RECURSIVE DNS IN ZIG

BUILDING A HIGH-PERFORMANCE DNS RESOLVER FROM FIRST PRINCIPLES

- WIRE-FORMAT PARSING
- UDP, TCP & EDNS(0)
- CACHING & TTL MANAGEMENT
- DNSSEC & SECURITY
- CONCURRENCY & PERFORMANCE
- TESTING, FUZZING & DEPLOYMENT

IVAN GEBHART

Recursive DNS in Zig

Building a High-Performance DNS Resolver from First Principles

Steve T. Publications

This book is available at https://leanpub.com/recursivednsinzig

This version was published on 2026-07-15

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Building a High-Performance DNS Resolver from First Principles

Every time you visit a website, send an email, or stream a video, the Domain Name System translates human-readable names into machine addresses. DNS is the internet's most critical infrastructure, yet most developers never look beneath the surface of a single library call. This book changes that. You will build a complete, production-quality recursive DNS resolver in Zig from zero to deployment, learning every layer along the way: wire-format packet parsing, UDP and TCP transport, iterative resolution against root servers, caching with TTL management, EDNS(0) extensions, DNSSEC validation fundamentals, security hardening against cache poisoning and amplification attacks, lock-free concurrent data structures, event-driven I/O, performance profiling, fuzzing, and production deployment. By the end, you will have a fully working recursive resolver and a deep understanding of one of the internet's most important protocols.

Introduction: Why Build a DNS Server?

In 2008, Dan Kaminsky demonstrated that the Domain Name System was fundamentally broken. With nothing more than a laptop and a powerful enough network connection, he could poison the cache of any major DNS resolver on the internet, redirecting millions of users to sites of his choosing. The vulnerability was not a bug in a particular implementation; it was baked into the protocol itself. DNS had been designed in the 1980s for a trusted, small internet, and the world had outgrown its assumptions.

The coordinated fix that followed, documented in RFC 5452, introduced source port randomization and stronger transaction ID entropy. It made Kaminsky's attack effectively impossible without DNSSEC, the cryptographic extension designed to authenticate DNS responses end to end. But understanding why the attack worked, and how the fix addressed it, requires understanding DNS at the level of individual bytes on the wire.

That is what this book is about. Not wrapping a library. Not configuring an existing resolver. Building one from scratch in a language that forces you to think about every byte, every allocation, and every concurrent access pattern.

What You Will Build

Over the course of this book, you will construct a recursive DNS resolver capable of handling real-world workloads. It will:

- Parse and serialize DNS packets according to RFC 1035, including full domain name compression support
- Communicate with upstream servers over both UDP and TCP
- Perform iterative resolution starting from the root hints, following NS referrals through the hierarchy
- Cache responses with proper TTL management and negative caching for failed lookups
- Support EDNS(0) for larger packet sizes and DNSSEC awareness
- Validate the DNSSEC chain of trust from the root zone down

- Handle both IPv4 and IPv6 in dual-stack environments
- Serve thousands of concurrent queries using Zig's async I/O model
- Protect against cache poisoning, amplification attacks, and random subdomain abuse
- Expose structured logging, Prometheus-compatible metrics, and health check endpoints
- Ship with comprehensive unit tests, integration tests, fuzzers, and benchmarks

The code starts from an empty project and grows incrementally. Each chapter ends with a working executable that you can test against real DNS servers.

Why Zig?

Zig is a systems programming language designed for clarity, control, and correctness. It has no hidden control flow: there is no preprocessor, no implicit allocations, no exception handling, and no macro system. Every allocation is explicit and passes an allocator parameter. Every error is a value you handle with `try` or `catch`. Every memory access can be bounded-checked in debug mode.

For a DNS server, these properties matter enormously. A DNS resolver is a network-facing service that processes untrusted binary data at high throughput. Buffer overflows, use-after-free errors, and race conditions are not just bugs; they are attack vectors. Zig's type system and explicit error handling make many of these classes of errors impossible at compile time. Its comptime metaprogramming lets us generate efficient code without runtime overhead. And its zero-cost abstractions mean that the clean, readable code you write in Chapter 2 can be optimized to match the performance of hand-tuned C in Chapter 14.

