# Network communication models

**Written by [Febogamedev]**

Data travels across a network in the form of packets.
Now let us examine the models used to exchange these packets between two applications.

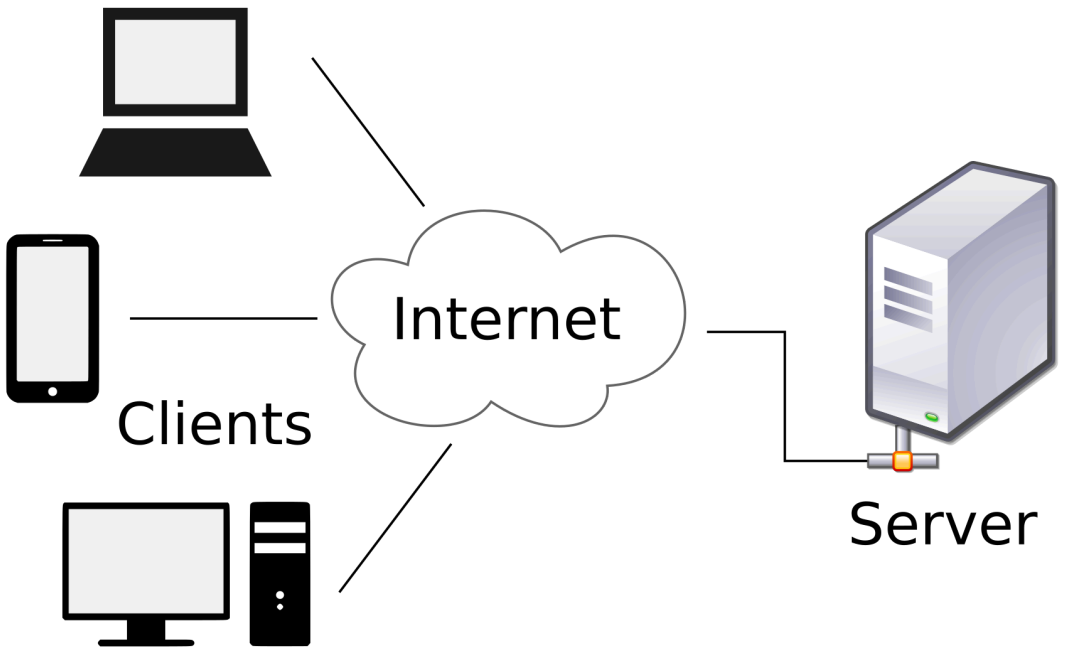
- Client-server model.
- Peer-to-peer model.

## Client-server model

**The client-server** model is a network architecture in which a device or software process, called a client, sends requests to another device or software process, called a server, in order to obtain a service Kurose & Ross, 2026; Tanenbaum et al., 2022).

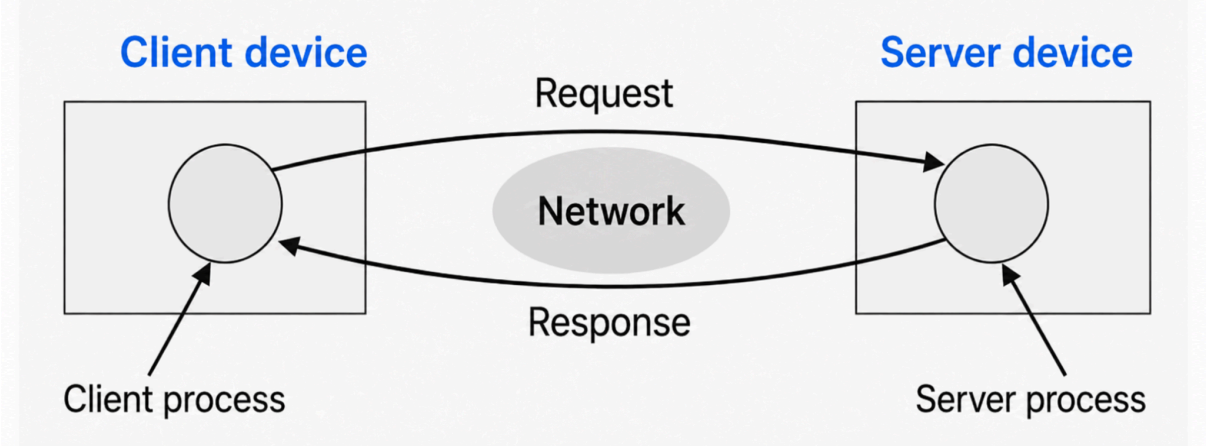
The presence of a server allows a number of clients to share its resources, while the server manages access to those resources in order to prevent usage conflicts.

