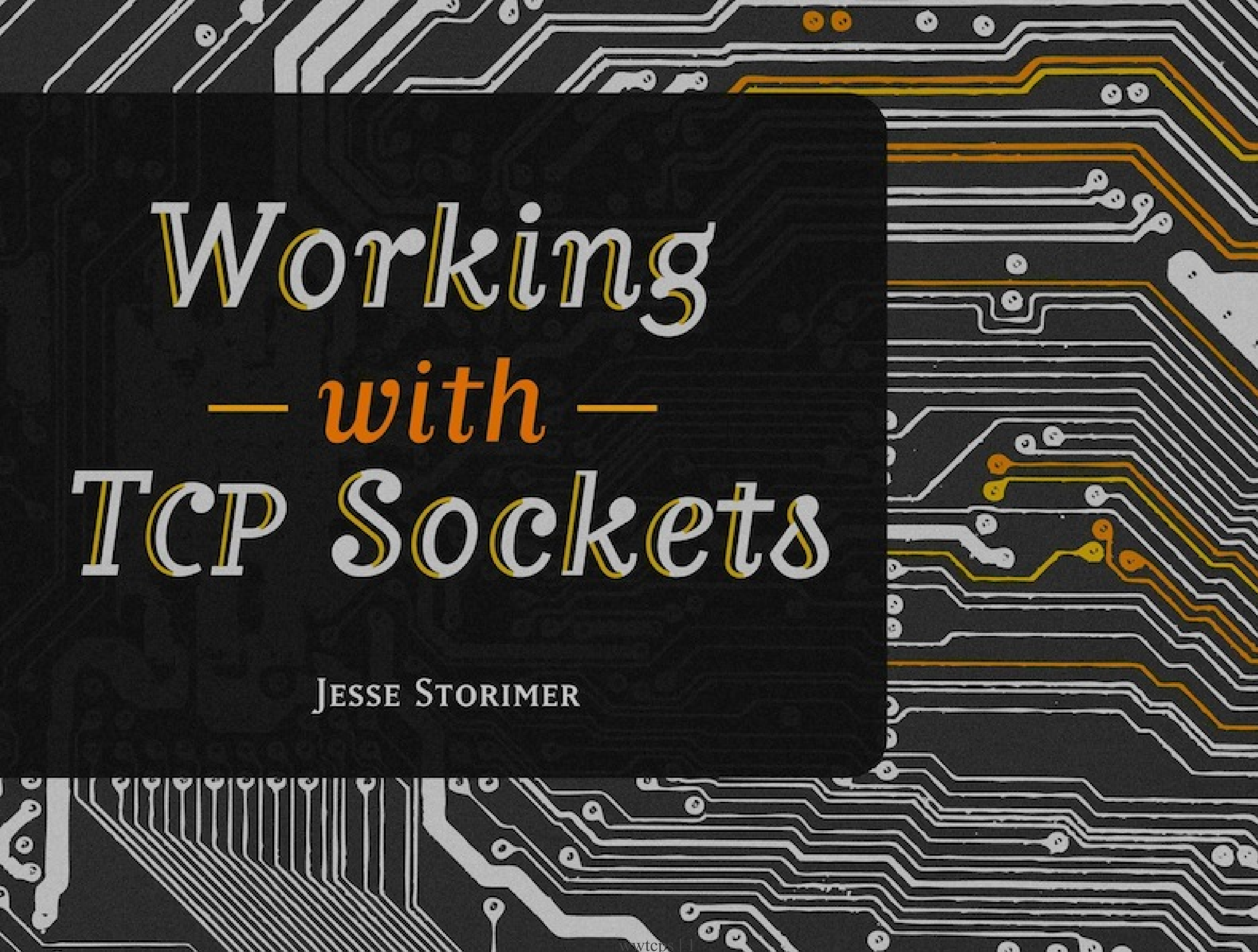

Working with TCP Sockets

Jesse Storimer

Copyright (C) 2012 Jesse Storimer.

Chapter 0

Introduction

Sockets connect the digital world.

Think for a minute about the early days of computing. Computers were something used exclusively by the scientists of the day. They were used for mathematical calculations, simulations; Real Serious Stuff™.

It was many years later when computers were able to connect people that the layperson became interested. Today, there are far more computers being used by laypeople than by scientists. Computers became interesting for this group when they could share information and communicate with anyone, anywhere.

It was network programming, and more specifically the proliferation of a particular socket programming API that allowed this to happen. Chances are, if you're reading this book, then you spend time every day connecting with people online and working with technology built on the idea of connecting computers together.

So network programming is ultimately about sharing and communication. This book exists so that you can better understand the underlying mechanisms of network programming and make better contributions to its cause.

My Story

I remember my first interaction with the world of sockets. It wasn't pretty.

As a web developer I had experience integrating with all kinds of HTTP APIs. I was accustomed to working with high-level concepts like REST & JSON.

Then I had to integrate with a domain registrar API.

I got a hold of the API documentation and was shocked. They wanted me to open a TCP socket on some private host name at some random port. This didn't work anything like the Twitter API!

Not only were they asking for a TCP socket, but they didn't encode data as JSON, or even XML. They had their own line protocol I had to adhere to. I had to send a very specifically formatted line of text over the socket, then send an empty line, then key-value pairs for the arguments, followed by two empty lines to show the request was done.

