

TCP/IP A Visual Guide

Essential Foundations for New Systems Engineers

Reiner Braun

Sample Book — Chapters 1 to 2

Introduction

This book is designed for readers who are taking their first steps with TCP/IP, including systems engineers who want to build the right foundation from the start.

Rather than asking you to memorize more than you need, the book focuses on the essential concepts that will support your future study and day-to-day work. More advanced topics are included only when they are likely to become useful to many readers, and even then they are introduced only far enough to explain their purpose, context, and overall role.

Processes, relationships, and structures that can be difficult to picture from text alone are supported by diagrams. By moving between explanations, visuals, and examples, you can build a clearer and more practical understanding of the subject.

Product names, service names, logos, icons, and other identifying marks mentioned in this book are trademarks or registered trademarks of their respective owners. This book is not an official publication, certified training resource, or endorsed learning material of any company or organization.

How to Use This Book

Read the chapters in order when studying TCP/IP for the first time. Each chapter assumes the essential background established by the chapters before it, so this sequence builds the required knowledge step by step instead of presenting protocols as isolated facts.

Before reading the details of a chapter, review its opening big-picture section as the chapter's "**What You'll Be Able to Do**" orientation. It states the central conclusion and the practical capability that the chapter is intended to develop.

Use the diagrams to clarify **relationships, processes, boundaries, and structures** that are difficult to hold in mind from prose alone. Read the nearby explanation as well: each diagram supports the text but does not replace the definitions, limitations, or operational context.

Try the commands, code, and hands-on examples yourself whenever possible, using a lab, development system, or other environment in which you are authorized to perform the test. Before running a command, check its prerequisites and predict the expected result; then compare that prediction with the evidence you actually observe.

At the end of each chapter, use its concluding summary as a checkpoint. Treat the diagnostic questions and practical scenarios in the chapter as **review questions**, and confirm that you can explain the answer in your own words before moving on.

Advanced-topic overviews do not need to be memorized at this stage. Treat them as signposts: remember what problem each topic addresses and where it fits, then return to the details when a project or incident makes them relevant.

Table of Contents

Introduction

How to Use This Book

First Half: From the End-to-End Path to Packet-Size Constraints

Part I — Build the Mental Model

Chapter 1 — TCP/IP in One End-to-End Story

- The complete path is the most useful starting point
- A practical example: opening an internal project site
- The parcel-delivery analogy—and its limits
- Local, routed, and Internet communication
 - Communication on one local network
 - Communication through a router
 - Communication through the public Internet
- Where common devices appear
- Different stages use different identities
- A first troubleshooting walk-through
- Two important corrections
- Working model for the rest of the guide

Chapter 2 — Layers, Encapsulation, and the Meaning of an Address

- Layering separates responsibilities
- How the common models relate
- Encapsulation: adding the information each layer needs
- Names for data at each layer
- Four identifiers with different purposes
- What stays end to end and what changes at a router
- Where devices usually operate
- Using layers to troubleshoot without guessing
- Common mistakes to avoid
- The layered view to carry forward

First Half: From the End-to-End Path to Packet-Size Constraints

TCP/IP, named for Transmission Control Protocol and Internet Protocol, is the family of protocols that supports communication across interconnected networks. It is easiest to understand as one end-to-end path, not as a list of unrelated rules. IP supplies addressing and routed packet delivery, while TCP can provide a reliable stream between application endpoints. An application creates data, the operating system identifies the destination process, IP selects a path toward the destination host, the local network delivers the packet to the next hop, and the receiving system reverses those steps. When communication fails, the practical task is to identify the first stage at which expected progress stops.

This first half builds that foundation. It explains the path from an application on one host to an application on another, then examines Ethernet, both versions of IP, routing, transport ports, reliable byte streams, datagrams, and packet-size limits. Later services and network boundaries depend on these concepts, so they are intentionally not developed here.

Chapter 1 — TCP/IP in One End-to-End Story

The complete path is the most useful starting point

A **networked application succeeds only when every required stage of the communication path works**. A user may describe a problem as “the network is down,” but several independent mechanisms may be involved: local configuration, name resolution, route selection, local delivery, transport establishment, security negotiation, and the application itself.

A **network** is a set of connected systems that can exchange data. A **host** is a system that participates in that exchange, such as a laptop, server, virtual machine, phone, or network appliance. A **protocol** is an agreed set of rules for formatting and exchanging information. A **packet** is a formatted unit of data carried across a network; later chapters distinguish more precise terms used at different layers.

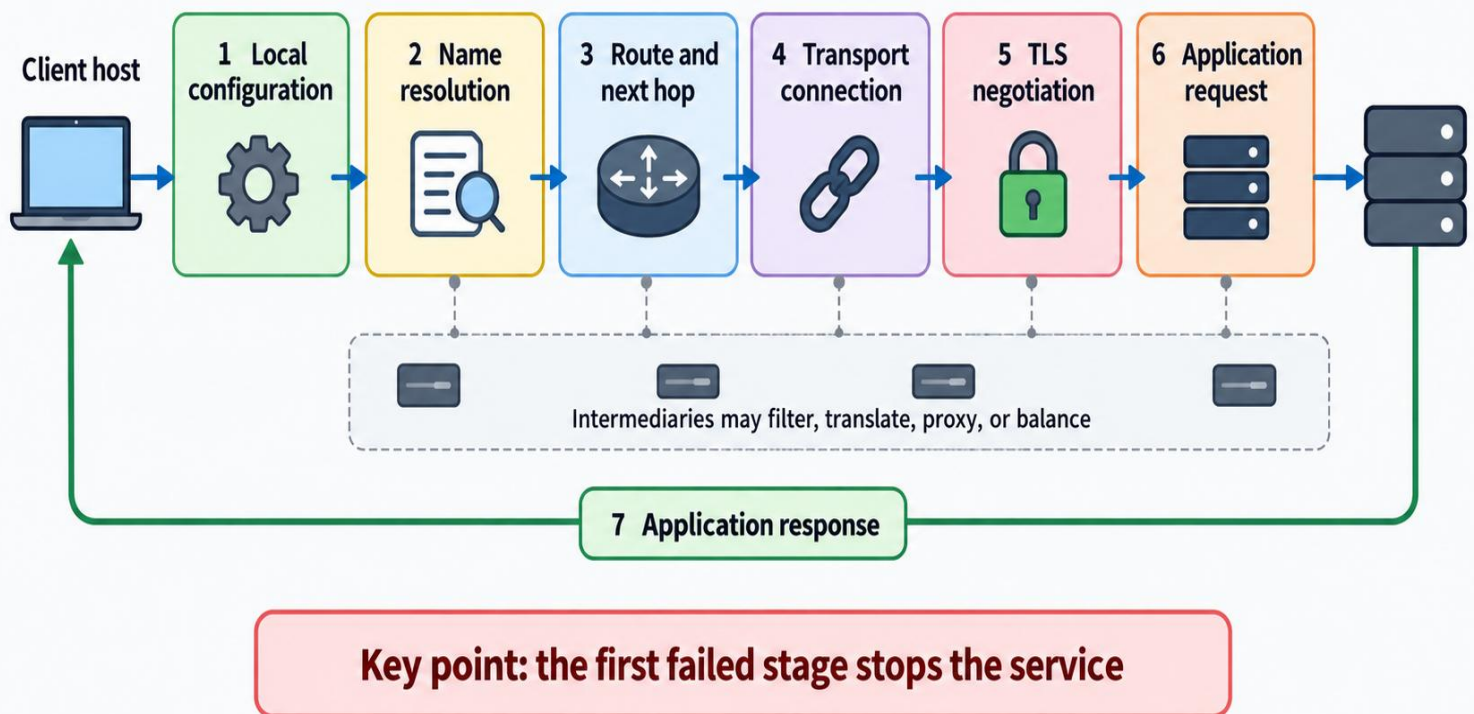
A **client** initiates communication to obtain a service, while a **server** waits for and handles client communication. A **request** asks a service to perform an operation or return information, and a **response** reports the result.

