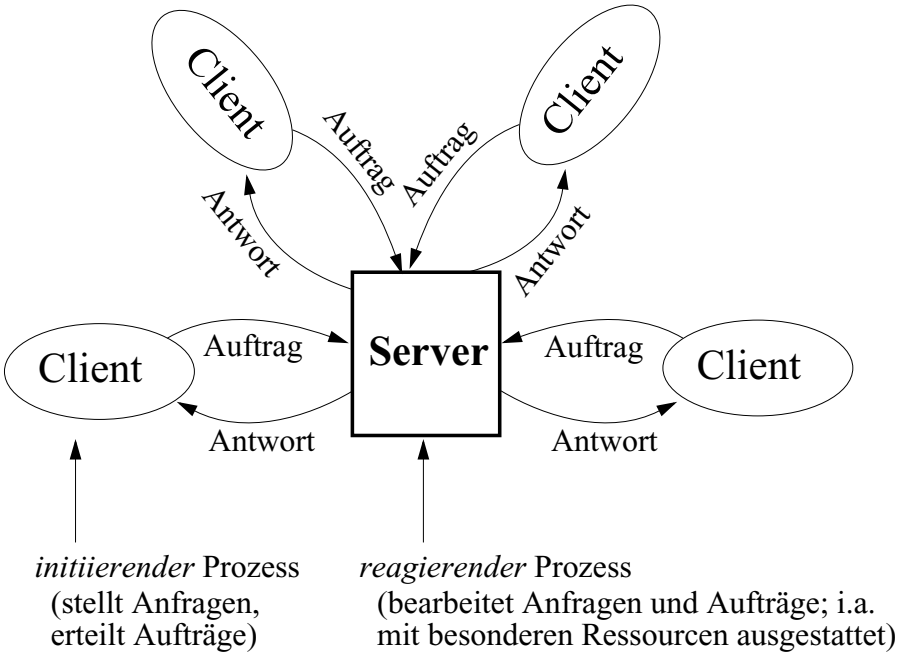
Client/Server-Modell

Das Client/Server-Modell



- Aufgabenteilung und asymmetrische Struktur
  - *Clients*: typischerweise Anwendungsprogramme und graphische Benutzungsschnittstelle (“front end”)
  - *Server*: zuständig für Dienstleistungen
- Typisches Kommunikationsparadigma: RPC

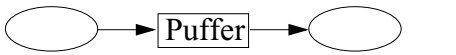
Eignung des Client/Server-Paradigmas

- Populär wegen des eingängigen Modells
  - entspricht Geschäftsvorgängen in unserer Dienstleistungsgesellschaft
  - gewohntes Muster → intuitive Struktur, gute Überschaubarkeit
- Effizienz durch spezialisierte „Dienstleister“
  - grosszügige Ausstattung (CPU-Leistung, Speicherkapazität usw.)
  - bestückt mit spezieller Software (Datenbank etc.)
- Kosteneffektivität durch bessere Auslastung wertvoller Ressourcen (z.B. bei “Compute Server”)
  - Clients brauchen oft kurzfristig Spitzenleistung
  - einzelner Client kann Ressourcen aber nicht dauerhaft auslasten
- Passend für viele Kooperationsbeziehungen, z.B.
  - Client erbittet Auskunft von einem spezialisierten Service
  - gefährdete Clients geben wertvolle Daten in Obhut des (gegen Missbrauch, Verlust, Diebstahl usw.) hoch gesicherten Servers

---

- Modell ist für viele Zwecke geeignet, jedoch nicht für alle (z.B. Pipelines, “peer-to-peer”, asyn. Mitteilung)!

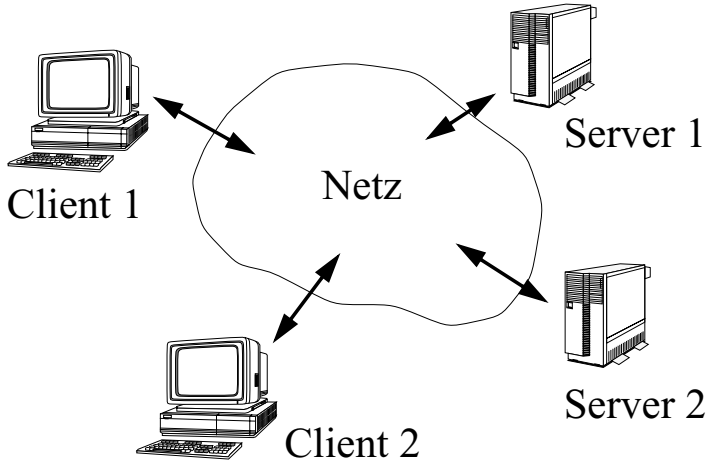
Beispiel Puffer als Pipeline:



- Puffer ist weder Client noch Server, sondern hat beide Rollen! (passiv gegenüber Produzent; aktiv gegenüber Konsument)
- Inversion der Kommunikationsbeziehung bei einem Puffer-Server!

Client- und Server-Maschinen

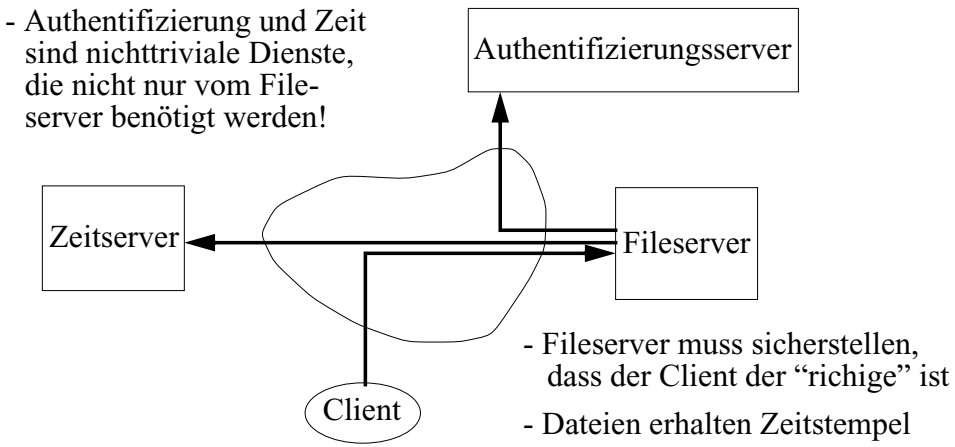
- Übertragung des Client/Server-Modells auf *Rechner*



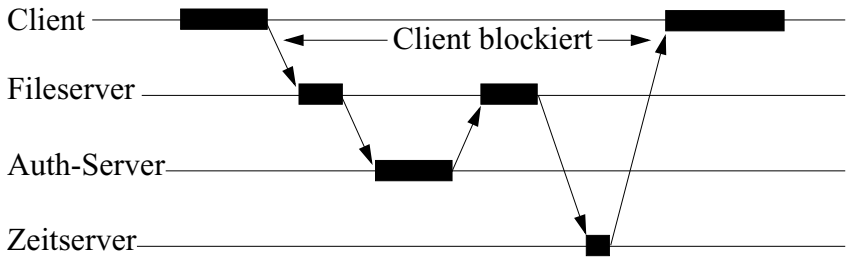
- Typischerweise PCs als Clients
  - u.a. mit graphischem Benutzungsinterface
- Andere, leistungsfähigere Rechner als Server
  - “zentrale” Dienste (z.B. Speicherserver)
  - gemeinsam benutzte Betriebsmittel
- Im allgemeinen müssen sich aber Server- und Client-*Prozesse* nicht auf dedizierten Rechnern befinden!

Client/Server-Rollen

- Server müssen ggf. zur Durchführung eines Dienstes die Dienstleistungen anderer Server in Anspruch nehmen



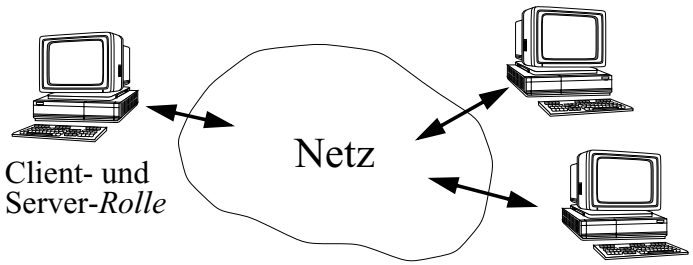
- Fileserver hat prinzipiell die *Rolle* eines Servers, zwischenzeitlich jedoch die *Rolle* eines Clients



