Introduction to Network Simulator NS2

Teerawat Issariyakul • Ekram Hossain

Introduction to Network Simulator NS2

Springer

Teerawat Issariyakul
TOT Public Company Limited
89/2 Moo 3 Chaengwattana Rd.
Thungsonghong, Laksi
Bangkok, Thailand 10210
teerawas@tot.co.th
iteerawat@hotmail.com

Ekram Hossain
Department of Electrical &
Computer Engineering
University of Manitoba
75A Chancellor's Circle
Winnipeg MB R3T 5V6
Canada
ekram@ee.umanitoba.ca

ISBN: 978-0-387-71759-3 e-ISBN: 978-0-387-71760-9
DOI: 10.1007/978-0-387-71760-9

Library of Congress Control Number: 2008928147

© 2009 Springer Science+Business Media, LLC
All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

springer.com

To our families

Preface

NS2 is an open-source event-driven simulator designed specifically for research in computer communication networks. Since its inception in 1989, NS2 has continuously gained tremendous interest from industry, academia, and government. Having been under constant investigation and enhancement for years, NS2 now contains modules for numerous network components such as routing, transport layer protocol, application, etc. To investigate network performance, researchers can simply use an easy-to-use scripting language to configure a network, and observe results generated by NS2. Undoubtedly, NS2 has become the most widely used open source network simulator, and one of the most widely used network simulators.

Unfortunately, most research needs simulation modules which are beyond the scope of the built-in NS2 modules. Incorporating these modules into NS2 requires profound understanding of NS2 architecture. Currently, most NS2 beginners rely on online tutorials. Most of the available information mainly explains how to configure a network and collect results, but does not include sufficient information for building additional modules in NS2. Despite its details about NS2 modules, the formal documentation of NS2 is mainly written as a reference book, and does not provide much information for beginners. The lack of guidelines for extending NS2 is perhaps the greatest obstacle, which discourages numerous researchers from using NS2. At this moment, there is no guide book which can help the beginners understand the architecture of NS2 in depth.

The objective of this textbook is to act as a primer for NS2 beginners. The book provides information required to install NS2, run simple examples, modify the existing NS2 modules, and create as well as incorporate new modules into NS2. To this end, the details of several built-in NS2 modules are explained in a comprehensive manner.

NS2 by itself contains numerous modules. As time elapses, researchers keep developing new NS2 modules. This book does not include the details of all NS2 modules, but does so for selected modules necessary to understand the basics of NS2. For example, it leaves out the widely used modules such as wireless node or web caching. We believe that once the basics of NS2 are grasped, the readers can go through other documentations, and readily understand the details of other NS2 components. The details of Network AniMator (NAM) and Xgraph are also omitted here. We understand that these two tools are nice to have and could greatly facilitate simulation and analysis of computer networks. However, we believe that they are not essential to the understanding of the NS2 concept, and their information are widely available through most of the online tutorials.

This textbook can be used by researchers who need to use NS2 for communication network performance evaluation based on simulation. Also, it can be used as a reference textbook for laboratory works for a senior undergraduate level course or a graduate level course on telecommunication networks offered in Electrical and Computer Engineering and Computer Science Programs. Potential courses include "Network Simulation and Modeling", "Computer Networks", "Data Communications", "Wireless Communications and Networking", "Special Topics on Telecommunications". In a fifteen-class course, we suggest the first class for an introduction to programming (Appendix A), and other 14 classes for each of the 14 chapters. Alternately, the instructor may allocate 10 classes for teaching and 5 classes for term projects. In this case, we suggest that the materials presented in this book are taught in the following order: Chapters 1–2, 3, 12, 4–5, 6, 7–8, 9–11, 13 and 14. When the schedule is really tight, we suggest the readers to go through Chapters 2, 4–7, and 9–10. The readers may start by getting to know NS2 in Chapter 2, and learn the main concepts of NS2 in Chapters 4–5. Chapters 6–7 and 9–10 present the details of most widely used NS2 modules. From time to time, the readers may need to visit Chapter 3, 8, and 12 for further information. If tracing is required, the readers may also have to go through Chapter 13. Finally, Chapter 14 would be useful for those who need to extend NS2 beyond it scopes.

We recommend the readers who intend to go through the entire book to proceed chapter by chapter. A summary of all the chapters in this book is provided below.

As the opening chapter, **Chapter 1** gives an introduction to computer networks and network simulation. The emphasis is on event-driven simulation from which NS2 is developed.