Many services available on the Internet use the client-server model. For example, when we visit a website, the browser sends a request to a web server, which returns the requested page.




More specifically, the client-server model involves two processes, meaning running programs: one on the client machine and one on the server machine.
Based on these premises, communication follows this logic:

- A client process sends a message through the network to the server process and then waits for a response message.
- When the server process receives the request, it performs the requested operation and returns a response.




It is important to note that a server is not necessarily a physical machine: it can also be a software process that listens on a network port while waiting for client requests. The same machine can therefore host several servers at the same time, meaning several running applications or services, each reachable by clients (Kurose & Ross, 2026).
A client is not necessarily a physical machine either: it can be a software process that sends requests to a server to obtain a service. The same machine can run several client processes at the same time, such as a web browser, an email app, an online game, or a messaging app.

Clients and servers communicate through a **communication protocol**: a set of shared rules that enables them to interact for a specific purpose (Kurose & Ross, 2026).

## Client

The term **client** refers to a computer component or subsystem that accesses the services or resources of a server.
A client can be hardware or software:

- A device connected to a server through a network and requesting one or more services through one or more network protocols is an example of a hardware client.
- An email program is an example of a software client because it continually communicates with the server that receives and sends email messages.

For example, one of the most important software clients is the **browser**, a program specialized in displaying web pages obtained from a server.

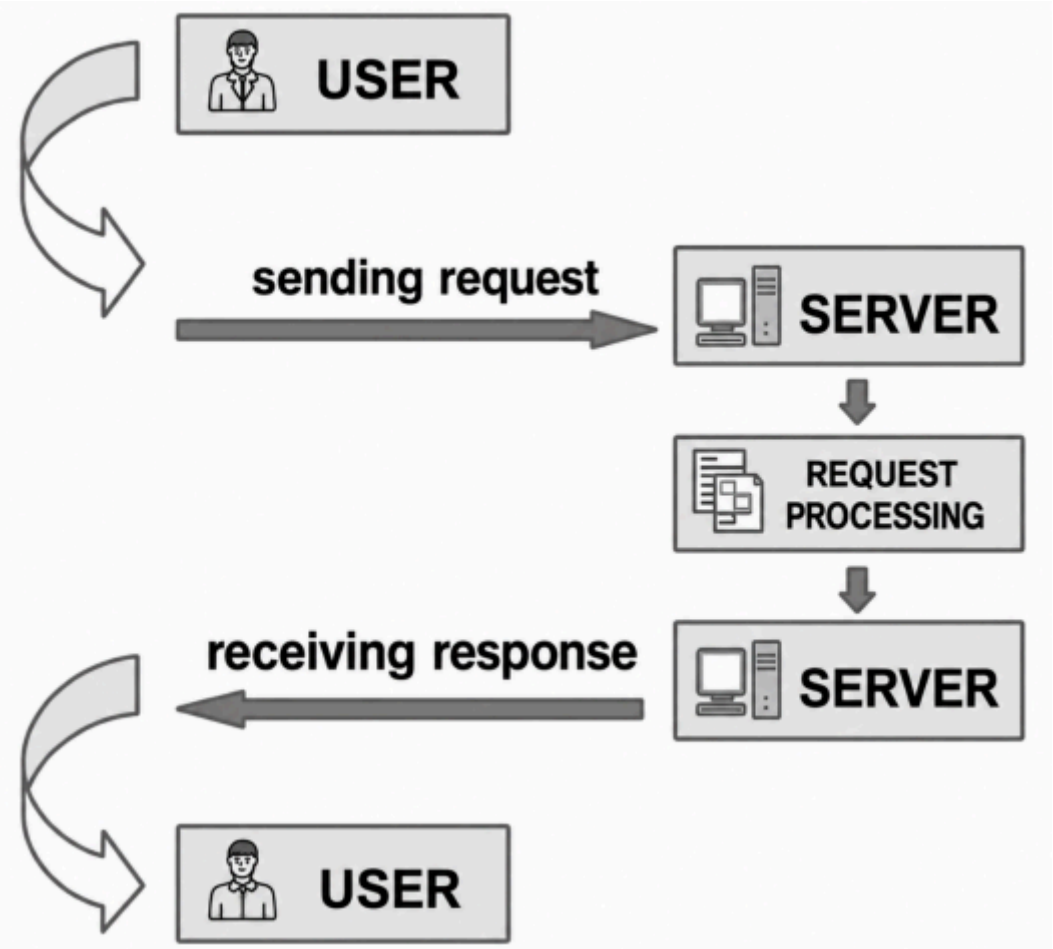
In some systems, the client mainly acts as an interface to the server; in others, it can be highly complex software capable of processing large amounts of data locally.