For systems engineers, the central working habit is:

Trace the communication in order, and ask where the expected next event failed to occur.

Chapter 1: TCP/IP in One End-to-End Story

1-1 Complete Service Path



001

Figure 1-1: Complete Service Path

This habit is more reliable than starting with random commands or changing several settings at once.

A practical example: opening an internal project site

Assume a workstation opens <https://project.example.internal/>. **Hypertext Transfer Protocol (HTTP)** defines the web request and response, while **Transport Layer Security (TLS)** protects that exchange; their combination is

commonly called HTTPS. The exact implementation differs among environments, but the communication usually follows this sequence:

1. **The workstation obtains or already has network settings.** It needs an address, a **network prefix** that identifies the local subnet boundary, a route toward remote networks, and usually one or more **name servers**, which answer requests that translate names into address information.
2. **The name is translated into an IP address.** An **IP address** identifies a network-layer source or destination, and **name resolution** is the process of converting a name such as `project.example.internal` into that form.
3. **The operating system selects a route.** A **route** is an instruction describing how to reach a destination prefix, either directly or through a **router**, which forwards IP packets between networks.
4. **The next hop is identified on the local link.** A **next hop** is the next locally reachable host or router to which the packet should be sent.
5. **A transport conversation is established.** A **transport protocol** carries data between application endpoints; for a conventional HTTPS connection over TCP, the client and server exchange control information before application data is sent.
6. **Security parameters are negotiated.** TLS establishes protected communication and normally authenticates the server. This stage is introduced here only to locate it in the path; its details belong later in the guide.
7. **The application request is sent.** The client sends an HTTP request, and the server returns an HTTP response.
8. **Intermediary devices may process the flow throughout the exchange.** A **firewall** applies traffic policy, a **proxy** communicates with another endpoint on a client's behalf, a **load balancer** distributes traffic among service instances, and an address-translation device can rewrite address or port information.
9. **The connection is reused or closed.** Modern applications often keep connections available for additional requests rather than creating a new connection each time.

