Performance Modeling of TCP and UDP over Packet Radio Networks using the ns-2 Network Simulator

Paul D. Wiedemeier, Ph.D., KE5LKY

The University of Louisiana at Monroe
Computer Science and CIS Department
College of Business Administration
700 University Avenue
Monroe, Louisiana, 71209

Presentation Outline

• Introduction
• Network Performance Modeling
• The ns-2 Network Simulator
• Writing Basic ns-2 Tcl Scripts
• Interpreting ns-2 Generated Trace Files
• Conclusions
Introduction

• A small percentage of the student body at many universities and colleges in the United States hold a F.C.C. amateur radio license.

• During this presentation we discuss how the ns-2 network simulator can be used to evaluate the performance of data communication protocols when used to transmit data over 2-meter amateur packet radio networks.

Network Performance Modeling

• Five methods are commonly used to evaluate the performance of data communication protocols, such as the TCP Reno and UPD.

1. Live (Network) Tests
2. Test-Beds
3. Hardware Emulation
4. Mathematical Models
5. Software Simulation
Live (Network) Tests

- The most comprehensive method for testing the performance of data communication protocols.

Live Network

- Advantage
  1. Real data is generated.

- Disadvantages
  1. Unable to assess the impact a new protocol has on other network traffic sharing the same communication channel.
  2. Often difficult to obtain access to private network environments for testing purposes.
Test-Beds

• Small isolated test-bed networks are often created and used to test data communication protocols because large networks contain numerous uncontrollable variables that can adversely affect protocol tests.

- Advantages
  1. They use actual hardware, but without the complexity associated with larger communication networks.
  2. Since they lack complexity, the impact associated with using a new protocol is easier to access.

- Disadvantages
  1. They are limited, in their complexity and speed, by the equipment on hand.
  2. If commercial operating systems are used, a researcher may be unable to modify proprietary protocol code.
Hardware Emulation

- A network emulator is a hardware based protocol evaluation tool.
  - A test-bed is constructed and a computer emulates the function of a specific piece of the communication network.

### Advantages

1. They test real protocol implementations.
2. The non-emulated routers, hubs, and switch protocols are easy to modify.

### Disadvantages

1. They simplify some of the modeled network’s real behavior.
2. They may not be able to represent complex or changing topologies.
3. A considerable amount of equipment may be required.
Mathematical Models

• Three equations can be used to mathematically model the performance of data communication protocols.

File Transmission Time = \( \frac{\text{File Size}}{\text{Channel Transmission Rate}} + \text{Channel Transmission Delay} \)

Channel Throughput = \( \frac{\text{File Size}}{\text{File Transmission Time}} \)

Channel Utilization = \( \frac{\text{Channel Throughput}}{\text{Channel Transmission Rate}} \)

Mathematical Models

• Hosts that transmit data using connection-oriented transport layer protocols, such as TCP, retransmit packets when an acknowledgement has not arrived within a specified time.

• Disadvantage
  – They do not account for channel congestion and packet corruption.

• Advantage
  – Computed file transmission times, channel throughputs, and channel utilizations are fairly close to actual data transmissions times generated by connectionless transport layer protocols, like UDP.
Software Simulation

- A simulator is a software based network evaluation tool that permits researchers to test data communication protocols when an actual network is unavailable.

Software Simulation

- Advantages

1. Extensive computer or networking equipment is not required when using a software simulator. Software simulators are often installed on personal computers (PCs) or workstations running the Microsoft Windows operating system or a UNIX/Linux operating system.

2. Researchers are able to execute numerous simulations in a short time period using software simulators. The tremendous amounts of data generated can be used to evaluate quickly the performance of data communication protocols.

3. Software simulators allow researchers to simulate communication networks they cannot access. Likewise, software simulators can simulate networks that do not yet exist due to current limitations (i.e. physical or monetary).
Software Simulation

- Advantages Continued

4. Complex communication networks, which cannot be created within a test-bed, are easily constructed and evaluated using software simulators.

5. All data generated during a simulation are maintained in files created by a software simulator. This feature permits researchers to identify easily which simulation variables produce significant impact.

6. Actual communication networks are not affected by software simulation.

Software Simulation

- Disadvantages

1. Rather than using actual data communication protocol code found in real operating systems, some software simulators use abstract implementations.

2. Software simulators often do not represent non-network events, such as operating system scheduler latency.

3. Software simulators often make assumptions concerning real world events, such as competing traffic.
The ns-2 Network Simulator

• Many network simulators exist and are used in business, government, and academia.

• We use the discrete event network simulator ns-2, version ns-2.31, because it supports simulation of TCP and UCP over wired and wireless networks.

The ns-2 Network Simulator

• The ns-2 project began in 1989.
  – A variant of the REAL network simulator.