Some systems, such as email, are divided into a client component, running on the client computer, and a server component, running on the server.

***Example***
When using Gmail via a browser or app, the user interface acts as the client. The messages, however, are stored on the service's server systems.
The client sends requests to the server, which retrieves the data and returns it for display to the user.

## Server

A server, from the verb “to serve”, is a device or software program that provides services to other devices or programs, called clients (Tanenbaum et al., 2022).

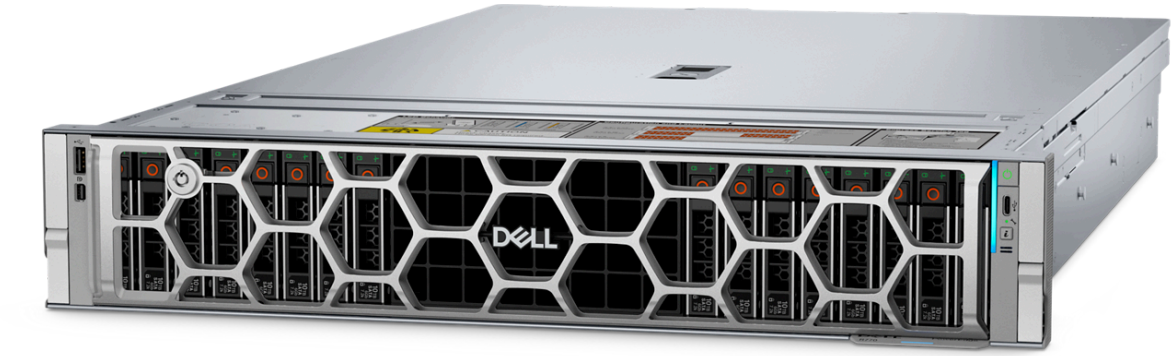



Depending on the context, the term “server” can therefore refer to:

- An ordinary computer used to provide services to other computers, regardless of its hardware characteristics.
- A specialized computer designed for use as a server, characterized by high reliability, greater performance, and additional features.

- Software that provides services to other software.

A server may manage user access, allocate and release resources, share data, and protect information.




*Typical server*

***Example***
An email server can be compared to a post office.
Users must be authorized before they can access their email account through a client. Similarly, a user must have the key to the mailbox located at a post office in order to collect their mail.

