Michael Whittaker

Beej's Guide to Network Programming

Thoughts and Notes

beejsockets github

Contents

1 Intro **5**
1.1 Audience . . . 5
1.2 Platform and Compiler . . . 5
1.3 Official Homepage and Books For Sale . . . 5
1.4 Note for Solaris/SunOS Programmers . . . 5
1.5 Note for Windows Programmers . . . 5

2 What is a socket? **6**
2.1 Two Types of Internet Sockets . . . 6
2.2 Low level Nonsense and Network Theory . . . 6

3 IP Adresses, `structs`, and Data Munging **8**

4 Jumping from IPv4 to IPv6 **9**

5 System Calls or Bust **10**

6 Client-Server Background **11**

7 Slightly Advanced Techniques **12**

Listings

Preface

This document contains the notes, musings, and thoughts generated during my reading of "Beej's Guide to Network Programming" by Brian Hall. The notes were taken primarily to encourage a thorough reading of the book and to help me recall the most important tidbits from the book upon a rereading of my notes. I can imagine the notes may be helpful to more than just me, so I am making them publicly available. A network programming novice, I cannot guarantee my notes are entirely correct, or even sensical at times. If you ever encounter a mistake, please contact me at mjw297@cornell.edu.

Along with the notes, I've thrown together some source code and other resources. Some of the code is taken directly from the text while some is original. All notes, code, and resources can be found at the beejsockets github.

Enjoy!

1 Intro

This book will teach network programming!

1.1 Audience

This is a tutorial, not a reference, for novice programmers.

1.2 Platform and Compiler

Compiled using Gnu's gcc.

1.3 Official Homepage and Books For Sale

Visit http://beej.us/guide/bgnet and http://beej.us/guide/url/bgbuy.

1.4 Note for Solaris/SunOS Programmers

You have to additional work (see the book).

1.5 Note for Windows Programmers

Switch to Unix :P

2 What is a socket?

A *socket* is a way to speak to other programs using standard Unix file descriptors. Recall that everything in Unix is a file. All I/O is done by reading and writing to a file descriptor, an integer associated with an open file. The file, however, can be many things: a network connection, a FIFO, a pipe, a terminal, etc. If we want to communicate with another program over the Internet, we'll do it via a file descriptor.We get, read, and write sockets using the `socket()`, `send()`, and `recv()` system calls.

There are many different kinds of sockets. This book will deal with DARPA Internet sockets.

2.1 Two Types of Internet Sockets

There are two types of sockets: "Stream Sockets" and "Datagram Sockets", also known as `SOCK_STREAM` and `SOCK_DGRAM` respectively.

Stream sockets Stream sockets are reliable two-way connected communication streams. The order of sent messages are maintained and the messaging is guaranteed to be error-free.

Applications such as telnet and the HTTP protocol use stream sockets.

Stream sockets use the Transmission Control Protocol, TCP, to guarantee their reliability.

Datagram sockets Datagram sockets are unreliable and connectionless. If you send a message it may not arrive and it may not arrive in the correct order. The only guarantee is that if the message does arrive, the data inside will be error-free.

Datagram sockets are connectionless because unlike stream sockets, you don't have to maintain an open connection. They are typically used in applications where dropping a few packets here and there is not important.

tftp, dhcpcd, multiplayer games, audio streaming, and video conferencing, all can use datagram sockets. Some applications like tftp and dhcpcd need additional protocols on top of UDP to ensure the packets make it, but other applications like gaming will simply ignore dropped packets (e.g. lag).

You would use and unreliable protocol like UDP for speed!

2.2 Low level Nonsense and Network Theory

Data encapsulation is how networking works. Essentially, data is wrapped in various headers and sent out, such as in Figure 1. When the packet is received, hardware will strip the ethernet header. The kernel will strip the IP and UDP headers. A TFTP program will stip the TFTP header and manipulate the unencapsulated data.

Such encapsulation is used in the *Layered Network Model.*

- Application

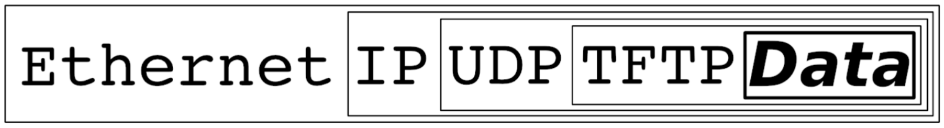

Figure 1: Data Encapsulation

- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

A model more consistent with Unix might be:

- Application Layer (telnet, ftp, etc.)
- Host-to-Host Transport Layer (TCP, UDP)
- Internet Layer (IP and routing)
- Network Access Layer (Ethernet, wi-fi, etc.)

3 IP Adresses, `structs`, and Data Munging

4 Jumping from IPv4 to IPv6

5 System Calls or Bust

6 Client-Server Background

7 Slightly Advanced Techniques