Core Networking Protocols and Ports Guide

A comprehensive reference for essential networking concepts, protocols, terminology, and port numbers

Fundamental Networking Terminology

What is a Protocol?

A **protocol** is a set of rules and standards that define how data is transmitted and received over a network. Think of it as a common language that devices use to communicate with each other.

What is a Port?

A **port** is a logical endpoint for network communications. Ports are numbered from 0 to 65535 and help identify which application or service should handle incoming data. Ports 0-1023 are called "well-known ports" and are reserved for standard services.

Client-Server Model

A **client** is a device or application that requests services or resources. A **server** is a device or application that provides services or resources. Most network protocols operate on this model.

Packet

A **packet** is a small unit of data transmitted over a network. Large messages are broken into packets, sent independently, and reassembled at the destination.

TCP and UDP - The Foundation

TCP (Transmission Control Protocol)

- **Connection-oriented** → Establishes a connection before data transfer (like a phone call)
- **Reliable** → Ensures data arrives correctly, in order, and without loss
- **Error Checking & Retransmission** → If packets are lost/dropped, TCP resends them
- **Flow Control & Congestion Control** → Prevents overwhelming the receiver or network
- **Slower than UDP** (because of overhead), but reliable

Real Life Analogy: Sending a registered letter with tracking → You know exactly when it's delivered.

UDP (User Datagram Protocol)

- **Connectionless** → No handshake, just sends data (like a postcard)
- **Unreliable** → No guarantee of delivery, order, or duplication checks
- **No Flow Control** → Just sends as fast as possible
- **Lightweight & Fast** → Minimal overhead, best for speed over reliability

Real Life Analogy: Sending a regular postcard → It might arrive, might get lost, but it's fast and simple.

Key Differences (TCP vs UDP)

Feature	TCP	UDP
Type	Connection-oriented	Connectionless
Reliability	Reliable (acknowledgements, retransmissions)	Unreliable (no delivery guarantee)
Speed	Slower (more overhead)	Faster (less overhead)
Ordering	Packets arrive in sequence	Packets may arrive out of order
Error Checking	Yes (and fixes errors)	Basic checksum only
Use Case	When accuracy matters (web, file transfer, emails)	When speed matters (streaming, calls, games)

Examples of Protocols using TCP:

- **HTTP/HTTPS** (web browsing) → `Port 80/443`
- **FTP** (file transfer) → `Port 21`
- **SMTP** (email sending) → `Port 25`
- **Telnet/SSH** (remote login) → `Port 23/22`

Examples of Protocols using UDP:

- **DNS** → `Port 53` (fast lookups)
- **DHCP** → `Ports 67/68` (quick IP assignment)
- **VoIP** (Skype, WhatsApp calls) → Needs speed > reliability
- **Online Gaming** → Dropping 1 packet doesn't matter, but speed does

🌐 DNS (Domain Name System)

DNS Overview

Purpose: Translates domain names → IP addresses.

Port: `UDP 53` (queries), `TCP 53` (zone transfers)

How it works: You type `google.com`, DNS resolves it to an IP (e.g., 142.250.72.206).

Example: Without DNS, you'd have to remember IPs for every website.

Key DNS Terminology

- **Domain Name:** Human-readable address (e.g., www.example.com)
- **IP Address:** Numerical address that computers use (e.g., 192.168.1.1)
- **DNS Server:** Server that maintains a database of domain names and their corresponding IP addresses
- **DNS Query:** Request sent to a DNS server to resolve a domain name
- **DNS Cache:** Temporary storage of DNS query results to speed up future lookups
- **Authoritative DNS Server:** The server that has the final answer for a domain

🖥️ DHCP (Dynamic Host Configuration Protocol)

DHCP Overview

Purpose: Automatically assigns IP addresses and network configurations.

Port: `UDP 67` (server), `UDP 68` (client)

How it works: Client sends a broadcast "I need an IP," DHCP server replies with IP, subnet mask, gateway, DNS.

Example: Your phone automatically gets an IP when joining Wi-Fi.