## Types of servers

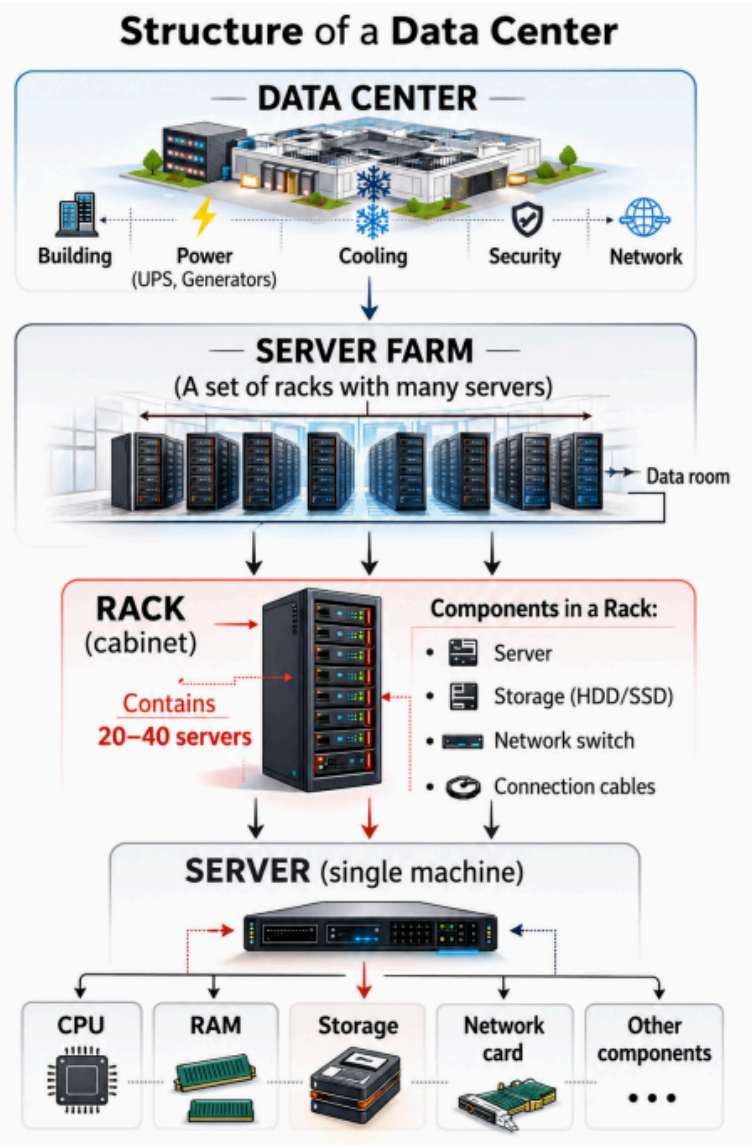
In many cases, when we talk about servers, we are referring to software services: programs that perform specific functions and respond to client requests.
The most important services are:

- File server: allows users to access files stored on a storage device as if they were on their own device, making information sharing easier.
- Database server: manages databases and responds to requests to read, write, modify, or delete data.
- FTP server: provides network access to public or authenticated folders.
- Web server: used to host websites.
- Application server: executes the logic of a web application, for example by managing functions, users, data, and communication between the interface and the database.
- Mail server: manages email.
- Game server: hosts resources that make multiplayer games possible.

## Data center

**A data center** is a specialized facility designed to house servers and all the components connected to them, including storage devices, networking equipment, security systems, and power infrastructure (Barroso et al., 2026).
A data center stores, processes, and distributes large volumes of data continuously, reliably, and securely.
In essence, data centers are “information factories”: places where data lives, moves, and is managed 24 hours a day.

A fundamental part of a data center is **the server farm**, an organized collection of servers installed in the same physical location.
A server farm consists of metal cabinets called **racks**, each of which contains dozens of servers.




These servers host websites, databases, business applications, streaming video, and even the virtual worlds of online games.
In addition to the server farm, a data center includes:

- Cooling systems to keep temperatures within acceptable limits, since servers generate a great deal of heat.
- Uninterruptible power supplies (UPSs) and generators to maintain power during outages.
- Very high-speed fiber-optic connections.
- Physical security systems and constant digital monitoring to prevent unauthorized access or failures.

The entire system is designed to provide high availability, meaning continuous operation even under critical conditions (Barroso et al., 2026).
An interruption lasting only a few seconds can cause enormous financial losses or reputational damage for digital service providers.

Without data centers, the Internet as we know it would not exist.
Most digital services we use every day, such as email, messaging, video streaming, social networks, and cloud platforms, rely on one or more data centers.
Corporate information systems, banks, hospitals, navigation apps, and digital public services also rely on these infrastructures.


*Rack containing servers*

# Peer-to-peer system

The client-server model is the most widely used on the Internet, but another popular model also exists: **peer-to-peer** (*P2P*).