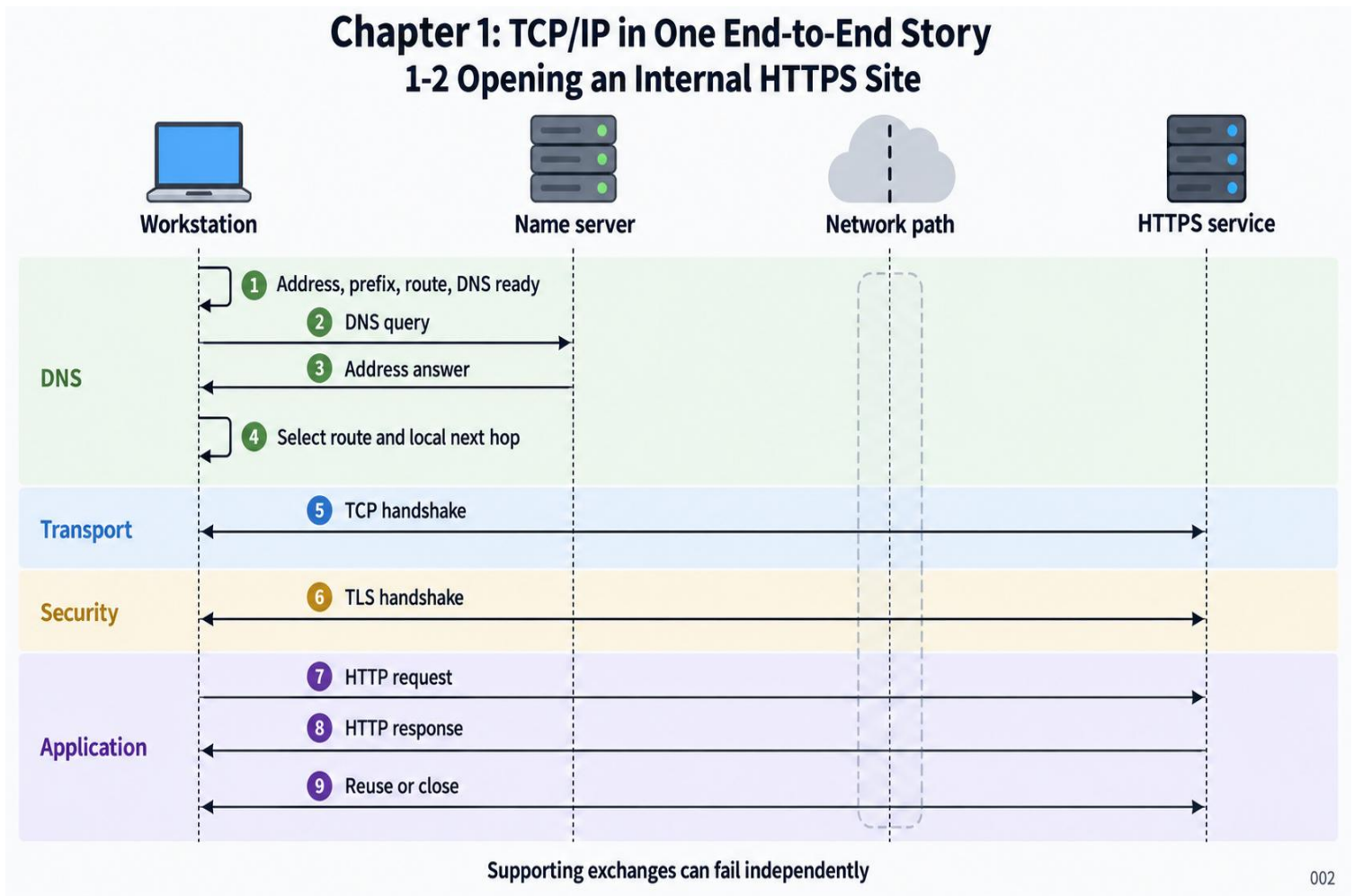


Figure 1-2: Opening an Internal HTTPS Site

The name-resolution request can itself require a routed network exchange with a name server; the sequence above follows the target service after that supporting exchange. These stages are related, but success at one stage does not

prove success at the next. A name may resolve correctly while the route to the service is missing. A TCP connection may succeed while the application returns an error. A small diagnostic packet may pass while larger packets fail.

The parcel-delivery analogy—and its limits

A parcel delivery provides a useful first model:

- The **recipient's name** resembles an application-level name that humans use.
- The **street address** resembles an IP address used to reach a host.
- A **local delivery office** resembles a router that sends the parcel toward the next region.
- A **department or room number** resembles a transport port that identifies the destination service.
- The **delivery rules and labels** resemble protocols.

The analogy helps separate responsibilities, but it is not a technical definition. Network communication can be duplicated, reordered, filtered, translated, retransmitted, or carried through several logical layers. The actual mechanisms in later chapters replace the analogy with precise behavior.

Local, routed, and Internet communication

The word “remote” can mean several different paths.

Communication on one local network

An **IP subnet** is a group of addresses that share a prefix and are treated as one network for local-delivery decisions. If the client and server are in the same subnet, the client can normally send the packet directly through the local network. A **switch** forwards local link-layer units called **frames**, but a router is not required for that specific exchange.

For example, a client at `192.0.2.10/24` and a server at `192.0.2.80/24` are in the same `/24` subnet; `/24` states that the first 24 address bits form the network prefix. The client treats the server as local and learns the server's local link address before sending the frame.

Communication through a router

If the destination is outside the client's local subnet, the client sends the packet to a **default gateway**, which is the router used when no more specific local path exists. The IP destination remains the remote server, but the first local frame is addressed to the router.

For example, a client at `192.0.2.10/24` cannot deliver directly to `198.51.100.25`. It sends the packet to a local router, perhaps `192.0.2.1`, which forwards it according to its own routing table.

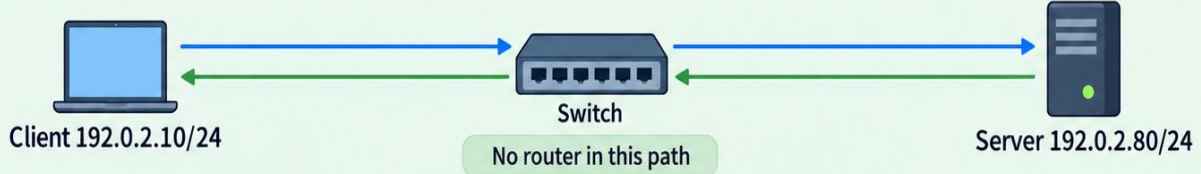
Communication through the public Internet

Internet communication is still routed communication. The packet may cross many independently operated networks, and the path in one direction may not be identical to the path in the other. Additional controls such as address translation, filtering, and load balancing are common, but the basic questions remain the same: which next hop is selected, whether the packet is forwarded, and whether a reply can return.

Chapter 1: TCP/IP in One End-to-End Story

1-4 Local, Routed, and Public-Internet Paths

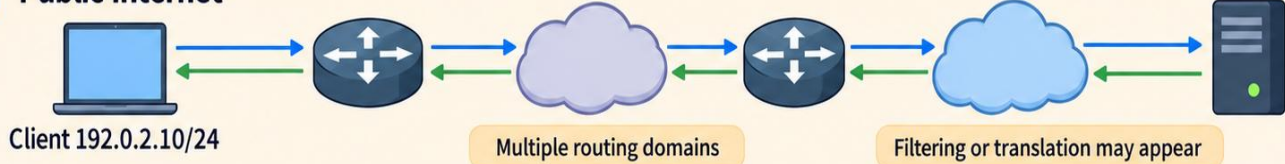
A Same subnet



B Routed network



C Public Internet



Core question: what is the next hop, and can the reply return?