An overview of Network Simulator 2 (NS2) is discussed in **Chapter 2**. Here, we briefly show the two-language NS2 architecture, NS2 directory and the conventions used in this book, and NS2 installation guidelines for UNIX and Windows systems. We also present a three-step simulation formulation as well as a simple example of NS2 simulation. Finally, we demonstrate how to use the `make` utility to incorporate new modules into NS2.

Chapter 3 explains the details of the NS2 two language structure, which consists of the following six main C++ classes: `Tcl`, `Instvar`, `TclObject`, `TclClass`, `TclCommand`, and `EmbeddedTcl`. **Chapters 4–5** present the very main simulation concept of NS2. While Chapter 4 explains implementation of event-driven simulation in NS2, Chapter 5 focuses on network objects as well as packet forwarding mechanism.

Chapters 6–11 present the following six most widely used NS2 modules. First, nodes (Chapter 6) act as routers and computer hosts. Secondly, links, particularly `SimpleLink` objects (Chapter 7), deliver packets from one network object to another. They model packet transmission time as well as packet buffering. Thirdly, packets (Chapter 8) contain necessary information in its header. Fourthly, agents (Chapters 9–10) are responsible for generating packets. NS2 has two main transport-layer agents: TCP and UDP agents. Finally, applications (Chapter 11) model the user demand for data transmission.

Chapter 12 presents three helper modules: timers, random number generators, and error models. It also discusses the concepts of two bit-wise operations, namely, bit masking and bit shifting, which are used throughout NS2.

Chapter 13 summarizes the post-simulation process, which consists of three main parts: debugging, variable and packet tracing, and result compilation.

After discussing all the NS components, **Chapter 14** demonstrates how a new module is developed and integrated into NS2 through two following examples: Automatic Repeat reQuest (ARQ) and packet schedulers.

Appendices A and B provide programming details which could be useful for the beginners. These details include an introduction to Tcl, OTcl, and AWK programming languages as well as a review of the polymorphism OOP concept.

As the final words, we would like to express sincere gratitude to our colleagues, especially, Surachai Chieochan, at the University of Manitoba, and the colleagues at TOT Public Company Limited, Bangkok, Thailand, for their continuous support. Last but not the least, we would like to acknowledge our families as well as our partners – Wannasorn and Rumana – for their incessant moral support and patient understanding throughout this endeavor.

TOT Public Company Limited *Teerawat Issariyakul*
University of Manitoba *Ekram Hossain*
July 2008

Contents

1 Simulation of Computer Networks 1
1.1 Computer Networks and the Layering Concept 1
1.1.1 Layering Concept 2
1.1.2 OSI and TCP/IP Reference Models 3
1.2 System Modeling 5
1.2.1 Analytical Approach 6
1.2.2 Simulation Approach 6
1.3 Basics of Computer Network Simulation 6
1.3.1 Simulation: The Formal Definition 7
1.3.2 Elements of Simulation 7
1.4 Time-Dependent Simulation 9
1.4.1 Time-Driven Simulation 9
1.4.2 Event-Driven Simulation 10
1.5 A Simulation Example: A Single-Channel Queuing System 12
1.6 Chapter Summary 18

2 Introduction to Network Simulator 2 (NS2) 19
2.1 Introduction 19
2.2 Basic Architecture 20
2.3 Installation 21
2.3.1 Installing an All-In-One NS2 Suite on Unix-Based Systems 22
2.3.2 Installing an All-In-One NS2 Suite on Windows-Based Systems 22
2.4 Directories and Convention 23
2.4.1 Directories 23
2.4.2 Convention 23
2.5 Running NS2 Simulation 26
2.5.1 NS2 Program Invocation 26
2.5.2 Main NS2 Simulation Steps 26
2.6 A Simulation Example 27

2.7 Including C++ Modules into NS2 and the *make* Utility 33
2.7.1 An Invocation of a Make Utility 33
2.7.2 A Make Descriptor File 33
2.7.3 NS2 Descriptor File 35
2.8 Chapter Summary 36

