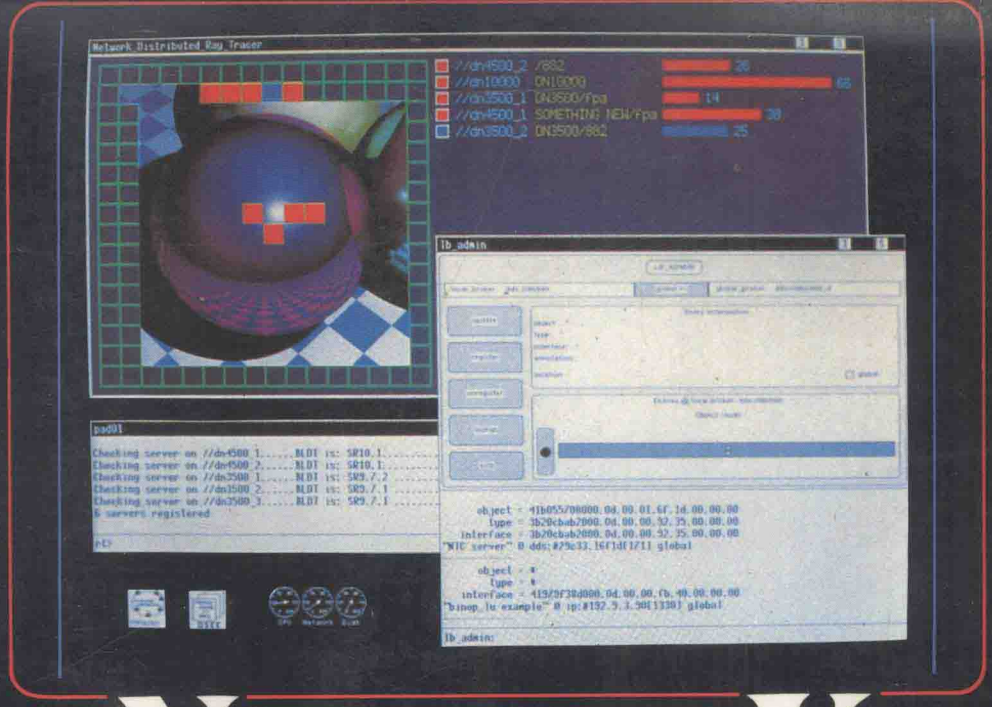

NETWORK COMPUTING SYSTEM TUTORIAL

TOM LYONS

Network Computing System Tutorial

Tom Lyons

Prentice Hall
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Editorial/production supervision: *Mary P. Rottino*
Cover design: *Lundgren Graphics*
Manufacturing buyer: *Kelly Behr/Susan Brunke*

Published by Prentice-Hall, Inc.
A division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

The publisher offers discounts on this book when ordered
in bulk quantities. For more information, write:

Special Sales/College Marketing
Prentice-Hall, Inc.
College Technical and Reference Division
Englewood Cliffs, NJ 07632

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-617242-3

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Preface

The *Network Computing System Tutorial* explains how to use Version 1.5.1 of the Network Computing System (NCS) to create heterogeneous distributed computing applications. We've organized this tutorial as follows:

Chapter 1	Discusses the basic concepts of using remote procedure calls in distributed computing and introduces the Network Computing Architecture (NCA) and the Network Computing System (NCS).
Chapter 2	Shows how to make a remote procedure call.
Chapter 3	Describes the basics of distributing an application.
Chapter 4	Explores some of the details of distributing an application.
Chapter 5	Catalogs NCS tools for distributed application development.
Chapter 6	Demonstrates the design of a distributed application.
Chapter 7	Demonstrates how to revise and extend a distributed application.
Chapter 8	Demonstrates some recovery techniques.
Chapter 9	Describes how to use the NCS object-oriented binding model.

There is a glossary and an index at the end of the book.