Then I had to read back a response in the same way. I was thinking "What in the...".

I showed this to a co-worker and he shared my trepidation. He had never worked with an API like this. He quickly warned me: "I've only ever used sockets in C. You have to be careful. Make sure you always close it before exiting otherwise it can stay open forever. They're hard to close once the program exits".

What?! Open forever? Protocols? Ports? I was flabbergasted.

Then another co-worker took a look and said "Really? You don't know how to work with sockets? You *do* know that you're opening a socket every time you read a web page, right? You should really know how this works."

I took that as a challenge. It was tough to wrap my head around the concepts at first, but I kept trying. I made lots of mistakes, but ultimately completed the integration. I think I'm a better programmer for it. It gave me a better understanding of the technology that my work depends upon. It's a good feeling.

With this book I hope to spare you some of that pain I felt when I was introduced to sockets, while still bringing you the sweet clarity that comes with having a deep understanding of your technology stack.

Who is This Book For?

The intended audience is Ruby developers on Unix or Unix-like systems.

The book assumes that you know Ruby and makes no attempts to teach Ruby basics. It assumes little to no knowledge of network programming concepts. It starts right at the fundamentals.

All of the example code is written using Ruby 1.9 and is not tested on earlier versions.

What to Expect

This book is divided into three main parts.

The first part gives an introduction to the primitives of socket programming. You'll learn how to create sockets, connect them together, and share data.

The second part of the book covers more advanced topics in socket programming. These are the kinds of things you'll need once you get past doing 'Hello world'-style socket programming.

The third part applies everything from the first two parts of the book in a 'real-world' scenario. This section goes past just sockets and shows you how to apply concurrency to your network programs. Several architecture patterns are implemented and compared to solve the same problem.

The Berkeley Sockets API

The main focus of this book will be the Berkeley Sockets API and its usage. The Berkeley Sockets API first appeared with version 4.2 of the BSD operating system in 1983. It was the first implementation of the then newly proposed Transport Control Protocol (TCP).

The Berkeley Sockets API has truly stood the test of time. The API that you'll work with in this book and the one supported in most modern programming languages is the same API that was revealed to the world in 1983.

Surely one key reason why the Berkeley Sockets API has stood the test of time: **You can use sockets without having to know the details of the underlying protocol**. This point is key and will get more attention later.

The Berkeley Sockets API is a programming API that operates at a level above the actually protocol implementation itself. It's concerned with stuff like connecting two endpoints and sharing data between them rather than marshalling packets and sequence numbering.

The de facto Berkeley Sockets API implementation is written in C, but almost any modern language written in C will include bindings to that lower-level interface. As such, there are many places in the book where I've gone to the effort of making the knowledge portable.

That is to say, rather than *just* showing the wrapper classes that Ruby offers around socket APIs I always start by showing the lower level API, followed by Ruby's wrapper classes. This keeps your knowledge portable.

When you're working in a language other than Ruby you'll still be able to apply the fundamentals you learn here and use the lower level constructs to build what you need.

What's Not Covered?

I mentioned in the last chapter that one of the strengths of the Berkeley Sockets API is that you don't need to know anything about the underlying protocol in order to use it. This book heartily embraces that.

Some other networking books focus on explaining the underlying protocol and its intricacies, even going as far as to re-implement TCP on top of another protocol like UDP or raw sockets. This book won't go there.

It will embrace the notion that the Berkeley Sockets API can be used without knowing the underlying protocol implementation. It will focus on how to use the API to do interesting things and will keep as much focus as possible on getting real work done.

However, there are times, when making performance optimizations, for example, when a lack of understanding of the underlying protocol will prevent you from using a feature properly. In these cases I'll yield and explain the necessary bits so that the concepts are understood.

Back to protocols. I've already said that TCP won't be covered in detail. The same is true for application protocols like HTTP, FTP, etc.. We'll look at some of these as examples, but not in detail.

If you're really interested in learning about the protocol itself I'd recommend Stevens' TCP/IP Illustrated [2].

netcat

There are several places in this book where the `netcat` tool is used to create arbitrary connections to test the various programs we're writing. `netcat` (usually `nc` in your terminal) is a Unix utility for creating arbitrary TCP (and UDP) connections and listens. It's a useful tool to have in your toolbox when working with sockets.

If you're on a Unix system it's likely already installed and you should have no issues with the examples.

Acknowledgements

First and foremost I have to thank my family: Sara and Inara. They didn't write the text, but they contributed in their own unique ways. From giving me the time and space to work on this, to reminding me what's important, if it weren't for them this book certainly wouldn't exist.

Next up are my awesome reviewers. These people read drafts of the book and together provided pages and pages of insights and comments that improved this book. Big thanks to Jonathan Rudenberg, Henrik Nyh, Cody Fauser, Julien Boyer, Joshua Wehner, Mike Perham, Camilo Lopez, Pat Shaughnessy, Trevor Bramble, Ryan LeCompte, Joe James, Michael Bernstein, Jesus Castello, and Pradeepto Bhattacharya.

2. http://www.amazon.com/TCP-Illustrated-Vol-Addison-Wesley-Professional/dp/0201633469