3 Linkage Between OTcl and C++ in NS2 37
3.1 The Two-Language Concept in NS2 38
3.2 Class Tcl 42
3.2.1 Obtain a Reference to the Tcl Instance 42
3.2.2 Invoking a Tcl Procedure 43
3.2.3 Pass or Receive Results to/from the Interpreter 44
3.2.4 Reporting Error and Quitting the Program 45
3.2.5 Retrieve the Reference to TclObjects 45
3.3 Class InstVar 46
3.3.1 Real and Integer Variables 47
3.3.2 Bandwidth 47
3.3.3 Time 48
3.3.4 Boolean 48
3.4 Class TclObject 49
3.4.1 Reference to a TclObject 50
3.4.2 Creating and Destroying a Shadow TclObject 50
3.4.3 Binding Variables in the Compiled and Interpreted Hierarchies 54
3.4.4 OTcl Commands 56
3.5 Class TclClass 61
3.5.1 An Overview of Class `TclClass` 61
3.5.2 TclObject Creation 62
3.5.3 Naming Convention for Class `TclClass` 62
3.5.4 Instantiation of Mapping Variables 63
3.6 Class TclCommand 63
3.6.1 Invoking a TclCommand 63
3.6.2 Creating a TclCommand 64
3.6.3 Defining Your Own TclCommand 65
3.7 Class EmbeddedTcl 66
3.8 Chapter Summary 67

4 Implementation of Discrete-Event Simulation in NS2 69
4.1 NS2 Simulation Concept 70
4.2 Events and Handlers 70
4.2.1 An Overview of Events and Handlers 70
4.2.2 Class `NsObject`: A Child Class of Class `Handler` 71
4.2.3 Classes `Packet` and `AtEvent`: Child Classes of Class `Event` 72
4.3 The Scheduler 74
4.3.1 Main Components of the Scheduler 74

4.3.2 Data Encapsulation and Polymorphism Concepts 75
4.3.3 Main Functions of the Scheduler 75
4.3.4 Dynamics of the Unique ID of an Event 77
4.3.5 Scheduling-Dispatching Mechanism 78
4.3.6 Null Event and Dummy Event Scheduling 78
4.4 The Simulator 80
4.4.1 Main Components of a Simulation 81
4.4.2 Retrieving the Instance of the Simulator 82
4.4.3 Simulator Initialization 83
4.4.4 Running Simulation 83
4.5 Instprocs of OTcl Class Simulator 84
4.6 Chapter Summary 84

5 Network Objects: Creation, Configuration, and Packet Forwarding 87
5.1 Overview of NS2 Components 87
5.1.1 Functionality-Based Classification of NS2 Modules 87
5.1.2 C++ Class Hierarchy 88
5.2 NsObjects: A Network Object Template 90
5.2.1 Class `NsObject` 90
5.2.2 Packet Forwarding Mechanism of NsObjects 91
5.3 Connectors 91
5.3.1 Class Declaration 92
5.3.2 OTcl Configuration Commands 93
5.3.3 Packet Forwarding Mechanism 96
5.4 Chapter Summary 98

6 Nodes as Routers or Computer Hosts 99
6.1 An Overview of Nodes in NS2 99
6.1.1 Architecture of a Node 99
6.1.2 Related Instproc of Class Node 101
6.1.3 Default Nodes and Node Configuration Interface 102
6.2 Routing Mechanism in NS2 103
6.3 Route Logic 105
6.4 Classifiers: Multi-target Packet Forwarders 107
6.4.1 Class `Classifier` and Its Main Components 107
6.4.2 Hash Classifiers 111
6.4.3 Port Classifiers 116
6.4.4 Installing Classifiers in a Node 116
6.5 Routing Modules 118
6.5.1 An Overview of Routing Modules 118
6.5.2 C++ Class `RoutingModule` 121
6.5.3 OTcl Class `RtModule` 124
6.5.4 C++ Class `BaseRoutingModule`and OTcl class `RtModule/Base` 125

6.6 Node Object Configuration 125
6.6.1 Relationship Among Instvars `module_list_`, `reg_module_`, `rtnotif_`, and `ptnotif_` 126
6.6.2 Adding/Deleting a Routing Entry 127
6.6.3 Agent Attachment/Detachment 127
6.6.4 Node Construction 128
6.6.5 Route Configuration 133
6.7 Chapter Summary 137

7 Link and Buffer Management 139
7.1 Introduction to SimpleLink Objects 139
7.1.1 Main Components of a SimpleLink 139
7.1.2 Instprocs for Configuring a `SimpleLink` Object 140
7.1.3 The Constructor of Class `SimpleLink` 142
7.2 Modeling Packet Departure 143
7.2.1 Packet Departure Mechanism 143
7.2.2 C++ Class `LinkDelay` 144
7.3 Buffer Management 146
7.3.1 Class `PacketQueue`: A Model for Packet Buffering 147
7.3.2 Queue Handler 148
7.3.3 Queue Blocking and Callback Mechanism 149
7.3.4 Class `DropTail`: A Child Class of Class `Queue` 151
7.4 A Sample Two-Node Network 152
7.4.1 Network Construction 153
7.4.2 Packet Flow Mechanism 154
7.5 Chapter Summary 154