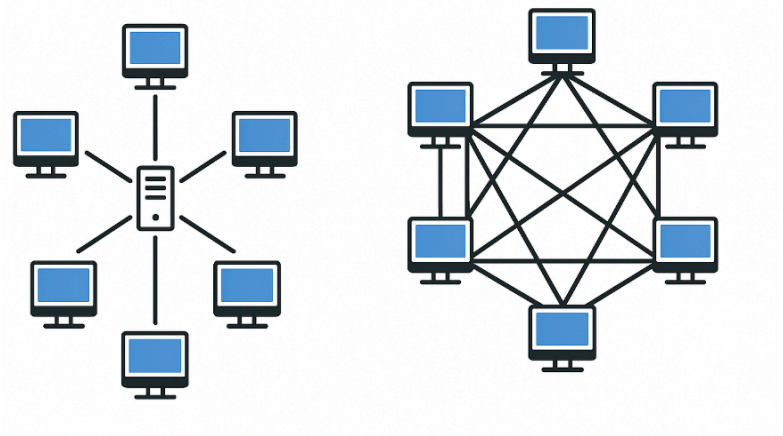
A peer-to-peer network is a network model in which connected devices, called **peers**, can communicate directly with one another without necessarily depending on a central server (Kurose & Ross, 2026; Tanenbaum et al., 2022).
The English word peer means “equal”. This helps explain the main concept: in a peer-to-peer network, computers are not rigidly divided into clients and servers, but can perform both roles.
A device can therefore:

- Request data from another device.
- Send data to other devices.
- Share files, resources, or services.
- Participate in the overall operation of the network.

Each peer can act as a client when requesting a resource and as a server when providing one.

In the client-server model, communication is organized around one or more main servers. Clients do not always communicate directly with one another and often communicate through the server.
For example, when we visit a website, our browser is the client and the web server is the system hosting the requested pages. The client sends a request, and the server responds by sending the necessary data.

In the peer-to-peer model, however, devices can exchange data directly.
There is not necessarily a central server controlling all communication.
This is the most important difference: the client-server model is more centralized, whereas the peer-to-peer model is more distributed (Kurose & Ross, 2026).



*A client-server system on the left and a P2P system on the right*

Each peer can make some of its resources available, for example files, storage space, processing power, network bandwidth, and information useful to the operation of the system. For this reason, the overall operation of a peer-to-peer network depends on cooperation among many peers.

***Example***
Imagine a classroom in which every student has a different file on their computer.

- In the client-server model, all files would be uploaded to a single central computer, such as the teacher's computer or a school server. Students would download the files from there.
- In the peer-to-peer model, however, each student could share their file directly with the others. If a student needs a file, they can receive it directly from a classmate's computer.

Not all peer-to-peer networks work in the same way. In simplified terms, we can distinguish between pure **P2P networks** and **hybrid P2P networks**.

- In a pure P2P network, there is no central server. All peers have a similar role and cooperate directly with one another. The network is completely distributed.
- In a hybrid P2P network, by contrast, a central server may help peers find one another, while the actual exchange of data takes place directly between devices. For example, a server might store a list of available peers without directly hosting all the files being exchanged. In this case, the server does not remove the peer-to-peer nature of the network, because the main data is still transferred between peers (Tanenbaum et al., 2022).

One of the best-known uses of peer-to-peer networks is file sharing.
In a *P2P* file-sharing system, a file can be divided into many parts.
Each peer can download different parts of the file from different peers and, at the same time, share with others the parts it already has.
This mechanism is very different from traditional downloading from a single server. In the client-server model, if many users download the same file from the same server, that server can become overloaded. In the *P2P* model, by contrast, as more users join the network, the number of possible download sources also increases (Kurose & Ross, 2026).
For this reason, *P2P* can be very efficient for distributing large amounts of data.

The peer-to-peer model offers several advantages:

- Load distribution: because there is not necessarily a single central server, the workload can be divided among many devices.
- Scalability: in some P2P systems, when new peers join the network, not only the number of requests increases, but also the available resources. Each new peer can contribute bandwidth, data, or processing capacity (Kurose & Ross, 2026).
- Fault tolerance: if a central server stops working, a client-server system may become unreachable. In a P2P network, however, the failure of a single peer does not necessarily bring down the entire network (Tanenbaum et al., 2022).

However, the peer-to-peer model also has some disadvantages:

- Management and control: in a client-server system, it is easier to control who can access the data, which resources are available, and how they are managed. In a P2P network, nodes are distributed and can join or leave the network at any time.
- Security: because data can come from different peers, it is important to verify that it has not been altered and does not contain malicious software.

- Resource availability: if only a few peers share a particular file, that file may become difficult to find. In addition, if many peers disconnect, the quality of service may decrease.