DHCP Process (DORA)

1. **Discover** - Client broadcasts to find DHCP servers
2. **Offer** - DHCP server offers an IP address
3. **Request** - Client requests the offered IP address
4. **Acknowledge** - Server confirms and assigns the IP

Important DHCP Terms

- **Lease Time:** Duration for which an IP address is assigned to a device
- **Scope:** Range of IP addresses that a DHCP server can assign
- **Reservation:** Permanent IP address assignment for specific devices (based on MAC address)
- **Default Gateway:** Router's IP address that connects the local network to other networks

HTTP and HTTPS

HTTP (HyperText Transfer Protocol)

Purpose: Web browsing (not secure).

Port: `TCP 80`

How it works: Client (browser) requests a webpage → server responds with HTML.

Example: Visiting a website using `http://`.

HTTPS (HyperText Transfer Protocol Secure)

Purpose: Secure web browsing (encrypted with SSL/TLS).

Port: `TCP 443`

How it works: Same as HTTP but with encryption → prevents eavesdropping.

Example: Online banking, shopping, secure logins (`https://`).

HTTP Methods

- **GET** - Retrieve data from server
- **POST** - Submit data to server
- **PUT** - Update existing resource
- **DELETE** - Remove resource
- **HEAD** - Retrieve headers only

HTTP Status Codes

Code Range	Meaning	Examples
2xx	Success	200 OK, 201 Created
3xx	Redirection	301 Moved Permanently, 302 Found
4xx	Client errors	404 Not Found, 403 Forbidden
5xx	Server errors	500 Internal Server Error, 503 Service Unavailable

SSH (Secure Shell)

SSH Overview

Purpose: Secure remote login and command execution.

Port: `TCP 22`

How it works: Encrypts all traffic (unlike Telnet). Used mainly in Linux/Unix.

Example: A system admin logs into a Linux server from home securely.

Common SSH Commands

```
# Connect to remote server $ ssh user@hostname # Connect using custom port $ ssh -p 2222 user@hostname # Securely copy files $ scp file.txt user@hostname:/path/ # Generate SSH key pair $ ssh-keygen # Copy public key to server $ ssh-copy-id user@hostname
```

SSH Key Authentication

- **Public Key:** Can be shared freely; placed on servers you want to access
- **Private Key:** Must be kept secret; stored on your local machine
- **Key Pair:** Works together - server verifies you have the private key matching the public key

FTP (File Transfer Protocol)

FTP Overview

Purpose: Transfer files between client and server.

Port: `TCP 21` (control), `TCP 20` (data)

How it works: Client connects to FTP server, authenticates, then uploads/downloads files.

Example: A web developer uploads website files to a hosting server.

FTP Modes

- **Active Mode:** Server initiates data connection back to client
- **Passive Mode:** Client initiates both control and data connections (better for firewalls)

Secure FTP Alternatives

- **FTPS (FTP Secure):** FTP with SSL/TLS encryption → `Port 990`
- **SFTP (SSH File Transfer Protocol):** File transfer over SSH → `Port 22`

⚠ **Security Note:** Regular FTP transmits data in plain text, including passwords. Always prefer FTPS or SFTP for secure file transfers.

Email Protocols

SMTP (Simple Mail Transfer Protocol)

Purpose: Sending emails from client to server, or between mail servers.

Port: `TCP 25` (standard), `TCP 587` (submission), `TCP 465` (SMTPS)

How it works: Your email client sends message to SMTP server → server forwards to recipient's mail server.

Example: Sending an email from Gmail to Outlook.

POP3 (Post Office Protocol v3)

Purpose: Downloading emails from server to client.

Port: `TCP 110` (standard), `TCP 995` (POP3S with SSL/TLS)

How it works: Downloads emails to your device and usually deletes them from server.

Example: Checking email on your phone, emails are downloaded and removed from server.

IMAP (Internet Message Access Protocol)

Purpose: Accessing and managing emails on the server.

Port: `TCP 143` (standard), `TCP 993` (IMAPS with SSL/TLS)

How it works: Emails stay on server, synced across all devices.

Example: Reading emails on phone and laptop - both show the same inbox state.