• Today, ns-2 development is guided by …
  – DARPA
    – University of Southern California’s Information Sciences Institute
    – University of California at Berkeley’s International Computer Science Institute’s Center for Internet Research.
The ns-2 Network Simulator

- Ns-2 is written using two programming languages.
  - C++ and OTcl
  - Referred to as a “split-language” program

The ns-2 Network Simulator

- The C++ programming language implements the simulator's Internet protocols.
  - Used for processing at the packet level because compilation generates a fast, detailed, and complete binary.

- The OTcl programming language implements the simulator's interface.
  - Used to setup and control the simulation because it is an interpreted language.
  - As such, instructions are easy to write and change.
The ns-2 Network Simulator

• The official source for the ns-2 network simulator distribution and associated documentation is the ns-2 web site.
  – http://www.isi.edu/nsnam/

• The ns-2 distribution can also be obtained from the SourceForge web site.
  – http://sourceforge.net/projects/nsnam/

The ns-2 Network Simulator

• The full ns-2 distribution is actually comprised of several software packages.

• The ns-2 distribution can be obtained via a single package and installed en-masse.

• Alternatively, the individual packages that comprise ns-2 can be downloaded individually and installed separately.
The ns-2 Network Simulator

- Ns-2 was developed to run on top of a wide range of hardware, including PCs and workstations, and a variety of operating systems, including FreeBSD, Linux, Microsoft Windows, and Sun Microsystems SunOS/Solaris.

- A C++ compiler is primary tool required to build and install the ns-2 network simulator.
  - The UNIX/Linux OS’s often have a c++/gcc compiler preinstalled.
  - To install ns-2 on Microsoft Windows the Cygwin or MinGW C++ compilers are required.

The ns-2 Network Simulator

- We currently have ns-2 installed on a Dell Precision M90 laptop running Fedora Linux, version 2.6.20-1.2963.fc6.

- Travis Williamson, an ULM undergraduate CS student, is attempting to install ns-2 on several Dell Optiplex 745 desktops running Microsoft Windows XP Pro. using Cygwin.
Writing Basic ns-2 Tcl Scripts

• To simulate the transmission of data communication protocols using ns-2, one must write a ns-2 Tcl script.

  – “Tcl (Tool Command Language; and pronounced "tickle") is a scripting language created by John Ousterhout.” ♦

  – “It is most commonly used for rapid prototyping, scripted applications, GUls and testing.” ♦

  ♦ http://en.wikipedia.org/wiki/Tcl

• An ns-2 Tcl script is comprised of several sections.

• The number and type of sections included in a ns-2 Tcl script depends on whether you wish to simulate the transmission of data using …

  – Connection-oriented transport layer protocols (e.g. TCP)

  – Connection-less transport layer protocols (e.g. UDP)
Writing Basic ns-2 Tcl Scripts

- The ns-2 Tcl script sections
  1. Define finish() procedure
  2. Define variables
  3. Instantiate a ns-2 simulator object
  4. Define ns-2 and nam trace files
  5. Define all nodes
  6. Define links between nodes
  7. Define an error module (only for TCP transmissions)
  8. Define transport layer protocol values (e.g. packet size)
  9. Define color for data transmissions when viewed using nam
 10. Define the source and destination nodes
 11. Define the number of segment to transmit
 12. Define the start and stop times; Start the ns-2 simulation

The 2-meter packet radio network we simulated using ns-2 includes a sender, a receiver, and an interference device.

- When simulating data transmission using TCP Reno, the interference device is a ns-2 error module.

- When simulating data transmission using UDP, the interference device is non-existent.
The 2-meter packet radio network exhibits a transmission rate of 1200 bits per second (bps) and a 25 microsecond (μs) transmission delay.

We chose a 25 μs transmission delay for our simulations because, in the future, we intend to conduct live tests between transceivers located at the author’s home and work.

- The physical distance between these two locales is approximately 4.6 miles, which corresponds to a 25 μs transmission delay.
File Transmissions using FTP and TCP Reno

1 proc finish {} {
  2   global ns
  3   global filed
  4   $ns flush-trace
  5   close $filed(tr)
  6   close $filed(namtr)
  7   exit 0
  8 }
  9
  10 set opt(start_time) 0.0
  11 set opt(stop_time) 100000.0
  12 set opt(bw) 1200
  13 set opt(del) 0.00025
  14 set opt(error_unit) bit
  15 set opt(error_rate) 0.0
  16 set opt(ifq) DropTail
  17 set opt(file_size) 65536
  18 set opt(ack_size) 40
  19 set opt(max_win_size) 65536
  20 set opt(src) TCP/Reno
  21 set opt(sink) TCP/Sink
  22 set opt(app) FTP

