Aufgabenteilung in Netzwerken

Client-Server Prinzip

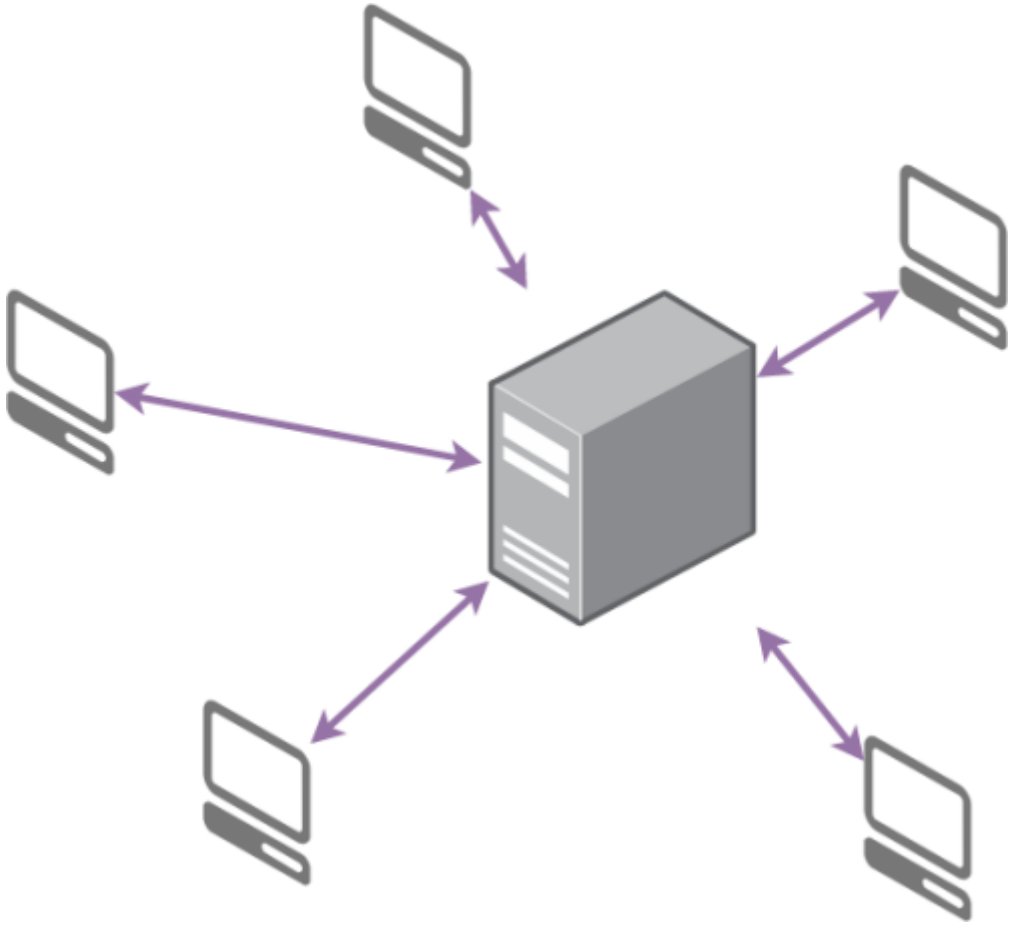
Im Abschnitt Kommunikationsarten wurde bereits ein denkbare Aufgabenteilung in Netzwerken beschrieben: Die Client-Server Architektur.

Ein **Server** ist ein Programm, das Dienste zur Verfügung stellt, häufig wird es auf einem Server (Gerät) dauerhaft im Hintergrund ausgeführt.

Ein **Client** ist ein Programm, das auf die Dienste eines Serverprogramms zugreift. Es wird meist auf dem Clientgerät ausgeführt, z.B. dem Laptop oder Smartphone, mit dem ein Benutzer mit Netzwerkdiensten in Interaktion tritt.

"Client" und "Server" sind also mehrdeutige Begrifflichkeiten, sie können sowohl die Programme als auch die Geräte bezeichnen.

Für zahlreiche Anwendungsfälle ist es sinnvoll, eine Client-Server-Architekur anzustreben: Beispielsweise möchte man jederzeit Mails empfangen können, auch wenn man gerade nicht am Laptop sitzt. Es macht daher Sinn, diese Aufgabe an einen Mailserver zu delegieren, mit dem der Clientcomputer nur dann interagiert, wenn er prüft, ob neue Mails angekommen sind.



Andere Beispiele sinnvoller Client-Server Anwendungen sind:

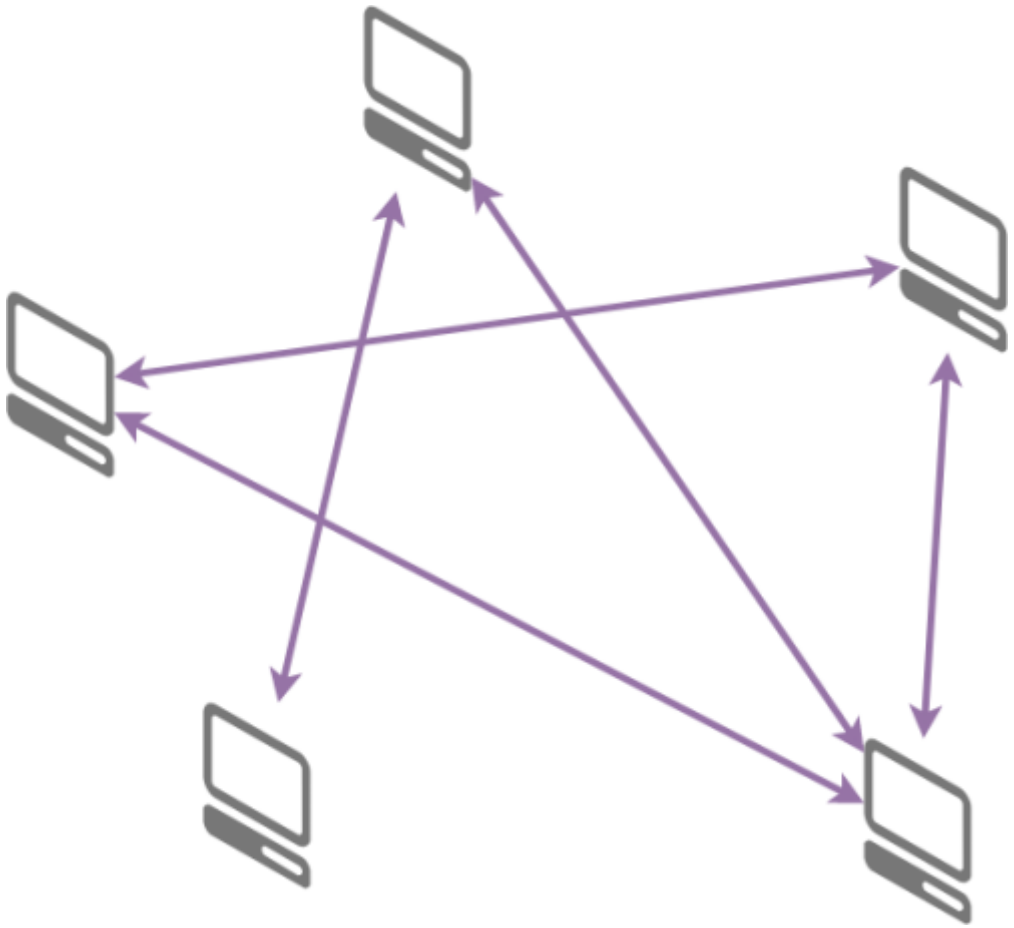
- **Datenbankserver**: Zentrale Datenhaltung, regelbasierter Zugriff mit Clientprogrammen.
- **Proxyserver**: Vermittler zwischen dem Clientprogramm und dem Kommunikationsziel. Zur Zwischenspeicherung von Inhaltzen oder der Umgehung von Netzwerkrestriktionen.
- **Webserver**: Stellen Webseiten zur Verfügung, erlauben den Zugriff auf APIs von Webanwendungen und Apps.
- **Printserver**: Verteilen und managen Druckaufträge in einem Netzwerk.
- * **Cloud-Server**: Server die Ressourcen wie Rechenkapazität, Speicherplatz und Netzwerkbandbreie zur Verfügung stellen, ohne dass der Anwender tatsächliche Servergeräte betreiben muss. (AWS, Azure u.ä.)

Nachteile:

- Die **Zentralisierung** schafft potentiell einen "Single Point Of Failure" - wenn das Serversystem nicht mit genügend Redundanz versehen ist, führt ein Ausfall des Servers dazu, dass unter Umständen sehr viele Clientgeräte und Dienste ohne Funktion sind. Wenn der Datenbankserver ausfällt, kann keine der Sekretärinnen mehr auf die Schülerdaten zugreifen, gleichgültig wie viele funktionierende PCs im Sekretariat vorhanden sind.
- Die Hardware für Servergeräte muss je nach Anwendungsfall sehr leistungsfähig sein, um die Last zahlreicher gleichzeitiger Zugriffe performant verarbeiten zu können.
- Die Bündelung weltweiterRechenkapazitäten in Cloud-Server-Rechenzentren hat in der Vergangenheit bei Ausfällen in Amazon, Google oder Microsoft Rechenzentren teilweise dazu geführt, dass große Teile des Internets (Netflix, Spotify u.ä.) ohne Funktion waren.

Peer-to-Peer Architektur

Anders als innerhalb einer Client-Server-Architektur sind in einem Peer-to-Peer Netz alle Geräte "gleichberechtigt": Auf allen Geräten werden Programme ausgeführt, die sowohl Dienste zur Verfügung stellen als auch diese Dienste nutzen. Jedes Gerät ist also Client und Server zugleich.



Beispiele für Dienste und Programme die eine4 Peer-to-Peer Architektur verwenden sind:

- Briar: Ein dezentraler verschlüsselter Messenger.
- Bittorrent: Dateiverteilungssoftware, kann sehr effizient große Daten mengen verteilen.
- Gnutella: In Filius ist ein etwas in die Jahre gekommenes P2P Protogramm implementiert, das zum Dateiaustausch verwendet werden kann.

In einem P2P Netz sind die Teilnehmenden Geräte meist sehr unterschiedlich was Rechenleistung, Bandbreite und andere Spezifikationen angeht. Vorteile sind vor allem hohe Autonomie der Teilnehmenden sowie – bei guter Implementation der Protokolle – eine hohe Verfügbarkeit und Eigenschaften wie Zensurresistenz.