Zig version 0.16.0, the stable release we target, introduces a new `std.Io` interface that unifies synchronous and asynchronous I/O behind a single abstraction. This means our resolver can use the same networking code whether it runs on a threaded backend or an event-driven backend like io_uring, without polluting the protocol logic with async/await annotations.

Prerequisites

You should be comfortable with Zig basics: variables, structs, error unions, slices, and the `std` library. You do not need prior experience implementing a DNS server or deep networking knowledge. The book explains every concept from first principles. If you have written network code in C, Rust, Go, or any other language, the patterns will feel familiar even if the syntax is new.

How to Use This Book

Read the chapters in order. Each one builds on the previous one, and the code accumulates. You can follow along by creating a Zig project and typing in the code as you read, or you can download the complete source for each chapter from the companion repository.

Every chapter includes exercises at the end to test your understanding, and a summary that ties the concepts together. The references section lists all RFCs, papers, and other sources cited throughout the book.

Let us begin with the protocol itself. Before we write a single line of code, we need to understand what DNS actually is, how its messages are structured, and how the resolution process works from the root of the hierarchy down to a final answer.

Chapter 1: The DNS Protocol from the Ground Up

Before we write a single line of Zig code, we need a thorough understanding of what DNS is, how its messages are structured on the wire, and how the resolution process works. This chapter covers the complete DNS protocol as defined in RFC 1034 and RFC 1035, along with key extensions that every modern resolver must support. By the end, you will understand exactly what bytes flow across the network when someone types `example.com` into their browser, and what our resolver needs to do to answer that query.

How DNS Powers the Internet

The Domain Name System is a distributed, hierarchical database that maps human-readable names to machine addresses and other associated data. When you type `www.example.com` into your browser, something must translate that string of characters into an IP address like `93.184.216.34`. DNS does that translation.

DNS is organized as a tree. At the top is the root zone, represented by a single dot (`.`). Below the root are the top-level domains (TLDs): `.com`, `.org`, `.net`, `.uk`, and so on. Below each TLD are second-level domains: `example.com`, `wikipedia.org`. And below those can be any number of subdomains: `www.example.com`, `mail.wikipedia.org`, `en.wikipedia.org`.

Each node in this tree is called a zone, and each zone is managed by one or more authoritative name servers. The root zone is managed by thirteen logical root server operators (labeled A through M) who run hundreds of physical servers around the world using anycast routing. The `.com` TLD is managed by Verisign. The `example.com` zone might be managed by the organization that registered the domain, or by a hosting provider on their behalf.

When a client wants to resolve a name, it sends a query to a recursive resolver. The resolver's job is to walk the hierarchy from the root down to the authoritative server for the requested name, collecting information along the

way, and return the final answer to the client. This walking process is called iterative resolution, and it is the core algorithm our book will implement.

The DNS protocol operates on port 53 using both UDP and TCP. UDP is preferred for most queries because it has less overhead: no three-way handshake, no connection state to maintain. TCP is required when responses exceed the UDP size limit (512 bytes without EDNS0, or up to 4096 bytes with EDNS0) and the server sets the truncation (TC) bit, signaling the client to retry over TCP.

The DNS Message Wire Format

Every DNS message, whether a query or a response, follows the same basic structure defined in RFC 1035 Section 4.1. It consists of a fixed 12-byte header followed by four variable-length sections:

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward lower level authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