8 Packets, Packet Headers, and Header Format 157
8.1 An Overview of Packet Modeling Principle 157
8.1.1 Packet Architecture 157
8.1.2 A Packet as an Event: A Delayed Packet Reception Event 160
8.1.3 A Linked List of Packets 161
8.1.4 Free Packet List 161
8.2 Packet Allocation and Deallocation 163
8.2.1 Packet Allocation 163
8.2.2 Packet Deallocation 166
8.3 Packet Header 168
8.3.1 An Overview of First Level Packet Composition: Offseting Protocol Specific Header on the Packet Header 169
8.3.2 Common Packet Header 170
8.3.3 IP Packet Header 171
8.3.4 Packet Type 172
8.3.5 Protocol Specific Headers 174

8.3.6 Packet Header Access Mechanism 178
8.3.7 Packet Header Manager 180
8.3.8 Protocol Specific Header Composition and Packet Header Construction 183
8.4 Data Payload 187
8.5 Customizing Packets 191
8.5.1 Creating Your Own Packet 191
8.5.2 Activate/Deactivate a Protocol Specific Header 192
8.6 Chapter Summary 194

9 Transport Control Protocols Part 1 – An Overview and User Datagram Protocol implementation 197
9.1 UDP and TCP Basics 197
9.1.1 UDP Basics 197
9.1.2 TCP Basics 198
9.2 Basic Agents 203
9.2.1 Applications, Agents, and a Low-level Network 203
9.2.2 Agent Configuration 205
9.2.3 Internal Mechanism for Agents 206
9.2.4 Guidelines to Define a New Transport Layer Agent ... 210
9.3 UDP (User Datagram Protocol) and Null Agents 210
9.3.1 Null (Receiving) Agents 211
9.3.2 UDP (Sending) Agent 211
9.3.3 Setting Up a UDP Connection 215
9.4 Chapter Summary 215

10 Transport Control Protocols Part 2 – Transmission Control Protocol (TCP) 217
10.1 An Overview of TCP Agents in NS2 217
10.1.1 Setting Up a TCP Connection 217
10.1.2 Packet Transmission and Acknowledgment Mechanism 218
10.1.3 TCP Header 219
10.1.4 Defining TCP Sender and Receiver 219
10.2 TCP Receiver 222
10.2.1 Class `Acker` 224
10.2.2 Class `TcpSink` 228
10.3 TCP Sender 230
10.4 TCP Packet Transmission Functions 230
10.4.1 Function `sendmsg(nbytes)` 231
10.4.2 Function `send_much(force,reason,maxburst)` 232
10.4.3 Function `output(seqno,reason)` 234
10.4.4 Function `send_one()` 236
10.5 ACK Processing Functions 237
10.5.1 Function `recv(p,h)` 238
10.5.2 Function `recv_newack_helper(pkt)` 239

10.5.3 Function `newack(pkt)` 241
10.6 Timer Related Functions 242
10.6.1 RTT Sample Collection 242
10.6.2 RTT Estimation 244
10.6.3 Overview of State Variables 244
10.6.4 Retransmission Timer 246
10.6.5 Function Overview 247
10.6.6 Function `rtt_update(tao)` 248
10.6.7 Function `rtt_timeout()` 250
10.6.8 Function `rtt_backoff()` 251
10.6.9 Function `set_rtx_timer()` and Function `reset_rtx_timer(mild,backoff)` 252
10.6.10 Function `newtimer(pkt)` 253
10.6.11 Function `timeout(tno)` 253
10.7 Window Adjustment Functions 255
10.7.1 Function `opencwnd()` 256
10.7.2 Function `slowdown(how)` 256
10.8 Chapter Summary 259

11 Application: User Demand Indicator 261
11.1 Relationship Between an Application and a Transport Layer Agent 261
11.2 Details of Class Application 265
11.2.1 Functions of Classes `Application` and `Agent` 265
11.2.2 Public Functions of Class `Application` 266
11.2.3 Related Public Functions of Class `Agent` 267
11.2.4 OTcl Commands of Class `Application` 268
11.3 Traffic Generators 268
11.3.1 An Overview of Class `TrafficGenerator` 268
11.3.2 Main Mechanism of a Traffic Generator 270
11.3.3 Built-in Traffic Generators in NS2 272
11.3.4 Class `CBR_Traffic`: An Example Traffic Generator 275
11.4 Simulated Applications 277
11.4.1 FTP (File Transfer Protocol) 278
11.4.2 Telnet 278
11.5 Chapter Summary 279