One of the fundamental questions in a peer-to-peer network is: how does a peer find the other peers?
In the client-server model, the client knows the server's address, for example through a URL or an IP address. In the P2P model, however, peers may be numerous, change over time, and connect from different networks.

To solve this problem, P2P networks can use different discovery mechanisms.
- Some use support servers that help peers obtain a list of other available peers.
- Other networks use more distributed systems. In this case, each node knows only part of the network and can point to other nodes to contact. When a new peer joins the network, it starts from a few already known nodes. From them, it obtains information about other peers and gradually expands its knowledge of the network.

# Torrent

One of the best-known examples of peer-to-peer technology is **BitTorrent**.
In everyday language, however, the word "**torrent**" is often used to describe a download system based on the BitTorrent protocol.

BitTorrent is a communication protocol designed to distribute files among many devices connected to a network (Cohen, 2008).
Unlike traditional downloading, in which a client downloads a file from a single server, a torrent system distributes the file among many users.
Every user participating in the exchange is a peer. This means that each peer can both receive data and send it to others.

Traditional downloading works quite simply: the client sends a request to a server, and the server sends the requested file.
For example, if we download a program from the manufacturer's official website, our computer connects to the company's server and receives the complete file from it.

In a torrent, however, the file does not necessarily come from a single server. It is divided into many small parts, called pieces or blocks. Our computer can download some pieces from one peer, others from another peer, and so on (Cohen, 2008).

At the same time, while downloading, we can share the pieces we have already received with other users.
This is the key point: in a torrent system, users who download can also help distribute the file.

A torrent download is often started using a small file with the ".torrent" extension.
This file does not contain the actual content we want to download. For example, if we want to download a video, the .torrent file does not contain the complete video.

Instead, the .torrent file contains information useful to the torrent program, such as:

- The name of the file to download.
- The size of the file.
- How the file is divided into pieces.
- Information used to verify that the downloaded pieces are correct.
- Instructions on how to find other peers.

A .torrent file is therefore a kind of “map” that tells the program what to look for and how to reconstruct the final file (Cohen, 2008).

Using a torrent requires a dedicated program called a **torrent client**.
Examples of torrent clients include qBittorrent, Transmission, µTorrent, and BitTorrent Web.

The torrent client reads the “.torrent” file or a link called a **magnet link**, searches for available peers, and begins downloading the pieces of the file (Hazel & Norberg, 2008).
The client also reassembles the downloaded pieces in the correct order until the complete file is obtained.

A magnet link is a link containing the essential information needed to identify the file to be downloaded. Instead of first downloading a “.torrent” file, the user can open the magnet link directly with a torrent client (Hazel & Norberg, 2008).
The client uses this information to find other peers and retrieve the data needed to begin the download.
From the user’s point of view, a magnet link is often more convenient because it only requires clicking a link.

Two terms are often used in torrent terminology: **seed** and **leech**.

- A seed is a peer that already has the complete file and is sharing it with others.
- A leech, or more precisely a peer that is still downloading, is a user who has not yet obtained the complete file. Even so, the peer can share with others the pieces it has already received.

The more seeds there are, the easier and faster it is to download the file, because more complete copies are available on the network.
If there are only a few seeds, the download may be slow or may stop because some pieces of the file can be difficult to find.

The set of peers sharing the same file is called a **swarm**.

A swarm therefore consists of all the users participating in the distribution of a specific piece of content: some have the complete file, while others have only part of it.

Torrent systems mainly use two methods to find the peers in the swarm for a specific piece of content: **trackers** and **DHT**.

A tracker is a server that helps peers find one another. It does not necessarily contain the file being downloaded, but it keeps information about the peers participating in the swarm. When

a torrent client wants to download a file, it can contact the tracker to obtain a list of other available peers (Cohen, 2008).

DHT stands for Distributed Hash Table. It is used to find other peers without relying on a single central server.
Instead of asking a tracker who is sharing a file, the torrent client queries a distributed network made up of many peers.
We can imagine it as a large directory distributed across many computers: each peer knows only part of the information, but by cooperating with the others it makes it possible to find the peers sharing a particular torrent.
Using DHT, the client does not query every peer in the network.
Instead, it uses the torrent information to progressively contact nodes in the distributed network until it finds peers sharing that content (Loewenstern & Norberg, 2008).