The header is always exactly 12 bytes and contains the fields shown in this bit diagram:

```
                                    1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                      ID                       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    QDCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ANCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    NSCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ARCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Let us go through each field:

ID (16 bits): A random identifier assigned by the resolver to correlate queries with responses. When we send a query, we set this field; when we receive a response, we check that it matches. This is our first line of defense against cache poisoning attacks. We will use `std.crypto.random` to generate cryptographically strong 16-bit values.

QR (1 bit): Query or Response flag. Zero means query; one means response.

Opcode (4 bits): The kind of query. Zero is a standard query (QUERY), which is what we care about for recursive resolution. Other opcodes include IQUERY (inverse query, obsolete) and STATUS (server status request, also largely unused). Modern extensions define additional opcodes like UPDATE (1) for dynamic DNS updates (RFC 2136).

AA (1 bit): Authoritative Answer. Set by a name server to indicate that it is authoritative for the data in the answer section. Our resolver will never set this bit when responding to clients, because we are not an authoritative server; we are a recursive resolver. But we will check this bit when talking to upstream servers to understand whether their answers are definitive or referrals.

TC (1 bit): Truncation. Set by the server when the response does not fit in the available space (512 bytes for non-EDNS0 UDP). When our resolver receives a truncated response during iterative queries, it must retry over TCP. We will also set this bit when responding to clients if our response is too large for their advertised EDNS0 buffer size.

RD (1 bit): Recursion Desired. Set by the client to request that the server perform recursive resolution. When we query upstream servers during itera-

tive resolution, we typically leave this bit clear because we want the server to give us a referral or an answer, not to do our work for us. But when responding to clients who set RD, we indicate that we performed recursion on their behalf.

RA (1 bit): Recursion Available. Set by the server to indicate it supports recursive queries. We will set this in our responses to advertise that clients can ask us to recurse.

Z (3 bits): Reserved for future use. Must be zero in queries and ignored in responses, though some extensions have repurposed individual bits within this field.

RCODE (4 bits): Response code. The most common values are:

- 0 (NOERROR): The query completed successfully. There may or may not be data in the answer section.
- 1 (FORMERR): The server was unable to interpret the query due to a format error.
- 2 (SERVFAIL): The server encountered an internal failure and could not process the query.
- 3 (NXDOMAIN): The domain name does not exist. This is different from "no data" – it means the name itself is not in the zone.
- 5 (REFUSED): The server refused to answer the query, typically due to policy restrictions.

EDNS(0) extends RCODE to 12 bits (see RFC 6891), adding codes like 16 (YXDOMAIN), 17 (YXRRTYPE), and so on. We will handle extended RCODEs in Chapter 7.

After the header come four sections:

Question section: Contains the name being queried, the type of record requested, and the class (almost always IN for Internet). A standard query has exactly one question. The counts in the header (`QDCOUNT`, `ANCOUNT`, `NSCOUNT`, `ARCOUNT`) tell us how many resource records appear in each of the remaining three sections.

Answer section: Contains resource records that directly answer the question. For an A record query for `www.example.com`, this might contain an A record with the IPv4 address.

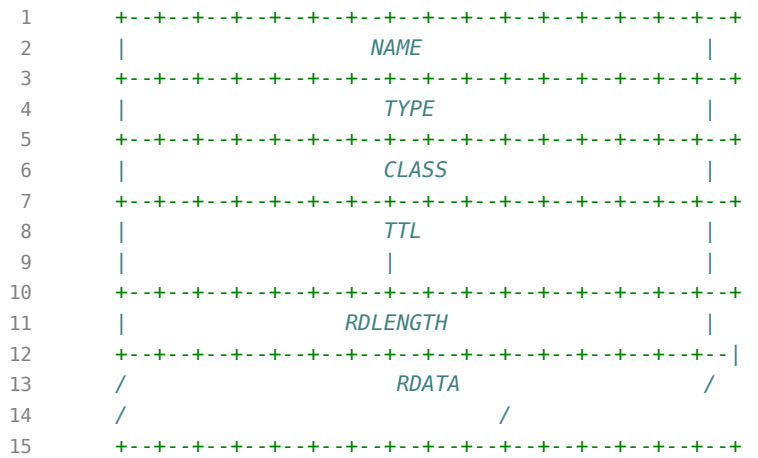
Authority section: Contains resource records that provide authoritative information, typically NS records pointing to lower-level name servers when

the responding server is not authoritative for the requested name. This is how referrals work.

Additional section: Contains helpful supplementary information, most commonly "glue" A or AAAA records that give the IP addresses of name servers mentioned in the authority section. Without glue records, resolving a domain would require an extra round trip for every referral: first get the NS names, then look up their addresses, then query them.

Resource Records and Their Types

Every resource record (RR) in the Answer, Authority, and Additional sections follows the same format defined in RFC 1035 Section 3.2.1:



NAME: The domain name to which this record applies. Uses the same compression mechanism as question names (we will cover this in detail in Chapter 2).

TYPE (16 bits): The kind of record. The most important types for our resolver are:

Type	Name	Description
1	A	IPv4 address (4 bytes)
2	NS	Name server hostname
5	CNAME	Canonical name (alias target)
6	SOA	Start of Authority (zone metadata)
12	PTR	Pointer (reverse DNS)
15	MX	Mail exchange (with priority)
28	AAAA	IPv6 address (16 bytes)
33	SRV	Service locator
41	OPT	EDNS(0) pseudo-record
43	DS	Delegation Signer (DNSSEC)
46	RRSIG	DNSSEC signature
47	NSEC	Next Secure (DNSSEC)
48	DNSKEY	DNSSEC key record

CLASS (16 bits): Almost always 1 (IN for Internet). Other classes like CS (CSNET), CH (Chaos), and HS (Hesiod) are historical curiosities that you will never encounter in practice.

