

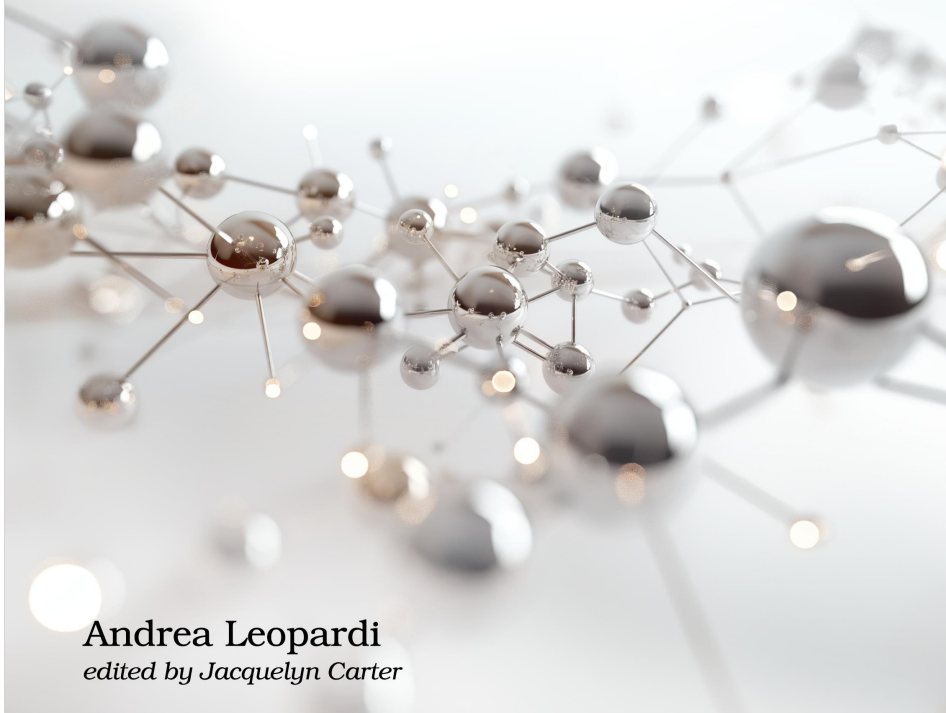
The  
Pragmatic  
Programmers



Your Elixir Source

# Network Programming in Elixir and Erlang

Write High-Performance, Scalable,  
and Reliable Apps with TCP and UDP



**Andrea Leopardi**  
*edited by Jacquelyn Carter*

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit <https://www.pragprog.com>.

Copyright © The Pragmatic Programmers, LLC.

# Introduction

---

Elixir and Erlang, and their shared ecosystems, are a fantastic choice for writing network code. After all, Erlang was invented to solve a specific network problem: routing and handling telephone calls. Even though networks have evolved since then, the underlying problems have remained the same. Real-time web applications have a lot in common with telephone calls—you have clients in different places that should ideally stay connected with each other and exchange data efficiently and reliably.

Erlang's standard library includes all the fundamental tools you need to work with network protocols and to build network applications. You'll learn about ecosystem libraries as well as the design patterns and best practices to use when writing this kind of software on the BEAM. Elixir's and Erlang's ecosystems also have plenty of third-party libraries that abstract away the most common network patterns and use cases (such as HTTP servers and clients).

I've been working in Elixir for more than ten years, mostly as part of its core team. I've never seen a platform this good at working with networks. Part of it, no doubt, is its telecoms heritage. But another part of it *has* to be the concurrency primitives and process-oriented design of the BEAM. With Elixir and Erlang, you're forced to think about the same problems you deal with when working with systems distributed across networks. You send data asynchronously to other processes, implement acknowledgments, and monitor other processes to detect failures. If all this sounds like networks, it's because it is: it involves exchanging data across nodes, request/response cycles, handling disconnections, and so on.

