How the Internet Works

Volume 2: Core Protocols and Standards

Generated: September 11, 2025

A Complete Technical Guide

From Physical Infrastructure to Future Technologies

Table of Contents

Core Protocols and Standards (TCP/IP Stack)

The internet's protocols are organized in layers, each handling specific aspects of communication. This layered approach, formalized in the OSI model and practically implemented in the TCP/IP stack, allows complex networking to be broken into manageable pieces. Changes at one layer don't require changes at others - a key reason the internet has been able to evolve continuously.

Understanding Protocol Layers

Network protocols work like a series of nested envelopes, with each layer adding its own envelope (header) around the data from the layer above. When data arrives at its destination, each layer removes its corresponding envelope and processes the information. This encapsulation allows each layer to operate independently while working together to deliver data.

The OSI (Open Systems Interconnection) model defines seven layers, though the internet primarily uses the simpler four-layer TCP/IP model. Understanding these layers is crucial for troubleshooting network issues and designing network applications. Each layer has specific responsibilities and communicates with its peer layer on the remote system.

The TCP/IP Stack Layers:

The Application Layer is where user applications operate (HTTP, FTP, SSH). It creates the actual data to be transmitted with no knowledge of network details. Examples include web browsers and email clients.

The Transport Layer handles end-to-end communication (TCP, UDP). It ensures reliable delivery (TCP) or fast delivery (UDP), manages flow control and congestion, and multiplexes multiple applications via port numbers.

The Network Layer manages routing between networks (IP). It handles addressing and routing with no guarantee of delivery, working across different physical networks.

The Link Layer manages local network communication (Ethernet, Wi-Fi). It handles communication on a single network segment using physical addressing (MAC addresses) and error detection for transmitted frames.

Network Layer (IP) - The Internet's Foundation

IP (Internet Protocol) is the fundamental protocol that creates the internet by allowing different networks to interconnect. It provides addressing and basic routing, creating a universal overlay that works regardless of the underlying physical network technology. IP is deliberately simple - it makes no guarantees about delivery, leaving reliability to higher layers.

IPv4 - The Original Internet Protocol

IPv4 was designed in the 1970s when the internet was a research project with a few hundred computers. Its 32-bit addresses seemed inexhaustible at the time, but the internet's explosive growth exhausted the address space. Despite this limitation, IPv4 remains dominant through various workarounds like NAT.

Address structure: IPv4 uses 32 bits represented as four octets in dotted decimal notation (e.g., 192.168.1.1). Each octet ranges from 0-255 (8 bits) for a total of 4,294,967,296 (2^32) addresses.

Special ranges include:

- 10.0.0.0/8: Private addresses (16 million addresses)
- 172.16.0.0/12: Private addresses (1 million addresses)
- 192.168.0.0/16: Private addresses (65,536 addresses)
- 127.0.0.0/8: Loopback (localhost)
- 224.0.0.0/4: Multicast
- 169.254.0.0/16: Link-local (self-assigned)

Packet structure: Headers contain routing information including:

- Version (4 bits): IPv4 = 4
- Header length (4 bits): Usually 20 bytes
- Type of Service (8 bits): Quality of service marking
- Total length (16 bits): Packet size including header
- Identification (16 bits): For reassembling fragments

- Flags (3 bits): Control fragmentation
- Fragment offset (13 bits): Position in original packet
- TTL - Time to Live (8 bits): Hop countdown preventing loops
- Protocol (8 bits): What's inside (TCP=6, UDP=17, ICMP=1)
- Header checksum (16 bits): Error detection
- Source IP (32 bits): Sender's address
- Destination IP (32 bits): Recipient's address
- Options (variable): Rarely used

Subnetting and CIDR organize address space:

- Subnet mask defines network vs host portions
- CIDR notation: 192.168.1.0/24 means first 24 bits are network
- VLSM (Variable Length Subnet Masking): Different sized subnets
- Route aggregation: Combining multiple networks into single route

IPv6 - The New Internet Protocol

IPv6 was designed to solve IPv4's address exhaustion while improving various aspects of the protocol. With 128-bit addresses, IPv6 provides enough addresses to assign thousands to every square meter of Earth's surface. Beyond addressing, IPv6 simplifies routing, improves security, and enables better autoconfiguration.