003

Figure 1-3: Local, Routed, and Public-Internet Paths

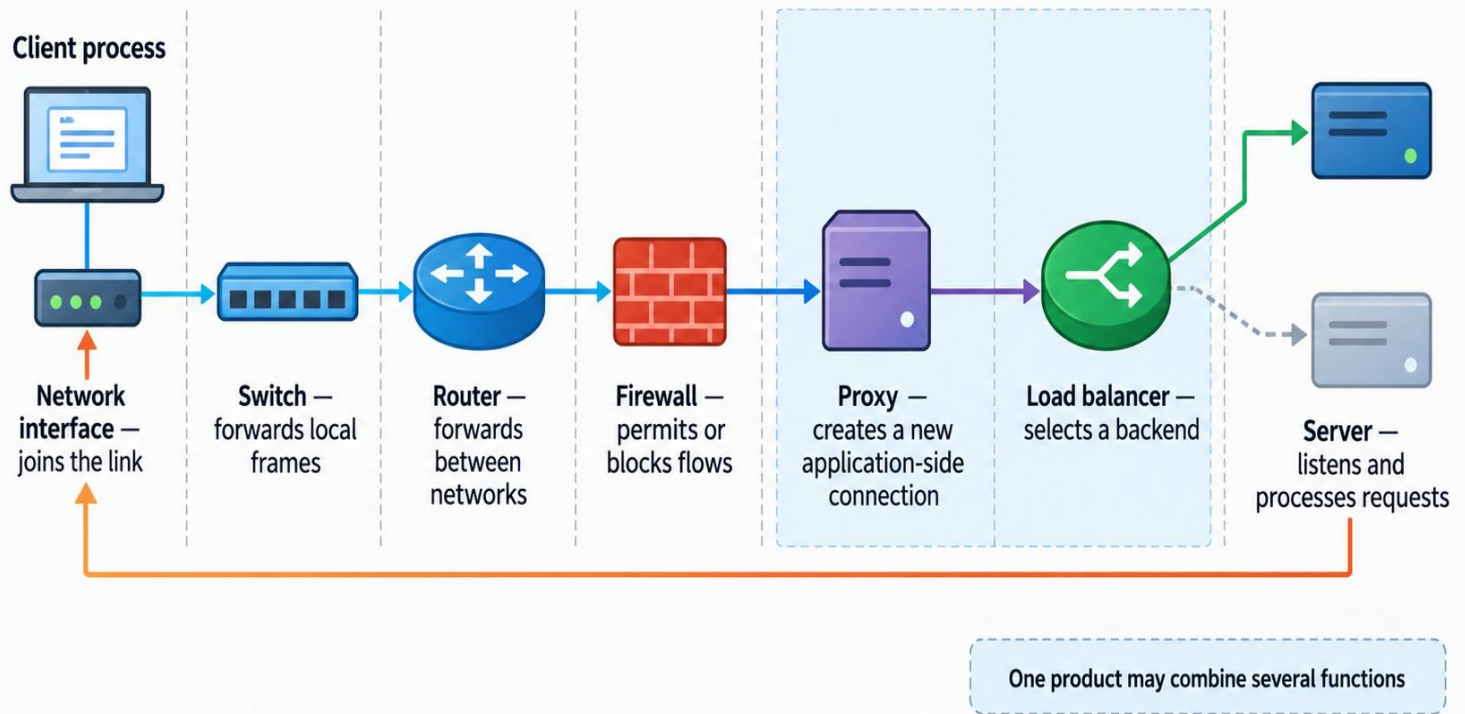
Where common devices appear

Several device names are often used as though they were interchangeable. Their primary functions differ.

Device or component	Main role in the path	Typical question it answers
Network interface	Connects a host to a network	Can this host send and receive on this link?
Switch	Forwards frames within a local link domain	Which local port leads toward this link-layer address?
Router	Forwards IP packets between networks	Which next hop leads toward this destination prefix?
Firewall	Applies traffic policy	Is this flow permitted in its current direction and state?
Proxy	Communicates on behalf of another endpoint	Should this application request be accepted and forwarded?
Load balancer	Distributes connections or requests	Which healthy backend should receive this traffic?
Server	Runs the destination service	Is the service listening and able to process the request?

Chapter 1: TCP/IP in One End-to-End Story

1-5 Where Common Devices Sit



004

Figure 1-4: Where Common Devices Sit

Products often combine several roles. A single appliance may route, filter, translate addresses, terminate TLS, and load-balance connections. Troubleshooting still improves when each function is considered separately.

Different stages use different identities

One communication flow can involve several forms of identification:

- A **name** identifies a service in a form useful to people and applications.
- An **IP address** identifies a network-layer destination or source interface.
- A **Media Access Control (MAC) address** identifies an interface for delivery on a local Ethernet domain.
- A **port number** identifies a transport-layer service endpoint on a host.

These values do not replace one another. A browser may start with a name, receive an IP address, send a local frame to a router's MAC address, and connect to a server port. All four can be present in one operation.

State is stored information about a current configuration or communication. Each stage maintains different state: a workstation may cache a name-resolution result, keep a local-neighbor entry, maintain a route table, and track a TCP connection. A firewall may keep independent flow state. A server process may track an application session. A stale or missing entry at one stage can fail while the other stages remain healthy.

A first troubleshooting walk-through

Suppose a developer reports that the internal project site does not open from a workstation. A disciplined first pass asks progressively narrower questions:

1. **Is the local interface operational and correctly addressed?**
2. **Does the name produce the expected destination address?**
3. **Which route does the workstation select for that address?**
4. **Can the workstation reach the required next hop?**
5. **Does the transport connection receive a reply?**
6. **Does the secure-session negotiation progress?**
7. **Does the application return a valid response?**

The answer may be “yes” through step five and “no” at step six. In that case, repeatedly testing basic IP reachability will not explain the failure. The useful evidence is at the security-negotiation stage.

A known-good comparison is especially valuable. If the same workstation can reach another service in the destination subnet, basic routing may be working. If another workstation on the same local network can reach the project site, the problem may be local to the first workstation. Comparisons do not prove a cause, but they reduce the scope efficiently.

Two important corrections

“The network is working” is not a precise technical conclusion. It may mean only that one test succeeded. A successful name lookup does not prove reachability, a successful ping does not prove that the service port is available, and a successful TCP connection does not prove that the application completed the request.

Reachability to a host is not the same as health of a service. A server can answer diagnostic traffic while its application is stopped, bound only to a local address, blocked on a different port, overloaded, or returning errors.

Working model for the rest of the guide

Keep one sentence in mind:

Application data moves through a layered stack, is delivered hop by hop across networks, and is finally directed to the correct process on the destination host.

The following chapters examine each part of that sentence.

Chapter 2 — Layers, Encapsulation, and the Meaning of an Address

Layering separates responsibilities

A **layered model** makes network communication understandable by assigning a main responsibility to each **layer**. An application does not need to know how a switch learns local addresses, and a switch does not need to interpret an HTTP request. Each layer uses the service of the layer below it and provides a service to the layer above it.

The five-layer teaching model is practical for systems work:

Layer	Main responsibility	Common examples
Application	Defines the meaning and format of application data	HTTP, remote-access protocols, name-resolution protocols
Transport	Delivers data between application endpoints	Transmission Control Protocol (TCP), User Datagram Protocol (UDP)
Internet	Addresses and routes packets between networks	Internet Protocol version 4 (IPv4), version 6 (IPv6), Internet Control Message Protocol (ICMP)
Link	Delivers frames across one local link	Ethernet, Wi-Fi link functions
Physical	Carries signals as electrical, optical, or radio energy	Copper, fiber, radio

The word **layer** describes a responsibility and an interface, not necessarily a separate device or software process. One operating-system kernel can implement several layers, and one network product can act at several layers. TCP supplies a reliable byte stream, while UDP supplies independent datagrams. IPv4 and IPv6 are the two IP versions covered here, and ICMP carries network-layer control and error information. **Ethernet** and Wi-Fi provide common local-link delivery mechanisms.