TTL (32 bits): Time To Live in seconds. This tells the resolver how long it may cache this record before considering it stale. A TTL of 300 means "cache for 5 minutes." A TTL of 0 means "do not cache" (though some resolvers interpret this as a minimum of a few seconds). All values are transmitted in network byte order (big-endian).

RDLENGTH (16 bits): The length of the RDATA field in bytes. This allows parsers to skip over record types they do not understand.

RDATA: The variable-length data specific to each record type. For an A record, it is 4 bytes containing an IPv4 address. For a CNAME, it is a domain name. For an MX record, it is a 2-byte priority followed by a domain name.

The Resolution Journey: Recursive vs Iterative

The terms "recursive" and "iterative" in DNS refer to two different modes of query processing, and they are often confused because a recursive resolver uses iterative queries internally.

A **recursive query** is one where the client asks the resolver to do all the work. The client sends a single query with the RD (Recursion Desired) bit set, and the resolver returns either a final answer or an error. The client does not care how many upstream servers the resolver talks to; it just wants the answer. This is what happens when your browser asks `8.8.8.8` to resolve `www.example.com`.

An **iterative query** is one where the server gives the best answer it can, which might be a referral to another server rather than a final answer. When our resolver performs recursive resolution on behalf of a client, it sends iterative queries upstream: first to a root server, then to a TLD server, then to an authoritative server for the zone, following referrals at each step.

The full resolution journey for `www.example.com` looks like this:

1. Our resolver checks its cache. If it has a fresh A record for `www.example.com`, it returns it immediately and we are done.
2. If not cached, our resolver queries a root server (one of the 13 anycast-rooted logical servers) with an iterative query for `www.example.com` type A.
3. The root server does not know about `www.example.com`, but it knows about the `.com` TLD. It responds with a referral: NS records in the Authority section pointing to the `.com` TLD name servers (like `a.gtld-servers.net`), along with glue A/AAAA records in the Additional section giving their IP addresses.
4. Our resolver picks one of the `.com` TLD name servers and queries it iteratively for `www.example.com` type A.
5. The TLD server does not know about `www.example.com` directly, but it knows which servers are authoritative for `example.com`. It responds with NS records pointing to `ns1.example.com` and `ns2.example.com`, along with glue A/AAAA records.
6. Our resolver queries one of the authoritative servers for `example.com` iteratively for `www.example.com` type A.
7. The authoritative server knows the answer: it responds with an A record `93.184.216.34` in the Answer section, with the AA bit set.
8. Our resolver caches the answer (along with all the intermediate NS records and glue), and returns the A record to the client.

This journey involves at most four round trips (root, TLD, authoritative, and possibly one more for CNAME resolution). In practice, it is often fewer because

of caching: our resolver may already have cached the `.com` TLD nameservers or the `example.com` authoritative servers from previous queries.

Root Hints, Referrals, and Glue Records

Root hints are a small file (or embedded data structure) containing the IP addresses of the root name servers. Every recursive resolver must have root hints to bootstrap the resolution process. Without them, the resolver would not know where to start. The current root hints file is published by IANA at `https://www.internic.net/domain/named.root` and contains 13 logical root servers (A through M), each with one or more IPv4 addresses and typically one or more IPv6 addresses.

In our implementation, we will embed the root hints directly in the source code as a constant data structure. This eliminates a file dependency and ensures the resolver always has root hints available at startup. We will also support loading root hints from a configuration file for operators who want to update them without recompiling.

Referrals are the mechanism by which DNS implements its hierarchical delegation. When a name server receives a query for a domain it is not authoritative for, but knows about a closer authority, it responds with NS records in the Authority section naming the servers responsible for the next level of the hierarchy. The resolver extracts these NS names and queries them to continue the resolution chain.

Glue records are A or AAAA records placed in the Additional section alongside a referral. They provide the IP addresses of the name servers named in the referral, preventing an extra round trip. Glue records are required when the name server's hostname is within the zone it serves (a "child" NS record). For example, if `example.com` has NS records pointing to `ns1.example.com`, the TLD referral must include glue A/AAAA records for `ns1.example.com`; otherwise, resolving `ns1.example.com` would require querying the very server we are trying to reach, creating a circular dependency.

Our resolver must handle both cases: referrals with glue (use the glue addresses directly) and referrals without glue (resolve the NS hostnames recursively before querying them). The latter case adds complexity because it means following a referral can trigger additional recursive lookups.

CNAME Chains and Redirects