12 Related Helper Classes 281
12.1 Timers 281
12.1.1 Implementation Concept of Timer in NS2 281
12.1.2 OTcl Implementation 283
12.1.3 C++ Based Class Implementation 285
12.1.4 Guidelines for Implementing Timers in NS2 295
12.2 Implementation of Random Numbers in NS2 296
12.2.1 Random Number Generation 296
12.2.2 Seeding a Random Number Generator 296

12.2.3 OTcl and C++ Implementation 298
12.2.4 Randomness in Simulation Scenarios 300
12.2.5 Random Variables 303
12.2.6 Guidelines for Random Number Generation in NS2 . . . 306
12.3 Built-in Error Models 307
12.3.1 OTcl Implementation: Error Model Configuration 309
12.3.2 C++ Implementation: Error Model Simulation 312
12.3.3 Guidelines for Implementing a New Error Model in NS2 320
12.4 Bit Operations in NS2 321
12.4.1 Bit Masking 321
12.4.2 Bit Shifting and Decimal Multiplication 323
12.5 Chapter Summary 324

13 Processing an NS2 Simulation: Debugging, Tracing, and Result Compilation 327
13.1 Debugging: A Process to Remove Programming Errors 327
13.1.1 Types of Programming Errors 327
13.1.2 Debugging Guidelines 329
13.2 Variable Tracing 332
13.2.1 Activation Process for Variable Tracing 332
13.2.2 Instvar Objects 334
13.2.3 `TracedVar` Objects 335
13.2.4 Tracers 335
13.2.5 Connections Among a TclObject, a `TracedVar` Object, a Tracer, and a Trace File 338
13.2.6 Trace File Format 339
13.3 Packet Tracing 341
13.3.1 OTcl Configuration Interfaces 342
13.3.2 C++ Main Packet Tracing Class Trace 347
13.3.3 C++ Helper `Class BaseTrace` 350
13.3.4 Various Types of Packet Tracing Objects 352
13.3.5 Packet Trace Format 355
13.4 Compilation of Simulation Results 356
13.5 Chapter Summary 360

14 Developing New Modules for NS2 363
14.1 Automatic Repeat reQuest (ARQ) 363
14.1.1 The Design 364
14.1.2 C++ Implementation 365
14.1.3 OTcl Implementation 371
14.1.4 ARQ Under a Delayed (Error-Free) Feedback Channel 374
14.2 Packet Scheduling for Multi-Flow Data Transmission 376
14.2.1 The Design 376

14.2.2 C++ Implementation 377
14.2.3 OTcl Implementation 381
14.3 Chapter Summary 387

A Programming Essentials 389
A.1 Tcl Programming 389
A.1.1 Program Invocation 389
A.1.2 A Simple Example 390
A.1.3 Variables and Data Types 391
A.1.4 Input/Output 394
A.1.5 Mathematical Expressions 396
A.1.6 Control Structure 397
A.1.7 Procedures 399
A.2 Objected Oriented Tcl (OTcl) Programming 400
A.2.1 Class and Inheritance 400
A.2.2 Class Member Procedures and Variables 401
A.2.3 Object Construction and the Constructor 402
A.2.4 Related Instprocs 403
A.3 AWK Programming 405
A.3.1 Program Invocation 405
A.3.2 An AWK Script 406
A.3.3 AWK Programming Structure 407
A.3.4 Pattern Matching 407
A.3.5 Basic Actions: Operators and Output 408
A.3.6 Variables 410
A.3.7 Control Structure 411

B A Review of the Polymorphism Concept in OOP 413
B.1 Fundamentals of Polymorphism 413
B.2 Type Casting and Function Ambiguity 416
B.3 Virtual Functions 417
B.4 Abstract Classes and Pure Virtual Functions 418
B.5 Class Composition: An Application of Type Casting Polymorphism 419
B.6 Programming Polymorphism with No Type Casting: An Example 420
B.7 A Scalability Problem Caused by Non Type Casting Polymorphism 421
B.8 The Class Composition Programming Concept 422

References 427

General Index 429

Code Index 433