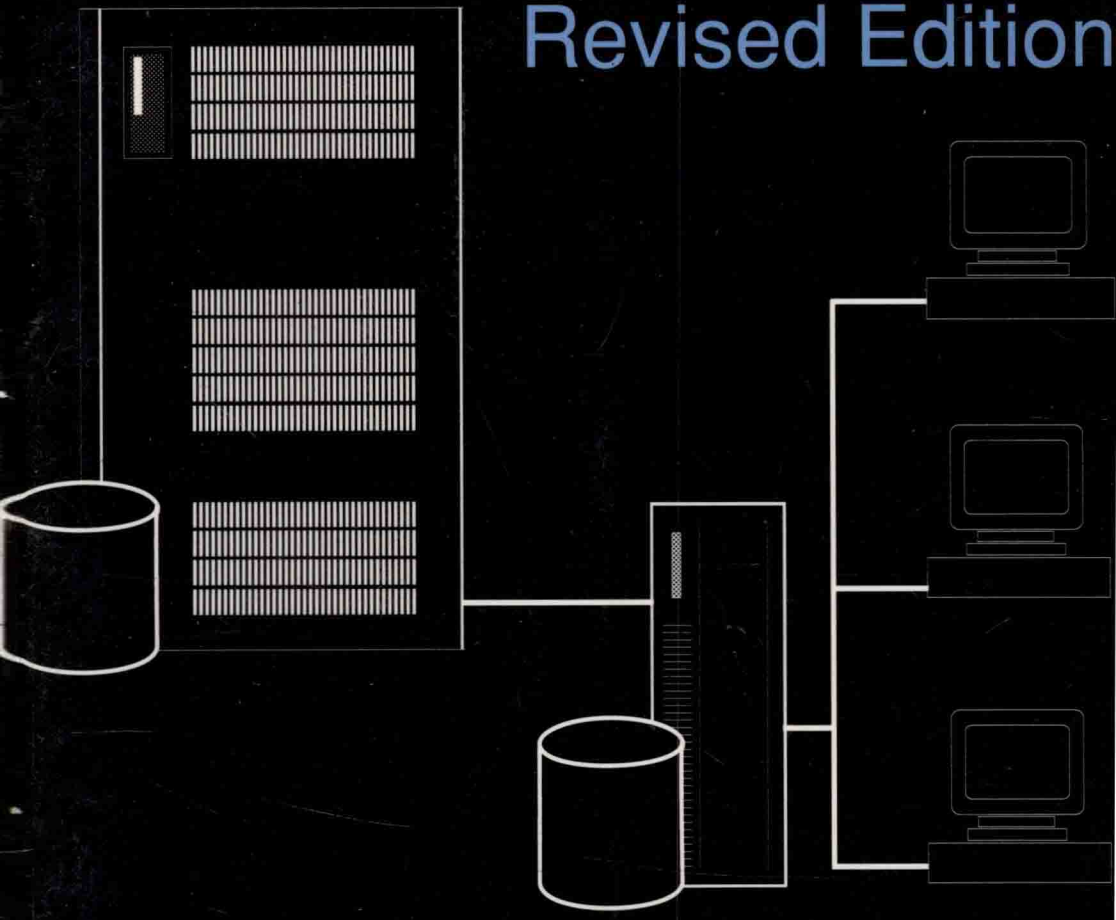
Developing Client/Server Applications

Revised Edition



W.H. Inmon

Developing Client/Server Applications

W.H. Inmon

A Wiley–QED Publication

John Wiley & Sons, Inc.

New York • Chichester • Brisbane • Toronto • Singapore

Library of Congress Cataloging-In-Publication Data

Inmon, William H.
Developing client/server applications / W.H. Inmon, Rev. ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-471-56906-2
1. Client/server computing. I. Title.
QA76.9.C55155 1993
005.2—dc20 93-12201
CIP

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Preface

A doctor can bury his mistakes, but an architect can only advise his clients to plant vines.

Frank Lloyd Wright

The client/server environment is seductive. Unlike the mainframe environment, the developer has complete control over the development and operational activities that occur in the machine in the client/server environment. Furthermore, the costs associated with hardware, software, and networks are so small (relative to other forms of processing) that a devil-may-care attitude pervades client/server processing. Because the developer has complete control over the processor and the costs of development and operations, it is easy to take a relaxed approach to the development process. The systems built for the client/server environment are often built in a lackadaisical, off-hand fashion. But such an attitude toward devel-

opment is patently dangerous. The size and scale of the environment and the opportunity to control all aspects of operations belie the mess that can be made. Simply stated, unless guided by architectural principles, designers in an autonomous environment such as the client/server world will create chaos. The larger the environment grows, the more chaotic it gets. Just as development in the mainframe environment is structured under a set of guidelines and principles, so the client/server environment should be governed. At least the client/server environment *ought* to be governed by underlying principles.