Chapter 2: Layers, Encapsulation, and the Meaning of an Address

2-1 TCP/IP Layers and Responsibilities

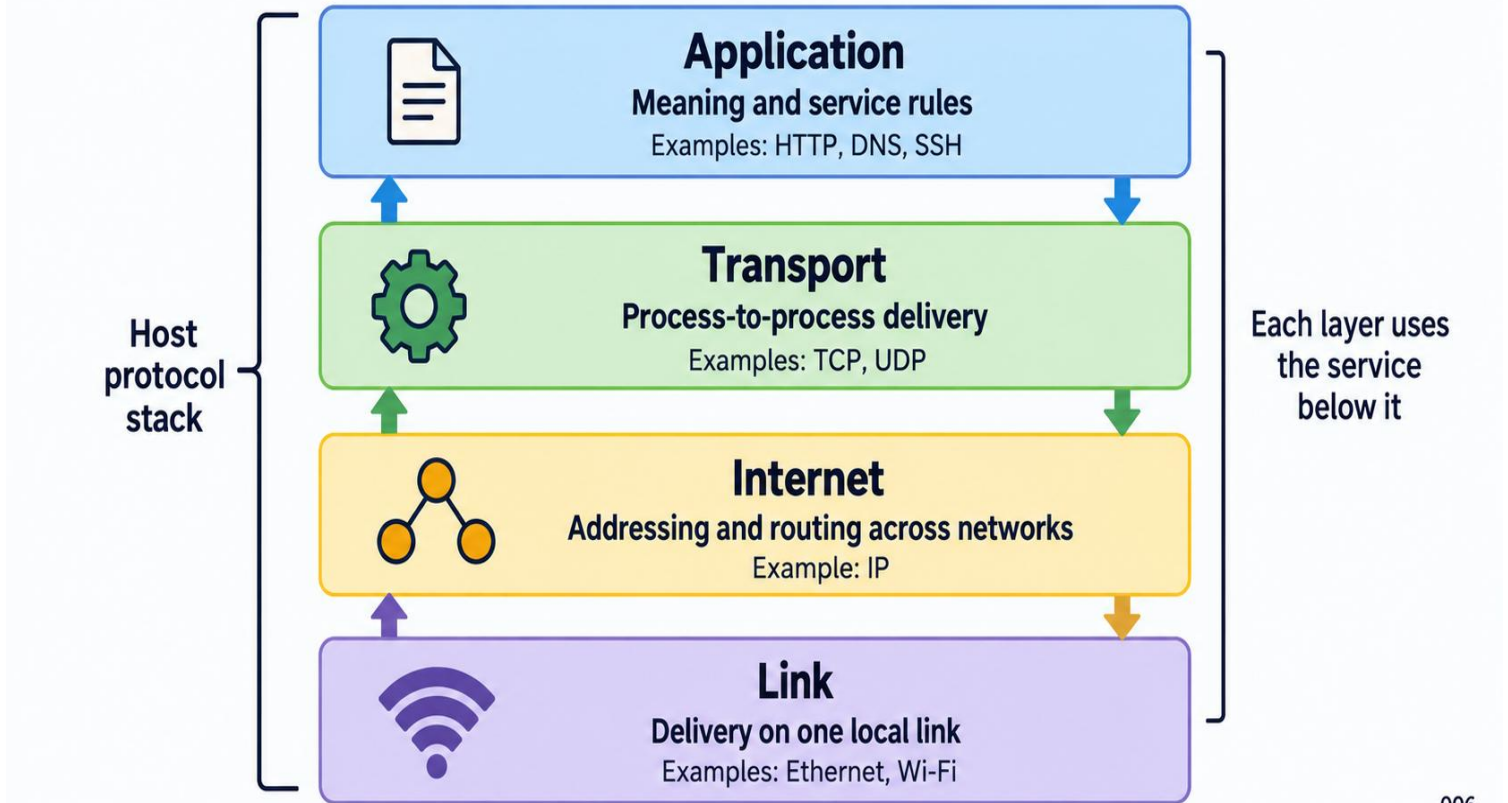


Figure 2-1: TCP/IP Layers and Responsibilities

How the common models relate

Engineers often encounter three models. They overlap, but their boundaries are not identical.

Five-layer teaching model	TCP/IP model	Open Systems Interconnection (OSI) reference model
Application	Application	Application, Presentation, Session
Transport	Transport	Transport
Internet	Internet	Network
Link	Link or Network Access	Data Link
Physical	Often included with Link	Physical

The **TCP/IP model** is a compact model associated with the Internet protocol suite. The **OSI model** is a seven-layer reference model used widely for vocabulary and reasoning. The five-layer model separates the physical and link responsibilities while keeping the upper layers simple.

These models are maps, not laws of nature. Real implementations can cross boundaries. A firewall may inspect both IP and application information. A modern application protocol may include functions that an OSI diagram would place in several layers. The models remain valuable because they let an engineer ask a focused question at each stage.

Encapsulation: adding the information each layer needs

Encapsulation is the process of placing data inside a new structure and adding a **header**, which is control information placed before the carried data, and sometimes a **trailer**, which is control information placed after it. The carried upper-layer content is the **payload**. **Decapsulation** is the reverse process at the receiver.

Consider an application sending the text `status` through TCP over IPv4 and Ethernet:

1. The application creates the bytes representing `status`.
2. TCP adds a TCP header containing transport information such as source and destination ports.
3. IPv4 adds an IP header containing source and destination IP addresses.
4. Ethernet adds a frame header containing local source and destination MAC addresses, plus a trailer used for error detection on the link.
5. The physical medium carries the resulting bits as signals.

At the receiving host, Ethernet validates and removes its framing information, IP processes and removes the IP header, TCP processes and removes the TCP header, and the application receives the original bytes.