  23 set ns [new Simulator]
  24 set filed(tr) [open "out.tr" w]
  25 $ns trace-all $filed(tr)
  26 set filed(namtr) [open "out.nam" w]
  27 $ns namtrace-all $filed(namtr)
  28 set node(0) [$ns node]
  29 set node(1) [$ns node]
  30 $ns duplex-link $node(0) $node(1) $opt(bw) $opt(del) $opt(ifq)
  31 $ns queue-limit $node(0) $node(1) 65536
  32 set error_model [new ErrorModel]
  33 $error_model unit $opt(error_unit)
  34 $error_model set_rate $opt(error_rate)
  35 $error_model ranvar [new RandomVariable/Uniform]
  36 $error_model drop-target [new Agent/Null]
  37 $error_model datapktsize $opt(sgmt_size)]
  38 $error_model cntrlpktsize $opt(ack_size)]
  39 $ns lossmodel $error_model $node(0) $node(1)
File Transmissions using FTP and TCP Reno

```
47 Agent/TCP set packetSize_ $opt(sgmt_size)
48 Agent/TCPSink set packetSize_ $opt(ack_size)
49 Agent/TCP set window_ [expr $opt(max_win_size) / $opt(sgmt_size)]
50 Agent/TCP set windowInit_ 1
51
52 set color(0) red
53 $ns color 0 $color(0)
54
55 set tcp0 [$ns create-connection $opt(src) $node(0) $opt(sink) $node(1) 0]
56 set ftp0 [$tcp0 attach-app $opt(app)]
57
58 set num_segs [expr $opt(file_size) / $opt(sgmt_size)]
59 set rem [expr $opt(file_size) % $opt(sgmt_size)]
60 if {$rem != 0} {set num_segs [expr $num_segs + 1]}
61 if {$num_segs == 0} {set num_segs 1}
62
63 $ns at $opt(start_time) "$ftp0 produce $num_segs"
64 $ns at $opt(stop_time) "finish"
65 $ns run
66 exit 0
```

Data Transmissions using CBR and UDP

```
1 proc finish () {
2    global ns
3    global filed
4    $ns flush-trace
5    close $filed(tr)
6    close $filed(namtr)
7    exit 0
8 }
9
10 set opt(start_time) 0.0
11 set opt(stop_time) 100000.0
12 set opt(bw) 1200
13 set opt(del) 0.00025
14 set opt(ifq) DropTail
15 set opt(file_size) 65536
16 set opt(sgmt_size) [expr [expr [expr $opt(sgmt_size) * 8.0] / $opt(bw)] + $opt(del)]
17 set opt(sgmt_interval) [expr [expr $opt(sgmt_size) * 8.0] / $opt(bw)] / $opt(del)
18 set opt(src) UDP
19 set opt(sink) Null
20 set opt(app) Traffic/CBR
```
Data Transmissions using CBR and UDP

```plaintext
set ns [new Simulator]
set filed(tr) [open "out.tr" w]
$ns trace-all $filed(tr)
set filed(namtr) [open "out.nam" w]
$ns namtrace-all $filed(namtr)
set node(0) [ns node]
set node(1) [ns node]
$ns duplex-link $node(0) $node(1) $opt(bw) $opt(del) $opt(ifq)
$ns queue-limit $node(0) $node(1) 65536
set color(0) red
$ns color 0 $color(0)
set udp0 [$ns create-connection $opt(src) $node(0) $opt(sink) $node(1) 0]
set cbr0 [$udp0 attach-app $opt(app)]
set opt(num sôngs) [expr [expr $opt(file_size) * 8.0] / [expr $opt(sgmnt_size) * 8.0]]
set opt(stop_time) [expr $opt(sgmnt_interval) * $opt(num sôngs)]
$ns at $opt(start_time) "$cbr0 start"
$ns at $opt(stop_time) "$cbr0 stop"
$ns exit 0
```
Interpreting ns-2 Generated Trace Files

- The ns-2 Tcl scripts shown on the previous slides create two types of ns-2 trace files.

  - Regular ns-2 trace files (i.e. `filed(tr)`)

  - Nam trace files (i.e. `filed(namtr)`)
Conclusions

- The ns-2 network simulator provides a means for packet radio enthusiasts to evaluate the performance of data transmission using TCP and UDP before creating an actual communication network.

- From an educational perspective, the ns-2 simulator permits students to simulate data transmission over 2-meter amateur packet radio networks before obtaining their F.C.C. amateur radio license.

Acknowledgements

- E. Benson Scott III, M.D., AE5V

- Allison M.D. Wiedemeier, Ph.D.

- The ULM Foundation and the Clarke M. Williams, Jr. Professorship in Computer Science

- The ULM Digital Communication Research Laboratory http://www.cs.ulm.edu/DCRL/
Questions?

Contact Information

Paul D. Wiedemeier, Ph.D, KE5LKY
ULM Clarke M. Williams, Jr. Endowed Professor of Computer Science
ULM Digital Communication Research Laboratory Director

Address: The University of Louisiana at Monroe
Computer Science and CIS Department
College of Business Administration
700 University Avenue
Administration Building, Room 2-37
Monroe, Louisiana, 71209

Phone: 318-342-1856
FAX: 318-342-1857
Email: wiedemeier AT ulm DOT edu or KE5LKY AT arrl DOT net
WWW: http://www.cs.ulm.edu/~pdw/
Office: Hemphill Airway and Computer Science Building, Room 348