POP3 vs IMAP Comparison

Feature	POP3	IMAP
Email Storage	Downloaded to device, removed from server	Stored on server, accessed remotely
Multi-Device Access	Limited (emails on one device)	Excellent (synced across devices)

Feature	POP3	IMAP
Server Storage	Minimal (emails deleted)	Uses server storage
Offline Access	Full access to downloaded emails	Limited without internet
Best For	Single device, limited server space	Multiple devices, cloud access

Remote Access Protocols

Telnet (Teletype Network)

Purpose: Remote command-line access to devices.

Port: `TCP 23`

How it works: Establishes text-based terminal connection to remote system.

Example: Configuring network routers and switches (legacy use).

⚠ **Critical Security Warning:** Telnet transmits everything (including passwords) in plain text. It has been largely replaced by SSH. Never use Telnet over untrusted networks!

RDP (Remote Desktop Protocol)

Purpose: Graphical remote desktop access for Windows systems.

Port: `TCP 3389`

How it works: Provides full GUI access to a remote Windows computer.

Example: IT support remotely accessing a user's Windows desktop to troubleshoot.

VNC (Virtual Network Computing)

Purpose: Cross-platform graphical remote desktop access.

Port: `TCP 5900` (and higher for multiple sessions)

How it works: Platform-independent remote desktop sharing.

Example: Accessing a Linux desktop from a Windows machine.

Network Management Protocols

SNMP (Simple Network Management Protocol)

Purpose: Monitoring and managing network devices.

Port: `UDP 161` (agent), `UDP 162` (trap)

How it works: Collects information from network devices (routers, switches, servers) for monitoring.

Example: Network monitoring software tracking bandwidth usage and device health.

SNMP Versions

- **SNMPv1:** Original version, basic security (community strings)
- **SNMPv2c:** Improved with better error handling
- **SNMPv3:** Enhanced security with authentication and encryption

NTP (Network Time Protocol)

Purpose: Synchronizing clocks across network devices.

Port: `UDP 123`

How it works: Clients query NTP servers to get accurate time and sync their clocks.

Example: All computers in an organization maintaining the exact same time.

> **Why NTP Matters:** Accurate time is crucial for logging, security certificates, database transactions, and troubleshooting network issues.

Database Protocols

MySQL

Purpose: MySQL database server connections.

Port: `TCP 3306`

How it works: Applications connect to MySQL server to query and manage databases.

Example: A web application connecting to its MySQL database backend.

PostgreSQL

Purpose: PostgreSQL database server connections.

Port: `TCP 5432`

How it works: Clients connect to PostgreSQL for data operations.

Example: Enterprise applications using PostgreSQL for complex queries.

Microsoft SQL Server

Purpose: MS SQL Server database connections.

Port: `TCP 1433`

How it works: Applications connect to SQL Server for data management.

Example: .NET applications connecting to SQL Server databases.

MongoDB

Purpose: MongoDB NoSQL database connections.

Port: `TCP 27017`

How it works: Applications connect to MongoDB for document-based data storage.

Example: Modern web apps using MongoDB for flexible JSON-like data storage.

Web & Application Protocols

LDAP (Lightweight Directory Access Protocol)

Purpose: Accessing and maintaining directory information services.

Port: `TCP 389` (standard), `TCP 636` (LDAPS with SSL)

How it works: Queries and modifies directory services like Active Directory.

Example: Corporate login systems authenticating users against Active Directory.

SMB (Server Message Block)

Purpose: File sharing and printer sharing in Windows networks.

Port: `TCP 445` (SMB), `TCP 139` (NetBIOS)

How it works: Enables sharing files, printers, and other resources on a network.

Example: Accessing shared folders on Windows network drives.

NFS (Network File System)

Purpose: File sharing in Unix/Linux environments.

Port: `TCP/UDP 2049`

How it works: Allows mounting remote directories as if they were local.

Example: Linux servers sharing storage across the network.

SIP (Session Initiation Protocol)

Purpose: Initiating, maintaining, and terminating VoIP calls.

Port: `TCP/UDP 5060` (unsecured), `TCP 5061` (TLS)

How it works: Sets up voice/video calls over IP networks.

Example: VoIP phone systems, video conferencing applications.

RTP (Real-time Transport Protocol)

Purpose: Delivering audio and video over IP networks.

Port: `UDP 5004` (variable, typically even ports from 16384-32767)

How it works: Transports real-time media with minimal delay.