# Supplementary materials

## Key points

- In the **client-server model**, the client sends requests and the server provides services or resources.
- Clients and servers are primarily programs in execution, and they communicate via shared protocols.
- There are various types of servers: web, file, database, mail, application, and video game servers.
- **Data centers** house servers, storage systems, networks, cooling systems, and backup power supplies.
- In the **peer-to-peer (P2P)** model, each device can request and provide resources directly to other devices.
- P2P networks can be either **pure** or **hybrid**, with servers used only to help peers find one another.
- P2P distributes the workload and is fault-tolerant, but it is more difficult to control and secure.
- **BitTorrent** divides a file into pieces that are downloaded and shared simultaneously among multiple peers.
- A **.torrent** file or **magnet link** contains the information needed to identify and download the content.
- A **seed** has the complete file, a **leech** is downloading it, and the group of peers forms a swarm.
- Peers are found through either a central **tracker** or the distributed DHT network.

## Exercises

**What Is the Client Actually Doing?**
A student is using the following applications simultaneously on their computer:

- A browser to visit the school website.
- An email client.
- An online video game.
- A messaging application.

The student says:
“My computer is the client, and all the computers it connects to are servers.”

Analyze this statement.

- Explain why simply referring to “client computers” and “server computers” can be imprecise.
- Identify which client processes might be running simultaneously on the student’s computer.
- Explain how a single machine can run multiple clients at the same time.

- Choose one of the applications listed above and describe the sequence request → processing → response.
- Explain the role of the communication protocol between client and server.

**Designing a School's Services**
A school wants to build its own IT infrastructure.
It needs to provide the following services:
- Store documents shared among teachers.
- Host the school website.
- Manage a database containing students, classes, and grades.
- Manage internal email.
- Run the logic of a new school web application.

For each service:
- Identify the most appropriate type of server among those described in the course materials.
- Explain which service it provides to clients.
- Give an example of a possible client that could use it.
- Explain why five different physical computers are not required in order to have five different servers.
- Describe a possible situation in which a single computer performs multiple server functions at the same time.

**A Service That Cannot Stop**
A streaming platform is used by millions of users. Its managers are considering hosting all the required services on several servers installed in a normal room at the company.
The room has an Internet connection and electrical power, but it has no dedicated cooling systems, generators, UPS systems, or advanced security systems.

Analyze the proposed solution.
- Identify at least four problems that could occur.
- Explain why a simple collection of servers does not necessarily constitute an adequate infrastructure for a service of this scale.
- Propose which data center components would be necessary to improve the reliability of the service.
- Explain the role of server farms and racks within the infrastructure.
- A blackout lasts five minutes: explain which systems should allow the service to continue operating.
- Explain why ensuring high availability is important for a platform of this kind.

**Client-Server or Peer-to-Peer?**
A school needs to distribute a 4 GB video file to 500 students.
Two solutions have been proposed.
- Solution A: all students download the file from a single school server.
- Solution B: initially, some students receive parts of the file, and then students can directly exchange with one another the parts they already possess.

Compare the two solutions.

- Identify which solution mainly follows the client-server model and which follows the peer-to-peer model.
- Describe how the file would travel in the two cases.
- Explain what might happen to the server in Solution A if all students started the download at the same time.
- Explain why, in Solution B, the arrival of new peers can also increase the available resources.
- Identify at least one advantage and one disadvantage of Solution B.
- Choose which solution you would use and justify your choice by considering load, reliability, control, and security.

**Pure P2P or Hybrid P2P?**
An application allows users to share files directly between their computers.
When a new user opens the program, however, the software contacts a central server that provides a list of other available users. Once the other devices have been found, the files are transferred directly between users and do not pass through the central server.

A student says:
“It cannot be peer-to-peer because there is a server.”

Evaluate this statement.
- Explain whether the system described can be considered P2P.
- Determine whether it is more precisely a pure P2P or hybrid P2P system.
- Describe the role of the central server.
- Describe the role of the peers.
- Explain what would change if the files were also stored on the server and all users had to download them exclusively from it.
- Now imagine a system with no central server at all: describe which problem would still need to be solved in order for the peers to communicate with one another.