This book is about those underlying foundations that shape the client/server environment—what they are, how they are implemented, and what happens if they are ignored. Together those principles form an architecture that applies generically across all client/server environments. This book is *not* about any specific client/server technology. Instead, the principles discussed and the resulting architecture that is described apply to *all* technologies that support client/server processing.

The intent of this book is to arm the reader with practical solutions. Very little in the way of theory is presented. Upon reading this book, the reader will be prepared to build very sound, very stable client/server applications. Some of the germane topics to be discussed include

- performance in the client/server environment,
- control of update/"ownership"—"stewardship" of client/server data,
- the difference between operational and DSS processing in the client/server environment,
- application-by-application development versus integration,
- client/server processing and the data warehouse,

- metadata across the client/server environment,
- "requirements-driven development" versus "data-driven development" in the client/server environment,
- DSS processing in the client/server environment, and much more.

This book is for developers, programmers, database designers, managers, database administrators, data administrators, and designers. Students of computer science should also find this book to be of interest.

The author would like to express thanks to the following people for their support throughout this project:

Sheryl Larsen, Platinum Technology
Mark Gordon, Knowledgeware
Cheryl Estep, Chevron
Gary Noble, EPNG
Bill Pomeroy, AGS Consulting
Patti Mann, AMS
Claudia Imhoff, Connect
John Zachman, independent consultant
Sue Osterfelt, Storagetek
Ed Young, Prism Solutions
Ed Berkowitz, attorney/systems analyst

In addition, thanks to Melba Novak for her back-office support.

WHI

Contents

1

Architecture in the Client/Server Environment

Always design a thing by considering it in its next larger context—a chair in a room, a room in a house, a house in an environment, an environment in a city plan.

Eero Saarinen

A working man's definition of architecture is not having to tear up the street when the power lines are being laid. Architected properly, the power lines are laid first, then the street is built. Preparing for all major requirements, then building the structure in a sequence so that the parts fit together in a nondisruptive fashion is a good working definition of architecture.

Architecture in the client/server world is best understood in terms of a negative example, when things are not built in a nondisruptive fashion. Consider the following

sequence of events, which are entirely possible in the client/server environment:

1. A small program is built to service an auto parts "onboard" database on one node of the client/server environment.
2. The database is redesigned to include foreign auto parts as well as domestic auto parts.
3. Code is reconstructed to allow update to the database, as well as loading/access of the database.
4. Data is added to show flow of parts—transactions—as well as parts on board.
5. Code is changed and data is altered once again to allow requests to come from a client through the server.
6. In order to determine who can change data, code is added and data is changed to implement an "update control" feature, and so on.

After a few iterations of change to code and data, the system turns into a mess. By not anticipating requirements—by taking the stance that the system was simple and adding complexities incrementally—the developer has produced a maintenance and operational nightmare. Just because client/server systems are smaller than their mainframe cousins does not mean that they are any easier to build or maintain. In some regards client/server systems are more complex than their mainframe cousins. In the mainframe environment, for all its costs and complexities, there are certain basic services done automatically that have to be done individually in the client/server environment. For these reasons then, it is very important to take an architected approach to client/server processing and development. What then, are the shaping factors of a client/server architecture?

1. technology
   - the processors in the network
   - data in the network
   - the network itself
   - programs—systems and applications
2. costs
3. usage
   - DSS processing and operational processing
4. autonomy vs. integration.
5. organizational dynamics
6. structure of data

CLIENT/SERVER PROCESSING—THE BASICS

In its simplest form, a client/server architecture can be depicted as shown in Figure 1.1.

Figure 1.1 shows that there is a network connecting two pc's or workstations (called "nodes" in the network). Other pc's or workstations (i.e., nodes) most certainly will be connected by means of the network, although they are not shown in the simple picture depicted by Figure 1.1. The two pc's in the figure have a special relationship. Pc-B holds and manages data that can be accessed by Pc-A

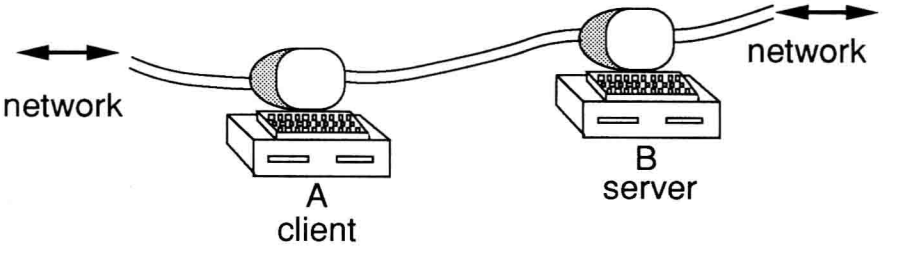


Figure 1.1 A simple perspective of the client/server environment.

