
📡 Protocols in the Client–Server Model

The **Client–Server model** is based on a **request–response** cycle:

- **Client** → sends a request (using a protocol).
- **Server** → processes request and sends a response (using the same protocol).

Different **layers of the network stack** (OSI model or TCP/IP model) have their own protocols.

🔹 1. Application Layer Protocols

These are **closest to the client and server applications**.

- **HTTP/HTTPS (HyperText Transfer Protocol / Secure)**
  - Most common for web apps.
  - Client (browser, app) sends GET, POST, etc. requests.
  - Server responds with HTML, JSON, XML, or files.
  - HTTPS adds **TLS encryption** for security.
- **FTP (File Transfer Protocol)**
  - Used for transferring files between client and server.
- **SMTP (Simple Mail Transfer Protocol)**, **IMAP**, **POP3**

    - Used in email systems.
    - Client email app ↔ Mail server.

- **DNS (Domain Name System)**
    - Client asks: “What is the IP of www.google.com?”
    - Server responds with IP address.

- **WebSockets (WS/WSS)**
    - Persistent, two-way communication.
    - Used in chat apps, stock updates.

- **gRPC (Google Remote Procedure Call)**
    - High-performance, binary protocol for client–server communication in microservices.

- **MQTT (Message Queuing Telemetry Transport)**
    - Lightweight publish–subscribe protocol, used in IoT.

🔹 2. Transport Layer Protocols

Handle **how data moves** between client and server.

- **TCP (Transmission Control Protocol)**
    - Reliable, ordered, connection-oriented.
    - Used by HTTP, HTTPS, FTP, SMTP, gRPC, WebSockets.

- **UDP (User Datagram Protocol)**
    - Faster, connectionless, less reliable.
    - Used for real-time apps: video calls (VoIP), gaming, streaming.
- **QUIC (Quick UDP Internet Connections)**
    - Google's protocol → combines UDP's speed + TCP's reliability.
    - Basis for **HTTP/3**.

🔹 3. Network Layer Protocols

Handle **addressing and routing** between client and server.

- **IP (Internet Protocol – IPv4/IPv6)**
    - Assigns unique addresses to clients and servers.
- **ICMP (Internet Control Message Protocol)**
    - Used for error reporting & diagnostics (ping, traceroute).
- **BGP (Border Gateway Protocol)**
    - Decides how data is routed between networks/ISPs.

🔹 4. Security Protocols

Ensure **safe communication** between client and server.

- **TLS/SSL (Transport Layer Security / Secure Sockets Layer)**
    - Encrypts traffic (basis of HTTPS, WSS, SMTPS).
- **SSH (Secure Shell)**
    - Secure remote login from client to server.
- **IPSec (Internet Protocol Security)**
    - Encrypts traffic at the network layer (VPNs).
- **OAuth, SAML, Kerberos**
    - Authentication & identity protocols (who the client is).

🔹 5. Example – Web Application (Client–Server)

When you open **www.example.com** in your browser:

1. **DNS (UDP/TCP)** → Resolve domain name to IP.
2. **TCP (Transmission Control Protocol)** → Establish connection to server.
3. **TLS (Transport Layer Security)** → Secure connection (if HTTPS).
4. **HTTP/HTTPS** → Client requests a webpage → Server responds with HTML.

🔹 6. Example – Chat Application

1. **TCP connection established**.

2. **WebSocket protocol** upgrades HTTP → persistent connection.
3. **TLS** secures it (WSS).
4. Messages flow in **real time** between client ↔ server.

✅ Summary

In the **Client–Server model**, multiple protocols work together:

- **Application Layer** → Defines *what* the client/server say (HTTP, gRPC, DNS, SMTP).
- **Transport Layer** → Defines *how* data is delivered (TCP, UDP, QUIC).
- **Network Layer** → Defines *where* it goes (IP, BGP).
- **Security Layer** → Ensures *safe communication* (TLS, SSH, IPSec).