About this Book

The first half of the book treats NCS as a conventional RPC system, a system that extends the procedure call mechanism over two machines connected by a network. Distributed applications are examined as extensions of conventional applications, with an emphasis on the features of NCS that make RPCs look like local procedure calls at the network interface. By the end of Chapter 3, nearly all of the ideas underlying NCS have been introduced. The features added in Chapter 4 improve the robustness and portability of applications, but

don't add anything fundamental to the discussion in the first three chapters. The first four chapters include nearly all of what normally characterizes commercial RPC systems today.

Chapter 5 is a survey of the tools NCS provides an application programmer. It reviews the Network Interface Definition Language (NIDL), the application library calls, and the NCS exception model. It also introduces NIDL features and NCS calls not encountered in previous examples, and describes some of the NCS support software. It thus forms a supplement to, but not a substitute for, the *Network Computing System Reference Manual.* By systematically introducing NIDL attributes, open arrays, open structures, and exception handling, Chapter 5 simplifies the presentation of advanced topics in the second half of the book.

The second part of the book emphasizes the differences between distributed computing and traditional single-process computing. NCS allows you to exploit those differences where they are useful to your application and hide them where they are not. It was designed to facilitate the building of reliable, large-scale, long-lived, portable, and extensible distributed systems. Whereas the first part of the book starts from the premise that you have a conventional local application that would benefit from being distributed, and goes on to show how to transparently "remote" an interface between two modules, the premise of the second part is that you want to create a new distributed application.

Chapter 6 begins with some suggestions for partitioning an application and designing good network interfaces. These suggestions are illustrated by creating a primitive **name server,** a network server program that maps names onto unique identifiers, and building an application around it. In Chapters 7 and 8, issues of maintenance, administration, and partial failure in distributed systems are identified in the process of extending the application of Chapter 6.

Chapter 9 introduces the object-oriented features of NCS: the use of UUIDs as invariant object identifiers, location transparency, and type-dependent binding of procedures. Object-orientation is central to the Network Computing Architecture (NCA) and NCS, but the casting of distributed computing in object-oriented terms can make both subjects harder to learn. That is why we've postponed the discussion of object-oriented distributed computing until the end of the book. Object-oriented solutions are a good fit to the problems of distributed computing, and we hope that Chapter 9 will help you take advantage of the object-oriented features of NCS. The *Network Computing System Reference Manual* includes more material on object-oriented programming with NCS.

Using this Book

We assume you have built applications in the C programming language. We don't assume you have any experience with networking software or remote procedure call (RPC) systems. The concepts of RPC and client/server computing as they apply to NCS are presented in Chapter 1. The networking basics you will need to understand NCS and use it effectively are presented in Chapter 2.

NCS can be used with a wide variety of programming languages, operating systems, and hardware architectures. However, it is impossible to cover in this book the full range of programming and operating environments supported by NCS, so we've had to make a few choices.

The sample programs in this book are written in C. The C language used in this book is basically that documented in the first edition of *The C Programming Language* by Kernighan and Ritchie (Prentice-Hall, 1978), with the addition of the **void** type specifier and structure assignment, but the programs are compatible with ANSI compilers as well.

We've tried to keep the examples portable. Unless otherwise noted, they should compile and run on any system that NCS runs on. Except where noted, the sample programs are demonstrated in the text by compiling and running them on an Apollo workstation under the Bourne shell but, aside from some differences noted in the text, the output should be virtually identical on any other UNIX* system.

If you are not using NCS on a UNIX system, you should know that the Bourne shell is a UNIX command line interpreter, which uses the dollar sign (**$**) as a standard prompt. When a compiler command is too long to fit on a single line, we've divided the command onto multiple lines with an escaped newline (a backslash, \, followed by RETURN) to continue it on the next line. By default, the Bourne shell issues the secondary prompt > (or >> on some systems) when expecting another line to complete a command. Thus the following three lines are a single compiler command to link nine object modules into a single executable. The second dollar sign prompt (**$**) indicates that the command has completed.

```
$ cc -o dbd12a dbd12a.o db12.o dba.o \
> db1_sstub.o db2_sstub.o dba_sstub.o \
> dba_cstub.o dba_cswtch.o util.o -lnck
$
```

Pathnames and command names shown in the text are correct for most UNIX systems. The details of using NCS on various systems are described in the *Network Computing System Reference Manual*. You should also consult the *Release Document* for your NCS product to learn of any changes peculiar to your system.