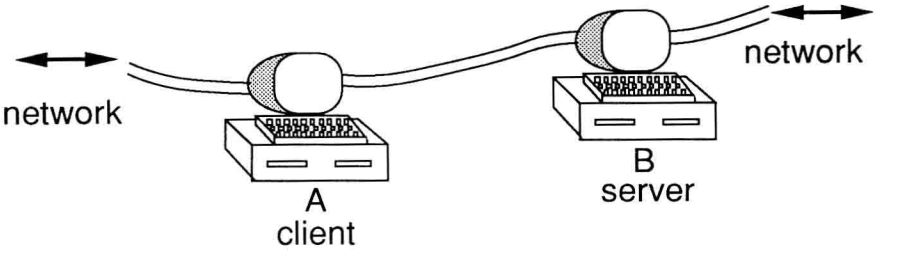


Figure 1.2 The different nodes of the network can be comprised of very different processors, such as pc's and workstations.

through the control of Pc-B. Pc-A is said to be the "client" and Pc-B is said to be the "server." In other words, Pc-B serves the needs of Pc-A. This simple configuration belies the complexity of the applications that are built in a client/server architecture. From a technical standpoint alone, there are many variations of the simple architecture shown in Figure 1.1. For example, Figure 1.2 shows a pc and a larger workstation linked together.

But disparity of processing size among nodes is not the only variable; another important variable is the network itself. Figure 1.3 shows an intermittent linkage between two nodes in the network.

A network may also be connected to another network, as shown by Figure 1.4.

The nodes of a network are organized into "rings." The ring consists of nodes that share the same network. Figure 1.5 shows a ring.

The speed of movement of data within a ring (under normal circumstances) is measured in terms of milliseconds.

Rings are connected to other rings by means of bridges. A bridge is nothing more than a device that controls the traffic from one ring to another. Figure 1.6 shows a bridge.

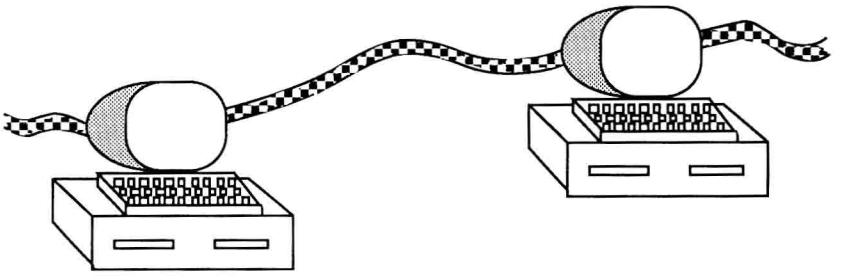


Figure 1.3 Not only can the technologies found at the nodes vary, but the technology itself may be based on a variety of technologies as well.

There are, then, *many* possibilities in the arrangement of nodes and networks in the client/server environment. But the major issue of client/server architecture does not center around the physical configuration of the nodes and the network. Instead, the major issues of client/server architecture center around the processing that occurs within the architecture and the data stored in and flowing

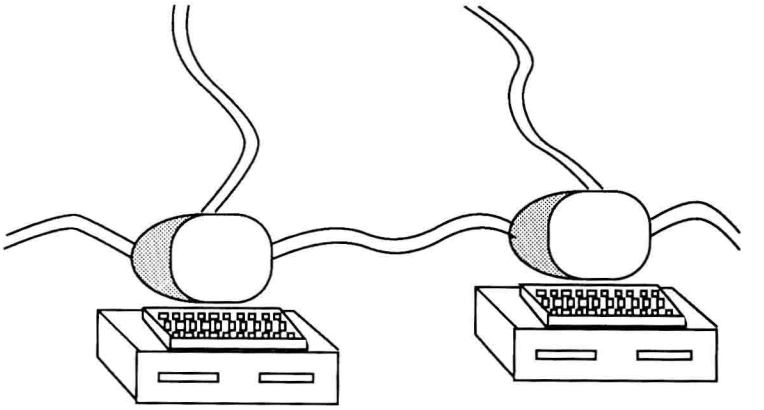


Figure 1.4 Multiply connected nodes to different networks.