Chapter 2: Layers, Encapsulation, and the Meaning of an Address

2-3 Encapsulation and Decapsulation

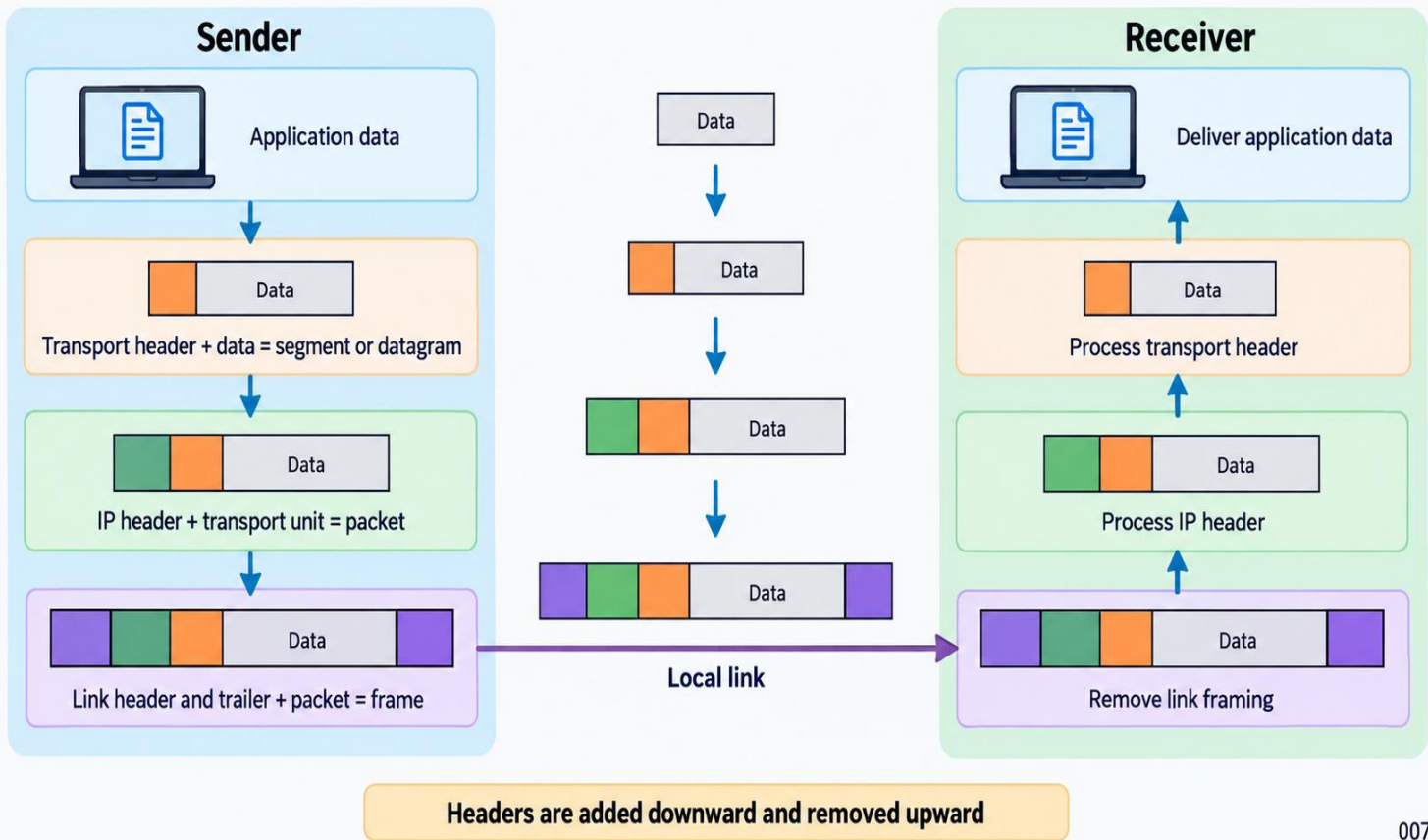


Figure 2-2: Encapsulation and Decapsulation

An envelope analogy is useful here. The application data is a letter. TCP places it in an envelope for the correct department. IP places that envelope inside another one addressed to the destination building. Ethernet places it in a local courier sleeve addressed to the next stop. At a router, the local courier sleeve is replaced, but the IP envelope continues toward the destination.

Names for data at each layer

A **protocol data unit** is the unit of data processed at a particular protocol layer. Common terms are:

- **Application data** for the bytes created or consumed by an application protocol.
- **TCP segment** for a TCP header and its associated data.
- **UDP datagram** for a UDP header and its associated message.
- **IP packet** for an IP header and its payload.
- **Ethernet frame** for an Ethernet header, payload, and trailer carried on one Ethernet link.

In everyday operations, people often use “packet” informally for several of these. Precise terms become important when discussing size limits, capture output, and which header a device can inspect.

Four identifiers with different purposes

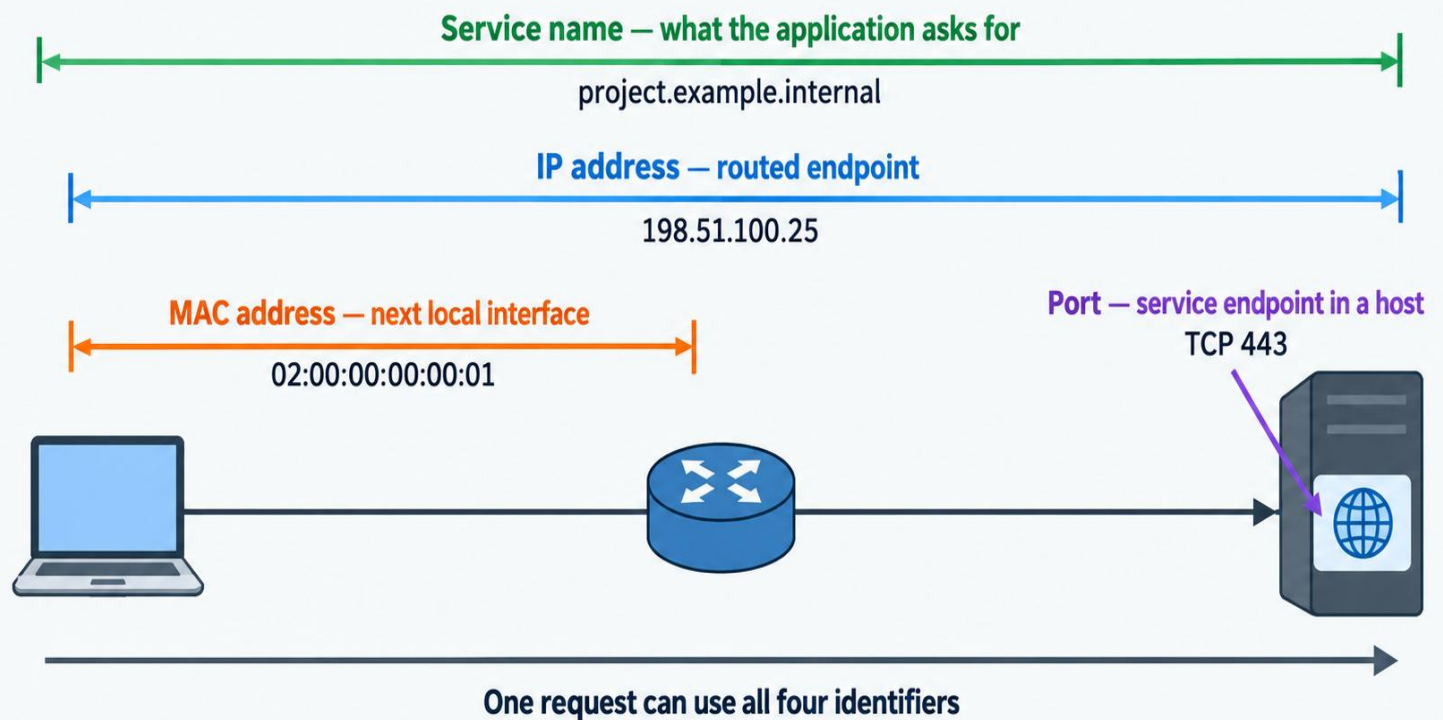
A common source of confusion is treating every identifier as another version of an address. They work at different scopes.