**Analyzing a Torrent Download**
You want to legally download an ISO image of a Linux distribution using BitTorrent.
You open a magnet link with a torrent client. The program finds a swarm consisting of:
- 8 seeds.
- 35 peers that are still completing the download.

During the download, your computer receives different pieces of the file from different peers while simultaneously sending other users some of the pieces it already has.

Analyze what is happening.
- Explain why your computer is acting as both a client and a server at the same time.
- Explain the difference between seeds, peers that are still downloading, and the swarm.
- Describe why the file can be received from multiple devices rather than from a single server.
- Explain how the torrent client can verify that the received pieces are correct and reconstruct the final file.

- Explain what a .torrent file is used for and why it does not necessarily contain the file you want to download.
- Explain the role that a tracker can play.
- Explain how DHT can instead be used to find peers without depending on a single central server.
- Imagine that all seeds disconnect while none of the remaining peers possesses a particular piece of the file: predict what might happen to the download and justify your answer using the principles of P2P.

# Bibliography


Barroso, L. A., Hölzle, U., & Ranganathan, P. (2026). *The data center as a computer: Designing warehouse-scale machines* (4th ed.). Springer. https://doi.org/10.1007/978-3-031-99489-0

Cohen, B. (2008). *The BitTorrent protocol specification* (BEP 3). BitTorrent Enhancement Proposals. https://www.bittorrent.org/beps/bep_0003.html

Hazel, G., & Norberg, A. (2008). *Extension for peers to send metadata files* (BEP 9). BitTorrent Enhancement Proposals. https://www.bittorrent.org/beps/bep_0009.html

Kurose, J. F., & Ross, K. W. (2026). *Computer networking: A top-down approach* (9th ed.). Pearson.

Loewenstern, A., & Norberg, A. (2008). *DHT protocol* (BEP 5). BitTorrent Enhancement Proposals. https://www.bittorrent.org/beps/bep_0005.html

Tanenbaum, A. S., Feamster, N., & Wetherall, D. J. (2022). *Computer networks* (6th ed.). Pearson.


# Further reading

- James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach*, 9th ed., Pearson, 2026 — Provides a clear introduction to network applications and communication architectures, including client-server and peer-to-peer models, application processes, protocols, and P2P file distribution. It is particularly useful for placing these communication models within the broader architecture of the Internet.
- Andrew S. Tanenbaum, Nick Feamster, and David J. Wetherall, *Computer Networks*, 6th ed., Pearson, 2022 — Offers a systematic treatment of computer networks and network applications, helping deepen the distinction between centralized client-server systems and distributed peer-to-peer architectures, as well as the role of servers, protocols, and networked applications.
- Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan, *The Data Center as a Computer: Designing Warehouse-Scale Machines*, 4th ed., Springer, 2026 — Provides a specialized exploration of modern data centers, treating them as large-scale computing systems and examining their architecture, hardware and software organization, operation, reliability, and design. It is especially useful for expanding the chapter's discussion of server farms and data-center infrastructure.

- Ralf Steinmetz and Klaus Wehrle, eds., *Peer-to-Peer Systems and Applications*, Springer, 2005 — Provides an extensive treatment of peer-to-peer architectures, including unstructured and structured P2P systems, peer discovery, Distributed Hash Tables, load balancing, reliability, and peer-to-peer applications. Despite its age, it remains particularly useful for understanding the architectural principles behind DHT-based systems.
- Larry L. Peterson and Bruce S. Davie, *Computer Networks: A Systems Approach*, 6th ed., Morgan Kaufmann/Elsevier, 2021 — Explores networking from a systems perspective, showing how protocols, applications, hosts, and network components interact within larger distributed systems. It is useful for connecting client-server and peer-to-peer communication models with the broader principles of network architecture and protocol design.

# Editorial notes

In producing these materials, I have used generative artificial intelligence tools to support the writing process, particularly to improve the form of the text, reorganize content, refine wording, and speed up certain editorial tasks.

As I work independently on the production of these materials, I try to automate any activities that can reasonably be automated, so that I can devote more time to research, design, and content development.

However, artificial intelligence does not determine the content of this work: the choice of topics, structure, ideas, interpretations, examples, and teaching approach are developed by me. AI is therefore used primarily as a tool to support production and formal revision, while authorship and responsibility for the design and content remain mine.