Example: Streaming audio/video in video conferencing and live broadcasts.

Security Protocols

SSL/TLS (Secure Sockets Layer / Transport Layer Security)

Purpose: Encrypting data in transit over networks.

Port: Wraps other protocols (HTTPS uses 443, SMTPS uses 465, etc.)

How it works: Establishes encrypted connection between client and server using certificates.

Example: The padlock icon in your browser when visiting secure websites.

> **Note:** SSL is deprecated; TLS is the modern standard. However, "SSL" is still commonly used to refer to TLS.

IPSec (Internet Protocol Security)

Purpose: Securing IP communications by encrypting and authenticating packets.

Port: `UDP 500` (IKE), `UDP 4500` (NAT-T), Protocol 50 (ESP), Protocol 51 (AH)

How it works: Encrypts entire IP packets for secure communication.

Example: VPN connections, site-to-site encrypted tunnels.

Kerberos

Purpose: Network authentication protocol using tickets.

Port: `TCP/UDP 88`

How it works: Uses tickets to prove identity without sending passwords over the network.

Example: Windows Active Directory authentication, single sign-on systems.

Network Discovery & Services

ARP (Address Resolution Protocol)

Purpose: Maps IP addresses to MAC addresses on local networks.

Port: Layer 2 protocol (no port number)

How it works: Broadcasts "Who has this IP?" and the device responds with its MAC address.

Example: Your computer finding the MAC address of the router on your local network.

ICMP (Internet Control Message Protocol)

Purpose: Network diagnostics and error reporting.

Port: Layer 3 protocol (no port number)

How it works: Sends error messages and operational information.

Example: The `ping` command uses ICMP to test connectivity.

```
# Test connectivity to a host $ ping google.com # Trace route to destination $ traceroute google.com # Check if port is open $ telnet hostname 80
```

IGMP (Internet Group Management Protocol)

Purpose: Managing multicast group memberships.

Port: Layer 3 protocol (no port number)

How it works: Routers use IGMP to learn which devices want to receive multicast traffic.

Example: IPTV streaming to multiple devices efficiently.

Cloud & Modern Protocols

WebSocket

Purpose: Full-duplex communication over a single TCP connection.

Port: `TCP 80` (WS), `TCP 443` (WSS - secure)

How it works: Establishes persistent connection for real-time bidirectional data flow.

Example: Chat applications, live sports scores, real-time collaboration tools.

MQTT (Message Queuing Telemetry Transport)

Purpose: Lightweight messaging for IoT devices.

Port: `TCP 1883` (unencrypted), `TCP 8883` (encrypted)

How it works: Publish/subscribe messaging model for low-bandwidth, high-latency networks.

Example: Smart home devices communicating with central hub.

gRPC

Purpose: High-performance RPC (Remote Procedure Call) framework.

Port: Typically `TCP 443` (runs over HTTP/2)

How it works: Enables efficient communication between microservices.

Example: Modern cloud-native applications with microservices architecture.

REST/HTTP APIs

Purpose: Web service communication using HTTP methods.

Port: `TCP 80/443` (HTTP/HTTPS)

How it works: Uses standard HTTP methods (GET, POST, PUT, DELETE) for CRUD operations.

Example: Mobile apps fetching data from backend servers.

Proxy & Caching Protocols

SOCKS (Socket Secure)

Purpose: Proxy protocol for routing network packets.

Port: `TCP 1080` (SOCKS5)

How it works: Acts as intermediary, forwarding traffic between client and server.

Example: Bypassing firewalls, anonymizing internet traffic.

Squid/HTTP Proxy

Purpose: Web proxy and caching server.

Port: `TCP 3128` (common default), `TCP 8080`

How it works: Caches web content and filters traffic.

Example: Corporate networks caching frequently accessed websites to save bandwidth.