CNAME (Canonical Name) records create aliases. When a zone contains a CNAME record for `www.example.com` pointing to `cdn.example.net`, any query for `www.example.com` returns the CNAME, and the resolver must then look up `cdn.example.net` to find the actual answer.

RFC 1034 Section 3.6.2 states that no other data may exist for a CNAME owner name: if a zone has a CNAME for a name, it cannot have A, AAAA, MX, or any other record type for that same name. This simplifies the resolver's job: when we encounter a CNAME, we know we must follow it, and we do not need to check for other record types at that name.

CNAME chains can be arbitrarily long in theory, though RFC 2181 recommends that resolvers limit the chain to prevent infinite loops caused by misconfiguration. A common limit is 10 hops. Our resolver will implement a configurable CNAME hop limit with a default of 10, returning SERVFAIL if the limit is exceeded.

CNAME handling adds complexity to every stage of resolution:

- When we receive a cached answer that is a CNAME, we must check if we also have the final target cached; if not, we must resolve the target.
- When we receive a CNAME from an authoritative server during iterative resolution, we must follow it by querying for the target name at the same servers.
- When we return a response to a client, we may include both the CNAME chain and the final answer in the response, or we may return just the CNAME and let the client follow it (though most clients expect the resolver to do the following).

Chapter Summary

This chapter established the DNS protocol fundamentals that underpin everything we will build:

- The 12-byte header with its flags, counts, and transaction ID

- The four-section message structure (Question, Answer, Authority, Additional)
- Resource records with their NAME, TYPE, CLASS, TTL, RDLENGTH, and RDATA fields
- The resolution journey from root to authoritative server through iterative queries
- Root hints as the bootstrap mechanism
- Referrals and glue records as the delegation mechanism
- CNAME chains and the need for loop detection

In the next chapter, we will translate this knowledge into Zig code. We will create our project structure, define the data types that mirror the DNS wire format, implement a parser that reads raw bytes into structured data, and write a serializer that does the reverse. Every byte will be accounted for, every field validated, and every edge case tested.

Exercises

1. Using `dig` or `nslookup`, trace the resolution of `www.example.com` from root to answer. Count the number of round trips and note which servers are queried at each step.
2. Write down the complete 12-byte header (in hexadecimal) for a standard query with transaction ID `0xDEAD`, asking for an A record for `example.com`. Assume QDCOUNT=1, ANCOUNT=NSCOUNT=ARCOUNT=0.
3. Research RFC 2181 Section 6 and summarize the rules for negative caching. What is the difference between NXDOMAIN and NODATA, and how should each be cached?
4. Look up the current root hints file at `https://www.internic.net/domain/named.roo` How many IPv4 addresses are listed? How many IPv6 addresses? Which root servers have multiple IPv4 addresses?

Chapter 2: Your First DNS Packet in Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Project Setup and Build System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

The DNS Header Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Parsing Domain Names with Compression Pointers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Serializing DNS Packets Back to Wire Format

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

First Unit Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

The Entry Point

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cumulative Project State: End of Chapter 2

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 3: UDP Transport and Basic Query Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Socket Programming in Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Sending a DNS Query Over UDP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Transaction IDs and Request Correlation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Timeout Handling and Retransmission

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

UDP Truncation Detection (TC Bit)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cumulative Project State: End of Chapter 3

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 4: TCP Transport for Large Responses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Why DNS Needs TCP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

The Two-Byte Length Prefix Protocol

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Connecting to Upstream Servers Over TCP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Fallback from UDP to TCP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Connection Pooling Basics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Testing Both UDP and TCP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 5: Iterative Resolution Engine

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

The Recursive Resolution Algorithm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Loading Root Hints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Following NS Referrals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Glue Record Collection and Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

CNAME Chain Traversal

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

NXDOMAIN and SERVFAIL Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cumulative Project State: End of Chapter 5

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 6: The Cache – TTL Management and Negative Caching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cache Data Structure Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