Address structure: IPv6 uses 128 bits in eight 16-bit groups with hexadecimal notation: 2001:0db8:85a3:0000:0000:8a2e:0370:7334

Abbreviation rules:

- Leading zeros omitted: 2001:db8 not 2001:0db8
- Consecutive zeros compressed: 2001:db8::8a2e:370:7334
- Only one :: compression per address

Total addresses: 340 undecillion (3.4×10^38)

Address types:

- Unicast: One-to-one communication
- Multicast: One-to-many communication
- Anycast: One-to-nearest (same address in multiple locations)

- No broadcast (replaced by multicast)

Address allocation:

- /48 typical for organizations (65,536 subnets)
- /56 for home users (256 subnets)
- /64 for single subnet (18 quintillion addresses)
- /128 for single host

Simplified header: More efficient than IPv4

- Fixed 40-byte header (vs variable IPv4)
- No fragmentation by routers (only by source)
- No header checksum (redundant with link layer)
- Flow label for QoS
- Next header for extension headers

Extension headers: Optional features

- Hop-by-hop options
- Routing header
- Fragment header
- Authentication header (IPSec)
- Encapsulating Security Payload (IPSec)

Autoconfiguration: Devices can self-assign addresses

- SLAAC (Stateless Address Autoconfiguration)
- Uses MAC address to create unique interface ID
- Router advertisements provide network prefix
- No DHCP required for basic connectivity

NAT - Network Address Translation

NAT became essential as IPv4 addresses became scarce. It allows multiple devices to share a single public IP address by modifying packet headers as they pass through a router. While NAT has extended IPv4's life by decades, it complicates protocols and breaks end-to-end connectivity.

Types of NAT:

- Static NAT: One-to-one mapping
- Dynamic NAT: Pool of public addresses
- PAT/NAPT (Port Address Translation): Most common, maps internal IP:port to external IP:port, allows thousands of devices behind one IP, connection tracking table maintains mappings

NAT traversal challenges:

- Incoming connections blocked by default
- Protocols embedding IP addresses break
- Peer-to-peer applications need workarounds
- Solutions: UPnP, STUN, TURN, ICE

Transport Layer - Reliable Delivery

The transport layer provides services that IP doesn't - reliability, flow control, and application multiplexing. The two main transport protocols, TCP and UDP, make different trade-offs between reliability and efficiency. Applications choose based on their specific needs.

TCP - Transmission Control Protocol

TCP provides reliable, ordered, error-checked delivery of data between applications. It's the workhorse of the internet, carrying the vast majority of traffic including web browsing, email, and file transfers. TCP handles packet loss, duplication, and reordering transparently, presenting a reliable byte stream to applications.

Connection establishment - Three-way handshake:

1. SYN: Client sends synchronize packet with initial sequence number (ISN) randomly chosen, maximum segment size (MSS) announced, and window size for flow control

2. SYN-ACK: Server acknowledges and synchronizes, acknowledging client's ISN + 1, providing server's own ISN, and server's receive window

3. ACK: Client acknowledges server, connection established, data can now flow both ways

Sequence and acknowledgment numbers:

- Every byte numbered sequentially
- ACK indicates next expected byte
- Sliding window for multiple unacknowledged segments
- Selective acknowledgments (SACK) for non-contiguous data

Flow control: Preventing receiver overflow

- Receive window: Buffer space available
- Window scaling: Multiply window by 2^n for large buffers
- Zero window: Receiver temporarily stops sender
- Window probes: Check if window reopened

Congestion control: Preventing network overflow

Slow start: Exponential growth from small window

- Initial window: 2-10 segments
- Double each RTT until threshold

Congestion avoidance: Linear growth after threshold

- Increase by one segment per RTT
- AIMD (Additive Increase Multiplicative Decrease)

Fast retransmit: Three duplicate ACKs trigger resend

Fast recovery: Don't return to slow start

Modern algorithms:

- Cubic: Default in Linux, better for high-bandwidth
- BBR: Google's algorithm, measures bandwidth and RTT
- Compound TCP: Windows, combines loss and delay signals

Connection termination:

- Four-way handshake (can be three-way)
- FIN flag indicates no more data

- Each direction closed independently
- TIME_WAIT state prevents old packets interfering

UDP - User Datagram Protocol

UDP provides minimal transport layer functionality - just application multiplexing via ports and optional checksums. It adds only 8 bytes of header compared to TCP's 20+. Applications that need speed more than reliability, or that implement their own reliability mechanisms, choose UDP.