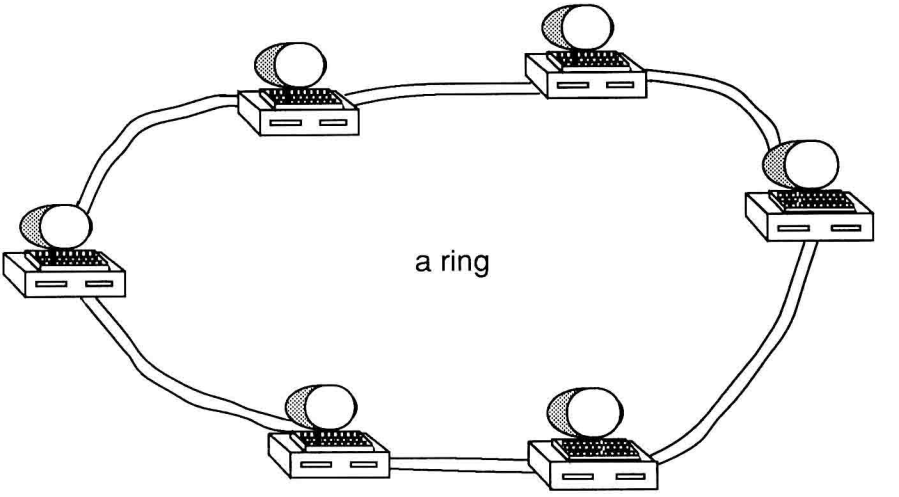


Figure 1.5 A "ring."

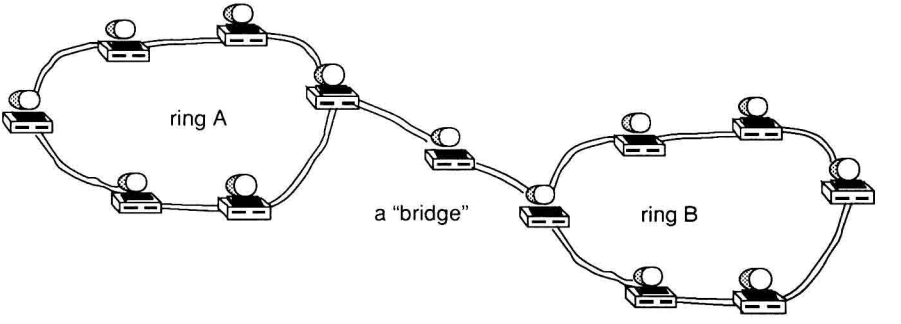


Figure 1.6 A bridge between rings A and B.

through the network itself. Focusing on the technology making up the client/server environment is an interesting thing to do. Certainly the developer/designer needs to be thoroughly grounded in the technology that the client/server environment will run on. But the proper focus of

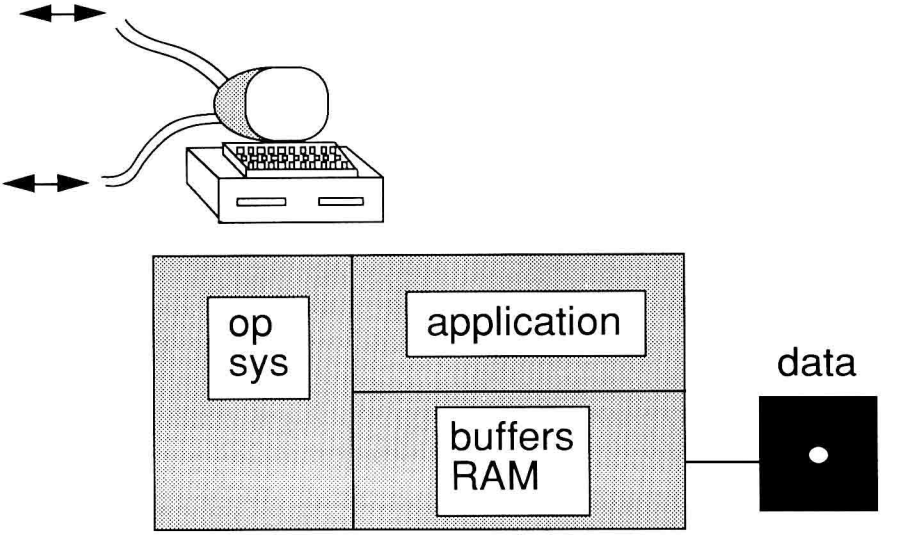


Figure 1.7 The components of each node in the network.

the developer should be not on technology, but on the application(s) being built and the larger structure of processing and data.

A simple arrangement of the important software components found within each node of the network is shown by Figure 1.7. Figure 1.7 shows that each node has an operating system, application programs, internal data, and external data. The external data may be on a wide variety of media, such as tape, floppy disk, etc. (The symbol for floppy disk will be used to represent *all* external media for data storage.) Each node is connected with other nodes by means of the network. Internodal communication is achieved by means of a layered protocol of packets of data. Some nodes have more powerful software components of one or more of these aspects than other nodes. But all nodes share these common characteristics.

One of the interesting questions is how does the client/