Hopefully, your journey through this book will make these similarities so obvious that you'll never stop seeing them. Let me tell a quick story about the origins of this project. I was learning another language at the time and tried to expand my knowledge of it by working on some networking challenges just for fun. None of the language (and ecosystem) abstractions matched

networks as well as the BEAM did. This made me appreciate Elixir and Erlang like never before. I want to share this feeling with as many folks as possible.

## Is This Book for You?

This book is mainly aimed at developers with Elixir and Erlang experience but without much experience in networks. It assumes a basic understanding of the syntax of these languages as well as of core concepts such as functions, processes, and message passing. You'll also need some understanding of OTP and its abstractions—such as GenServers and supervisors. But the book doesn't assume any network knowledge. Instead, it'll walk you through the basic principles of networks and network protocols from the ground up.

This book might also be a good fit for you if you are knowledgeable in networks but have no experience with Elixir or Erlang. You'll get a fantastic tool under your belt for working with networks. You might have to read up on Elixir- and Erlang-specific concepts here and there, but the book will show you how well these languages fit this domain.

## About This Book

This book is split up into three parts, each focusing on one of three important network protocols: TCP, UDP, and HTTP.

Let's take a more detailed look at the plan.

We start with an introduction to networks in [Chapter 1, What Is Network Programming Anyway?, on page ?](#). Learning how to use the tools that Elixir and Erlang provide requires some understanding of how networks work and the details of the protocols involved.

### Part I: TCP

The first part of the book is dedicated to TCP, the most widely used network protocol. [Chapter 2, TCP: Exploring the Basics, on page ?](#), introduces the protocol itself. The following two chapters, [Chapter 3, Designing a Chat Protocol and Its TCP Server, on page ?](#), and [Chapter 4, Scaling TCP on the Server Side, on page ?](#), focus on the server-side aspect of working with TCP. Then, [Chapter 5, Building TCP Clients, on page ?](#), and [Chapter 6, Scaling and Optimizing TCP Clients, on page ?](#), switch to the client side. The last chapter in this part, [Chapter 7, Securing Protocols: TLS, on page ?](#), is about securing network traffic over TCP by using TLS.

## Part II: UDP

In this second part, we explore UDP, a protocol that is quite different from TCP but in widespread use nonetheless. You'll start by learning the protocol basics in [Chapter 8, Same Layer, Different Protocol, on page ?](#). Then, you'll learn about techniques to increase the reliability of UDP with fine-grained control ([Chapter 9, Adding Guarantees to UDP, on page ?](#)). The last chapter in this part, [Chapter 10, UDP in the Wild: DNS, on page ?](#), is about DNS, the “Internet phone book” and a protocol mostly used on top of UDP.

## Part III: HTTP

In the last part of the book, we focus on HTTP, which is a protocol you'll most likely use directly at some point in your career (if not for most of it). In the first chapter, [Chapter 11, Talking the Internet Protocol: HTTP/1.1, on page ?](#), we explore HTTP/1.1 to build a foundational understanding of HTTP. Then, we explore HTTP/2—and even some HTTP/3!—in [Chapter 12, HTTP/2 and the Future, on page ?](#). Finally, we look at a protocol built on top of HTTP that powers many real-time interactions on the web: WebSockets ([Chapter 13, Communicating in Real Time with WebSockets, on page ?](#)).

## About the Code

The code in this book is mostly structured so that you can follow along and type it out yourself if that's your thing. But we'll sometimes just refer you to code that you can find in the online resources to avoid interrupting the flow too much.

All the example code is in Elixir, even though the book mentions Erlang in its title. We talk more about this in [Which One, Though: Elixir or Erlang?, on page ?](#). The concepts discussed in this book are not specific to either language, and the two languages share so much of their DNA that we believe a “translation guide” is enough for everyone to enjoy the content.

## Online Resources

The apps and example code in this book can be found at the Pragmatic Programmers website for this book.<sup>1</sup> You'll also find the errata submission form, where you can report problems with the text or make suggestions for future versions.

When you're ready, turn the page and we'll get started.

---

1. <https://pragprog.com/titles/alnpee/>