Identifier	What it identifies	Typical scope	Example
Name	A service or host in a human-usable namespace	Application and organizational scope	app.example.internal
IP address	A network-layer source or destination interface	Routed network scope	192.0.2.25
MAC address	An interface for delivery on one local Layer 2 domain	Local link scope	02:42:ac:11:00:02
Port number	A transport endpoint associated with an application	One host and one transport protocol	TCP port 443

A name can resolve to more than one IP address. An IP address can host many services on different ports. A host can have several interfaces and therefore several IP and MAC addresses. TCP port 53 and UDP port 53 are separate transport endpoints even though the numeric port is the same.

Chapter 2: Layers, Encapsulation, and the Meaning of an Address

2-5 Names, IP Addresses, MAC Addresses, and Ports



008

Figure 2-3: Names, IP Addresses, MAC Addresses, and Ports

What stays end to end and what changes at a router

Assume a client sends an IPv4 packet to a server through two routers. In a simple path without address translation:

- The **source and destination IP addresses normally remain the client and server addresses** from end to end.
- The **source and destination transport ports normally remain the application endpoints** from end to end.
- The **source and destination MAC addresses change on every Ethernet link** because each frame is used only for one local delivery step.
- The IPv4 **Time to Live**, or TTL, decreases at each router, and the IPv4 header checksum is updated accordingly.
- Frame-specific error-checking information is recalculated for each new frame.

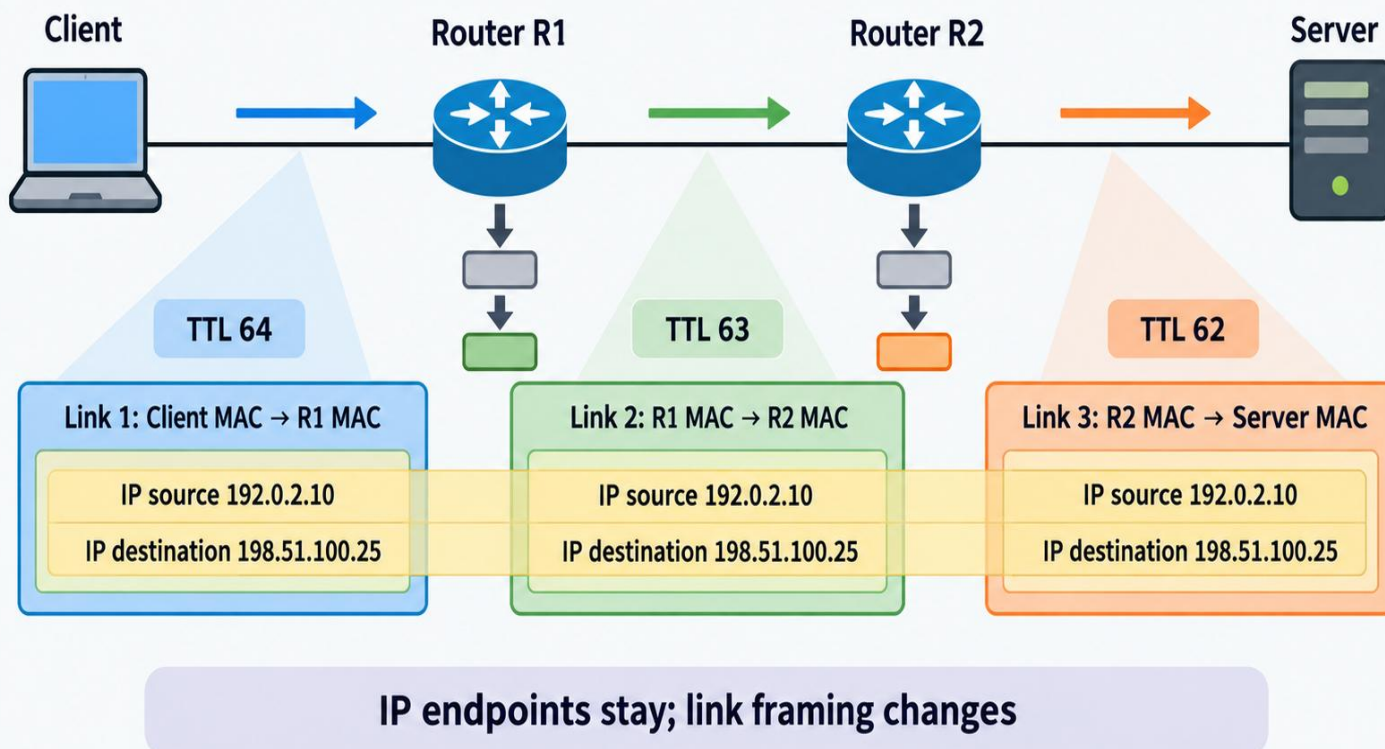
A simplified path looks like this:

Client ---- Router A ---- Router B ---- Server

On the first link, the frame destination is Router A's MAC address. On the second link, Router A creates a new frame addressed toward Router B. On the final link, Router B creates a frame addressed to the server. The destination IP address remains the server's address throughout this example.