Quick Reference: Common Ports Summary

Essential Ports to Remember

Port	Protocol	Transport	Description
`20/21`	FTP	TCP	File Transfer Protocol (data/control)
`22`	SSH/SFTP	TCP	Secure Shell, Secure File Transfer

Port	Protocol	Transport	Description
23	Telnet	TCP	Unsecured remote access
25	SMTP	TCP	Email sending
53	DNS	UDP/TCP	Domain Name System
67/68	DHCP	UDP	Dynamic IP assignment
80	HTTP	TCP	Web browsing (unsecured)
88	Kerberos	TCP/UDP	Network authentication
110	POP3	TCP	Email retrieval
123	NTP	UDP	Time synchronization
143	IMAP	TCP	Email access
161/162	SNMP	UDP	Network monitoring
389	LDAP	TCP	Directory services
443	HTTPS	TCP	Secure web browsing
445	SMB	TCP	Windows file sharing
465	SMTPS	TCP	Secure email sending
587	SMTP	TCP	Email submission
636	LDAPS	TCP	Secure LDAP
993	IMAPS	TCP	Secure IMAP
995	POP3S	TCP	Secure POP3
1433	MS SQL	TCP	Microsoft SQL Server

Port	Protocol	Transport	Description
1883	MQTT	TCP	IoT messaging
2049	NFS	TCP/UDP	Network File System
3306	MySQL	TCP	MySQL database
3389	RDP	TCP	Remote Desktop Protocol
5060/5061	SIP	TCP/UDP	VoIP signaling
5432	PostgreSQL	TCP	PostgreSQL database
5900	VNC	TCP	Remote desktop access
8080	HTTP Alt	TCP	Alternative HTTP/Proxy
27017	MongoDB	TCP	MongoDB database

Port Ranges Explained

Range	Name	Description	Examples
0-1023	Well-Known Ports	Reserved for system services and standard protocols	HTTP (80), HTTPS (443), SSH (22)
1024-49151	Registered Ports	Assigned by IANA for specific services	MySQL (3306), RDP (3389)
49152-65535	Dynamic/Private Ports	Used for temporary or private purposes	Client-side ports, custom applications

Best Practices & Security Tips

✓ Security Best Practices

- **Always use encrypted protocols** - HTTPS over HTTP, SSH over Telnet, FTPS/SFTP over FTP
- **Change default ports** - Reduces automated attacks (e.g., SSH on port 2222 instead of 22)
- **Implement firewall rules** - Only open ports that are absolutely necessary
- **Keep services updated** - Patch vulnerabilities regularly
- **Use strong authentication** - SSH keys, multi-factor authentication
- **Monitor open ports** - Regular security audits using tools like nmap
- **Disable unused services** - Close ports for services you don't need

⚠ Common Security Risks

- **Open unnecessary ports** - Each open port is a potential entry point
- **Using default credentials** - Always change default usernames/passwords
- **Unencrypted protocols** - FTP, Telnet, HTTP transmit data in plain text
- **Outdated software** - Old versions may have known vulnerabilities
- **No firewall** - Exposing services directly to the internet

Troubleshooting Commands

```
# Check which ports are listening $ netstat -tuln $ ss -tuln # Scan ports on a remote host $ nmap hostname $ nmap -p 1-1000 hostname # Test specific port connectivity $ telnet hostname 80 $ nc -zv hostname 80 # Check DNS resolution $ nslookup google.com $ dig google.com # View routing table $ route -n $ ip route show # Test connectivity $ ping -c 4 google.com $ traceroute google.com # Check firewall rules (Linux) $ sudo iptables -L $ sudo ufw status # Windows equivalent > netstat -ano > Test-NetConnection hostname -Port 80
```

Key Takeaways

Remember These Core Concepts:

- **TCP is reliable, UDP is fast** - Choose based on your needs
- **Lower port numbers (0-1023) are for system services** - Require admin privileges
- **Always prefer secure variants** - HTTPS, SFTP, SSH, IMAPS, etc.
- **Ports are just logical endpoints** - Multiple services can run on one IP with different ports
- **Firewalls control port access** - Essential for security
- **Some protocols use multiple ports** - FTP (20, 21), DHCP (67, 68)
- **Layer matters** - Some protocols operate at Layer 2/3 (ARP, ICMP) without port numbers

---

Additional Resources

For more information, refer to:

- IANA Port Number Registry
- RFC Documents for protocol specifications
- Network+ and CCNA certification materials
- Wireshark for packet analysis

Created for networking professionals and students

Keep this guide handy for quick reference!