NS-2

Introduction to the network simulator 2
What NS-2 is not

• A replacement for lab experiments
• Fast – runtime
• Bug free
• Well written software
• Without opponents
• Alone
  – NS-3 – Written with Python bindings, fewer bugs, fewer features as of now...
  – OPNET – A commercial product
  – GTNetS – Open Source
What NS-2 is

• A discrete-event network simulator
• Fast – development time
• Well accepted in the academic community
• Tested
• Different...
Highest Level

• Nodes (Routers and end Systems)
• Links
  – Most of the configuration is done here
• Traffic generators
  – FTP
• Protocols
  – CBR, UDP, TCP
Pragmatic usage

- Highly recommend usage on Linux
- Multi-core will not help, but lots of memory will
- Need working knowledge of Tcl (similar to bash)
- Zip your output, because there will be a lot...
  - set tracefile [open "| gzip > file_name.gz" w]
- Avoid using NAM
- Run a small test case before long runs
Tcl To get Started

• Comments
  - # beginning of line comment
  - …; # end of line comment

• Assign to variable
  - set stop 1000

• Read from variable
  - set end $stop

• Output to stdout
  - puts “Stopping at $stop”
Tcl Things to Know

• Learn these before your first NS-2 attempt.
  – Arrays – they save memory and speed things up
  – Basic file I/O
  – Flow control (if, else, for)
  – Expressions are very strange/verbose
  – Functions and Procedures
Pragmatic Usage

set ns [new Simulator]
set src [$ns node]
set rcv [$ns node]
$ns duplex-link $src $rcv 100Mb 10ms DropTail
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $src $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $rcv $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
Under the Hood

- Could be completely coded in an interpreted language, but what would be the problem?
- C++ is used for speed
- Tcl is used for flexibility
- The interaction between them can be confusing...
Under the Hood Continued...

• Things I have hacked...
  – TCP Congestion Control Algorithm
  – Timer resolution
  – Event Trace format

• With the exception of the timer resolution all of my modifications have been in a handful of files.
Your First Exp.

• We will create a framework and then fill in the details...
• Start simple (just the basics) then gradually add complexity. Skills like unit testing will come in handy here.
• I can’t beat this…please look at this for the basics...

http://nile.wpi.edu/NS/
Output

• Tracing – see chapter 26 in the NS-2 documentation
• Packet based tracing
  – +, -, r, d
  – headers
  – flags (see my decoding file)
• Periodic tracing
  – $ns$ monitor-queue $n1$ $n2$ file_name interval
• Event tracing – undocumented but very useful
  – See code example
Learn GNU Plot

• Department machines have old versions of gnuplot, and things don’t always port 100%
• Different “terminals” will have different options.
• There is a lot of style and preferences, but this is what works for me.
• There is some separation between commands and terminal types.
  – One style for working data so I can view the details
  – Another style for placing in papers or presentations
set terminal jpeg giant transparent size 400,300
set output "file_name.jpg"
set title "A nice description"
set xlabel "Value (unit)"
set ylabel "Value (unit)"
set yrange [0:100]
set xrange [0:1000]
set logy
set grid
plot "< zcat file_name1.gz" using 1:2 with lines t "1",
"< zcat file_name2.gz" using 1:2 with lines t "2",
"< zcat file_name3.gz" using 1:2 with lines t "3"
quit
Resources

• NS by Example – http://nile.wpi.edu/NS/
• Scripts and Files –
  – Data processing
  – Basic experiments
  – Flag keys
• NS-2 source
  – Use allinone – it is huge but it is worth the time savings
  – There is source available on the NS-2 Site but we will be using our local git repository
• gnuplot – http://www.gnuplot.info/