Chapter 2: Layers, Encapsulation, and the Meaning of an Address

2-6 End-to-End and Hop-by-Hop Fields



009

Figure 2-4: End-to-End and Hop-by-Hop Fields

Intermediaries can deliberately change end-to-end values. Address translation can rewrite IP addresses or ports, and a proxy can terminate one connection and create another. Those mechanisms are deferred because the basic unchanged case must be clear first.

Where devices usually operate

It is useful to associate devices with a primary layer, while remembering that modern products often inspect more than one.

Component	Primary operating level	What it commonly examines
Ethernet switch	Link	MAC addresses and frame information
Router	Internet	Destination IP prefix and routing state
Stateful firewall	Internet and transport	IP addresses, protocol, ports, direction, connection state
Layer 4 load balancer	Transport	IP addresses, transport protocol, ports, flow state
Layer 7 proxy	Application	Application requests plus lower-layer connection information
Application server	Application	Protocol messages and application state

A router may also filter traffic. A switch may include routing functions. A firewall may inspect application data. The table describes the core decision each role is expected to make.

Using layers to troubleshoot without guessing

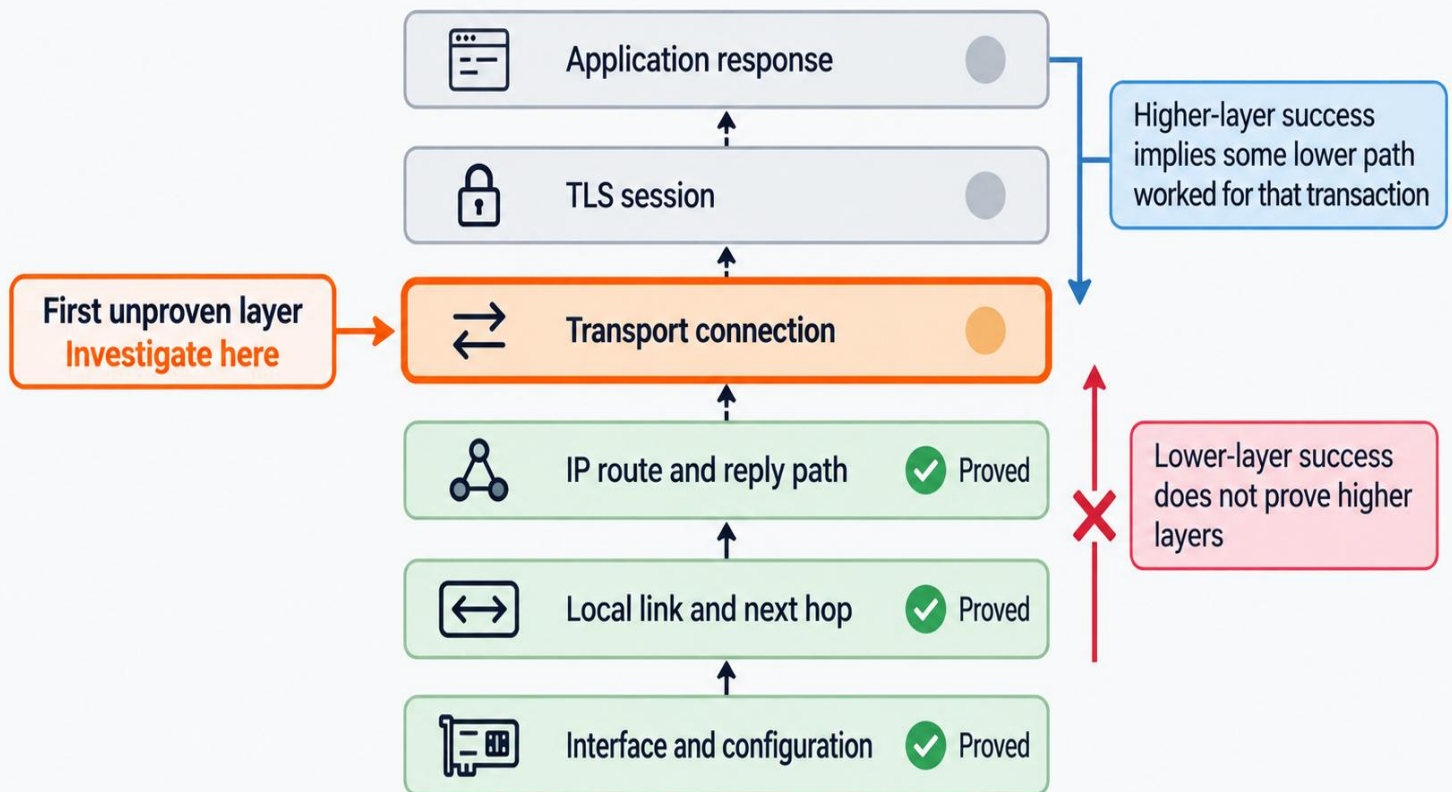
Layered troubleshooting begins with the lowest relevant evidence and moves upward:

1. **Physical:** Is the interface connected, and is the link reported as up?
2. **Link:** Can the host discover and reach the local next hop?
3. **Internet:** Does the host have the correct address and route, and can packets travel in both directions?
4. **Transport:** Is the destination port reachable, and does the handshake or datagram exchange behave as expected?
5. **Application:** Does the protocol request receive a valid application response?

This is not a requirement to test every layer for every incident. Evidence can justify starting higher. For example, an HTTP response code proves that lower layers carried enough traffic for the server application to respond. The model helps avoid testing conditions that are already demonstrated.

Chapter 2: Layers, Encapsulation, and the Meaning of an Address

2-8 Layered Troubleshooting as Evidence



010

Figure 2-5: Layered Troubleshooting as Evidence

Common mistakes to avoid

The OSI model is not a perfect description of every packet or product. Saying that a problem is “Layer 3” can be useful shorthand, but the evidence should still identify the actual failed behavior, such as a missing route or an incorrect source address.

MAC addresses, IP addresses, and ports are not interchangeable. A switch forwarding a frame by MAC address is not selecting an application service. A router forwarding by IP prefix is not deciding which process receives the data. The destination host uses the transport protocol and port for that final process-level delivery.

The layered view to carry forward

Encapsulation lets each layer add only the information needed for its responsibility. Local frames move hop by hop, IP packets move across routed networks, transport protocols identify application endpoints, and application protocols give the exchanged bytes meaning. This separation is the structure used throughout TCP/IP troubleshooting.