Header structure: Simple 8-byte header

- Source port (16 bits)
- Destination port (16 bits)
- Length (16 bits)
- Checksum (16 bits, optional in IPv4)

Use cases:

- DNS queries: Speed critical, retry at application layer
- VoIP/Video: Old data worthless, forward error correction
- Gaming: Latest state matters, not old positions
- QUIC: Implements TCP-like features over UDP
- Multicast: TCP doesn't support one-to-many

Advantages over TCP:

- No connection establishment delay
- No head-of-line blocking
- Constant packet rate possible
- Multicast/broadcast capable
- Lower overhead

QUIC - Quick UDP Internet Connections

QUIC is a modern transport protocol built on UDP that combines the reliability of TCP with improvements in latency and multiplexing. Developed by Google and standardized by the IETF, QUIC is the foundation of HTTP/3 and represents the first major transport protocol innovation in decades.

Key innovations:

- 0-RTT connection establishment for known servers
- Multiplexing without head-of-line blocking
- Connection migration (changing IP addresses)
- Built-in encryption (TLS 1.3)
- Better loss recovery than TCP

Performance benefits:

- Reduced latency for connection setup
- Better performance on lossy networks
- Seamless network transitions (Wi-Fi to cellular)
- Improved congestion control

Application Layer Protocols

Application layer protocols define how programs communicate over the network. These protocols handle everything from web browsing to email to file transfer. They rely on the transport layer for delivery while focusing on application-specific functionality.

HTTP/HTTPS - The Web's Protocol

HTTP (HyperText Transfer Protocol) is the foundation of the World Wide Web. It's a request-response protocol where clients (usually browsers) request resources from servers. HTTPS adds TLS encryption for security. The evolution from HTTP/1.1 to HTTP/2 to HTTP/3 shows how protocols adapt to changing requirements.

HTTP/1.1 - The longtime standard:

- Text-based protocol, human-readable
- Persistent connections (multiple requests per connection)
- Pipelining (send multiple requests without waiting)

Problems:

- Head-of-line blocking
- One request at a time per connection
- Browsers open multiple connections (typically 6)

HTTP/2 - Binary and multiplexed:

- Binary protocol for efficiency
- Multiplexing: Multiple streams over single connection
- Header compression (HPACK)
- Server push: Send resources before requested
- Stream prioritization
- Still uses TCP (head-of-line blocking at TCP layer)

HTTP/3 - Built on QUIC:

- Uses QUIC instead of TCP
- No head-of-line blocking
- Faster connection establishment
- Better performance on mobile networks
- Gradually being adopted (30%+ of web traffic)

Request methods:

- GET: Retrieve resource
- POST: Submit data
- PUT: Upload resource
- DELETE: Remove resource
- HEAD: Get headers only
- OPTIONS: Get allowed methods
- PATCH: Partial update

Status codes:

- 1xx: Informational
- 2xx: Success (200 OK, 201 Created)
- 3xx: Redirection (301 Moved Permanently, 302 Found)
- 4xx: Client error (404 Not Found, 403 Forbidden)
- 5xx: Server error (500 Internal Server Error, 503 Service Unavailable)

Important headers:

- Host: Required for virtual hosting
- User-Agent: Browser identification
- Accept: Content types client accepts
- Content-Type: Type of content being sent
- Cookie/Set-Cookie: Session management
- Cache-Control: Caching directives
- Authorization: Authentication credentials

DNS - Domain Name System Protocol

While DNS is a naming system, it's also an application protocol that defines how DNS queries and responses work. DNS primarily uses UDP for queries but can fall back to TCP for large responses or zone transfers.

Query types:

- Recursive: Resolver does all work
- Iterative: Client follows referrals
- Forward: Query passed to another resolver
- Reverse: IP address to domain name

Resource record types:

- A: IPv4 address
- AAAA: IPv6 address
- CNAME: Canonical name (alias)
- MX: Mail exchanger
- TXT: Text data (SPF, DKIM, etc.)

- NS: Name server
- SOA: Start of authority
- PTR: Pointer (reverse DNS)
- SRV: Service location

DNS over HTTPS (DoH) / DNS over TLS (DoT):

- Encrypts DNS queries
- Prevents eavesdropping and manipulation
- DoH uses port 443 (looks like HTTPS)
- DoT uses port 853 (identifiable as DNS)