

- **Sensor Networks**
Researches based on NS2

- Intserv/Diffserv (QoS)

- Multicast: Routing, Reliable multicast

- Transport: TCP Congestion control

- Application: Web caching Multimedia

- Sensor Networks: LEACH, Directed Diffusion, etc.

- etc.
NS2 research actions

- **NS2**: the simulator itself, now version: ns-2.29
  We will work with the part mostly.

- **NAM**: Network animator. Visualized trace tool(not really).
  My recommendation is that ”Don’t use nam at all”.

- **Pre-processing**:  
  Traffic and topology generators

- **Post-processing**:  
  Simple trace analysis, often in Awk, Perl(mostly), or Tcl

© Ke Liu, SUNY-Binghamton, 2006
Living under NS2
The NS2 Directory Structure

ns-allinone
  └── Tcl8.0
  └── TK8.0
  └── OTcl
  └── tclcl
  └── ns-2
  └── nam-1
       └── tcl
            └── C++ code

       └── ex
            └── examples

       └── test
            └── validation tests

       └── lib

       └── mcast
            └── OTcl code
Warning

- Try to avoid using ns2 with version before 2.27
- DO NOT use gcc 4.x, suggestion: gcc3.3
- If you work with MAC layer protocols, please be careful for the versions

© Ke Liu, SUNY-Binghamton, 2006
A Simple Simulation

We just need one Tcl script to do so.
A Simple Simulation, part 1: set up

$ns_ use-scheduler Heap

;#Specify the scheduler, default: List Scheduler

set ns [new Simulator]

;#Create a simulator object

$ns color 1 Blue
$ns color 2 Red

;#Define different colors for data flows (for NAM)

set nf [open out.nam w]
$ns namtrace-all $nf

set tf [open trace.tr w]
$ns trace-all $tf

;#Create trace files for simulation

proc finish {} {
    global ns nf tf
    $ns flush-trace

    #Close the NAM trace file
    close $nf

    #Close the ns2 trace file
    close $tf

    exit 0
}

© Ke Liu, SUNY-Binghamton, 2006
A Simple Simulation, part 2: n/w structure

#Create four nodes

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)

$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

#Monitor the queue for link (n2-n3). (for NAM)

$ns duplex-link-op $n2 $n3 queuePos 0.5

© Ke Liu, SUNY-Binghamton, 2006
A Simple Simulation, part 3: Transport and Traffic

# Setup a TCP connection: from node 0 to node 3

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_1

# Setup a FTP over TCP connection

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP; #for Name

# Setup a UDP connection: from node 1 to node 3

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_2
A Simple Simulation, part 4: traffic

# Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]
cbr attach-agent $udp $cbr
set type_CBR $cbr ; # used by Nam

set packet_size_ 1000 $cbr
set rate_ 1mb $cbr
set random_ false ; # generating traffic periodically

# Schedule events for the CBR and FTP agents

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

# Detach tcp and sink agents (not really necessary)

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

# Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

# Run the simulation

$ns run

© Ke Liu, SUNY-Binghamton, 2006
Steps in writing a simulating script

- Create the event scheduler
- Turn on tracing
- Create network
- Setup routing
- Insert errors
- Create transport connection
- Create traffic
- Transmit application-level data
The trace file

- Turn on tracing on specific links
  
```bash
$ns_trace-queue $n0 $n1
```

- Each line in the trace file is in the format:
  
```plaintext
<event> <time> <from> <to> <pkt-type> 
<pkt-size> <flags> <fid> <src.port> <dst.port> 
<seq> <unique pkt id>
```

- Trace example:
  
```plaintext
+ 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: `s` send, `r` receive, `+` enqueue, `−` dequeue, `d` drop, `f` forward,
The network Topology

© Ke Liu, SUNY-Binghamton, 2006
The Node Architecture

© Ke Liu, SUNY-Binghamton, 2006
The Packet Flow
Extending to NS2

© Ke Liu, SUNY-Binghamton, 2006
Class Hierarchy in NS2 (Partial, C++ code)

© Ke Liu, SUNY-Binghamton, 2006
Create New Component for NS2

Your research needs you to do so, no escaping(crying!!!).

- Extending ns in Otcl
  source your changes in your simulation scripts

- Extending ns in C++
  - Change Makefile (if created new files)
  - make depend
  - recompile
  - Makefile is in the ”ns-2.29” directory
Adding New Class

otcl

bind()

TclClass()

C++

variables   procedures

New Class

variables   procedures

New Class

© Ke Liu, SUNY-Binghamton, 2006
C++ Code Architecture of a Mobile Node (DSDV)
A component at some layer of each node

- Up target: except the highest layer component
- Down target: even the lowest layer component
- Timers: triggering some actions
- Send: either to up or down neighbor
-Recv: from either up or down neighbor
Assignments

• Finish the following sections of the tutorial on link http://nile.wpi.edu/NS/
  – Purpose
  – Overview
  – Basics
  – Post Simulation
  – Extending NS: where to Find What?

• Understand how to simulate the DSR and DSDV routing with ns2. You may use the NS2 manual.


© Ke Liu, SUNY-Binghamton, 2006
Project Assignment 1: requirement

- Wireless simulation: using **BOTH** DSDV and DSR routing;
- You should trace **Agent, Routing Agent, Mac**;
- You DON’T need to create nams trace;
- Create 9 **CBR** flows upon **UDP** transportation to one node.
- CBR Packet Size: 1024 Bytes, interval 1.0 sec, Simulation(Traffic) Time: 120 sec;
- No other parameter’s default value should be changed;
- All 9 flows start at the same time, so as end.
How to code a new Routing Agent
Steps

- Packet header design
- Packet header globalizing
- Routing Agent design
- Timer Design
- Tcl script commands Design
- Tcl linkage
Thank You!

My email: kliu@cs.binghamton.edu