TTL-Based Expiration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Negative Caching (NXDOMAIN, NODATA)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cache Size Limits and Eviction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Prefetching and Stale Responses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cache Statistics and Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Integrating the Cache with the Resolver

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 7: EDNS(0), DNSSEC Fundamentals, and Protocol Extensions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

EDNS(0) and the OPT Pseudo-Record

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Extended RCODEs and Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

DNSSEC Overview: DS, RRSIG, DNSKEY, NSEC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Basic DNSSEC Validation Chain

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Trust Anchors and Key Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

The DO Bit and Opt-Out

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 8: Dual-Stack IPv4/IPv6 Resolution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Dual-Stack Socket Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Resolving A and AAAA Records

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Happy Eyeballs and Address Selection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

IPv6 Root Servers and Anycast

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

NAT64/DNS64 Awareness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Testing Both Families

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 9: Concurrency, Event Loops, and Asynchronous I/O

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Why Single-Threaded Does Not Scale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Zig's Async/Await Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Building an Event Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Managing Thousands of Concurrent Queries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Asynchronous Socket Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Graceful Shutdown and Drainage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cumulative Project State: End of Chapter 9

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 10: Lock-Free Cache and Performance Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

The Cache Contention Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Sharded Hash Map Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Lock-Free Read Paths

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cache-Friendly Memory Layouts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Arena Allocators for DNS Packets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Performance Data Structure Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 11: Security Hardening – Cache Poisoning, Amplification, and Rate Limiting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cache Poisoning Attack Vectors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Source Port Randomization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Transaction ID Entropy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Response Rate Limiting (RRL)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Query Size Limits and Amplification Protection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Private Address Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 12: Logging, Metrics, Configuration, and Observability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Structured Logging Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Query-Level and Aggregate Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

YAML/JSON Configuration Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Health Check Endpoints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Debugging Tools and Packet Capture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 13: Testing, Fuzzing, and Quality Assurance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Unit Testing Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Integration Test Harness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Property-Based Testing for DNS Packets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Fuzzing DNS Parsers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Benchmarking and Performance Regression

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chaos Testing and Fault Injection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter 14: Profiling, Optimization, and Production Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Profiling with Zig Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Flame Graphs and Hot Path Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

SIMD and Compiler Optimizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Memory Allocation Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Systemd Service Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Docker Containerization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Production Deployment Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Cumulative Project State: Final Production Build

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

Conclusion: What We Built and Where to Go Next

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

What We Did Not Build

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

The Bigger Picture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/recursivednsinzig.