Sample Programs

The examples in this book are available in machine-readable form from the publisher. The examples distribution contains all the software in this book, together with makefiles for building the programs on UNIX systems and some additional code needed to make the examples complete programs or applications that you can run and test. Even though this book does not contain listings of all the files in the distribution, all the software in the

* UNIX is a registered trademark of AT&T in the U.S.A. and other countries.

distribution that illuminates the subject of distributed programming with NCS is included in the figures of this book.

All the code in this book and in the examples distribution is in the public domain and may be available from other sources in addition to the publisher. If you have a copy of the sample programs, feel free to make it available to other readers.

Related Manuals

For more information on the Network Computing Architecture and Network Computing System, see the following documents, which we list with their Hewlett-Packard order numbers:

- *Network Computing Architecture* (010201-A01)

  This book specifies the Network Computing Architecture in enough technical detail to write a new implementation that is compatible with NCS. It is mainly intended for programmers producing a new implementation of the architecture or porting an existing implementation, such as Hewlett-Packard's NCS, to a new platform. You do not need to read *Network Computing Architecture* in order to use this book. Architectural concepts from the NCA will be introduced and explained as needed in the text.

- *Network Computing System Reference Manual* (D-10200-C)

  This book is a comprehensive programmer's reference for NCS. You will want to consult it whenever you have a question that is not answered in this manual, or want to learn about features not used for the examples in this book. We will often refer to the reference when there is more to a subject than can be discussed here.

- *Managing NCS Software* (D-11895-C)

  This book explains how to set up and administer NCS software, including the Global and Local Location Broker Daemons. It is recommended to administrators of networks running software based on NCS. Programmers should find the information in the Release Document accompanying the NCS product sufficient for setting up the system required to compile and run the examples in this book.

The Hewlett-Packard order number for the *Network Computing System Tutorial* is (D-18355-B).

The *Release Document* for each NCS product from Hewlett-Packard contains installation procedures, descriptions of new or changed features, and lists of known and fixed bugs.

Typographic Conventions

Unless otherwise noted in the text, this manual uses the following conventions.

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term. In interactive examples, characters that you type appear in bold.
user-supplied values	Italic words or characters in formats and command descriptions represent values that you must supply.
`output/source code`	Information that the system displays appears in this `typeface`. Examples of source code also appear in this typeface.
^C	This indicates a program interrupt generated from the keyboard. On most UNIX systems, you can generate an interrupt by typing C while holding down the CTRL key.
. . .	A vertical ellipsis means that irrelevant parts of a figure or example have been omitted.
——— ⊞ ———	This symbol indicates the end of a chapter or part of a manual.

Acknowledgements

I want to thank Mike Kong for the *Network Computing System Reference Manual*, and his early guidance in the writing of this book. Special thanks go to Kevin Ackley of Course Six, Inc. for his close reading of several drafts, and for convincing me to write Chapter 2. My thanks also to Nat Mishkin, Liza Martin, and Dale Labossiere of the NCS team at Hewlett–Packard for reviewing the text, correcting errors of fact and emphasis, and answering numerous questions. I also want to express deep appreciation to Lisa Zahn and Richard Curtis for recognizing the need for a tutorial, and allowing me the time to write one. My sincerest thanks to Marilyn Kotwal, the editor, for her unflagging attention to language and detail. And finally, thanks to the NCS team past and present for writing such useful and elegant software.

Tom Lyons

Contents

Figures