Peer-to-Peer-Strukturen

"Gleichrangiger"

- Im Gegensatz zum asymmetrischen Client/Server-Modell



- Jeder Client fungiert zugleich als Server für seine Partner
  - → keine (teuren) dedizierten Server notwendig
  - → oft als Billiglösung von "echtem" Client/Server-Computing angesehen
- In der Reinform keine zentralisierten Elemente
  - dies wird gelegentlich in "politischer" Weise artikuliert (vgl. Tauschbörsen)

- *Nachteile*:
  - "Anarchischer" als *maschinenbezogene* Client/Server-Architektur
  - Rechner müssen leistungsfähig genug sein (cpu-Leistung, Speicher-ausbau), um für den "Besitzer" leistungstransparent zu sein
  - geringere Stabilität (Besitzer kann seine Maschine ausschalten...)
  - Datensicherung muss ggf. dezentral durchgeführt werden
  - Sicherheit und Schutz kritisch: Lizenzen, Viren, Integrität...

Zu vielen Aspekten von Peer-to-Peer-Systemen: R. Steinmetz, K. Wehrle (Eds): *Peer-to-Peer Systems and Applications*, Springer-Verlag, 2005

Zustandsändernde /-invariante Dienste

(Vgl. frühere Diskussion bzgl. RPC-Fehlersemantik!)

- Verändern Aufträge den Zustand des Servers wesentlich?

- Typische *zustandsinvariante* Dienste:
  - Auskunftsdienste (z.B. Name-Service)
  - Zeitservice

- Typische *zustandsändernde* Dienste:
  - Datei-Server

Idempotente Dienste / Aufträge

- Wiederholung eines Auftrags liefert gleiches Ergebnis

nicht zustandsinvariant!

- Beispiel: “Schreibe in Position 317 von Datei XYZ den Wert W”
- Gegenbeispiel: “Schreibe ans Ende der Datei XYZ den Wert W”
- Gegenbeispiel: “Wie spät ist es?” ← aber zustandsinvariant!

Wiederholbarkeit von Aufträgen

- Bei Idempotenz oder Zustandsinvarianz kann bei Verlust des Auftrags (timeout beim Client) dieser erneut abgesetzt werden (→ einfache Fehlertoleranz)

Zustandslose / -behaftete Server

stateless — statefull

- Hält der Server Zustandsinformation über Aufträge hinweg? (“session”)
  - z.B. (Protokoll)zustand des Clients
  - z.B. Information über frühere damit zusammenhängende (Teil)aufträge

- Aufträge an zustandslose Server müssen autonom sein

- *Beispiel*: Datei-Server

```
open(“XYZ”);
read;
read;
close;
```

In klassischen Systemen hält sich das Betriebssystem Zustandsinformation, z.B. über die Position des Dateizeigers geöffneter Dateien

- bei zustandslosen Servern entfällt open/close; jeder Auftrag muss vollständig beschrieben sein (Position des Dateizeigers etc.)

notw. Zustandsinformation ist beim Client

- zustandsbehaftete Server daher i.a. effizienter
- Dateisperren sind bei echten zustandslosen Servern nicht (einfach) möglich
- zustandsbehaftete Server können wiederholte Aufträge erkennen (z.B. durch Speichern von Sequenznummern) → Idempotenz

- *Crash* eines Servers: Weniger Probleme im zustandslosen Fall (→ Fehlertoleranz)! ← entscheidender Vorteil!

- NFS (Network File System von Sun): zustandslos
- RFS (Remote File System von UNIX System V): zustandsbehaftet

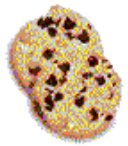
Sind Webserver zustandslos?

- Beim HTTP-Zugriffsprotokoll wird über den Auftrag hinweg keine Zustandsinformation gehalten
  - jeder link, den man anklickt, löst eine neue “Transaktion” aus
- Stellt ein Problem beim E-Commerce dar
  - gewünscht sind Transaktionen über mehrere Klicks hinweg und
  - Wiedererkennen von Kunden (beim nächsten Klick oder Tage später)
  - erforderlich z.B. für Realisierung von “Warenkörben” von Kunden
  - gewünscht vom Marketing (Verhaltensanalyse von Kunden)

Lösungsmöglichkeiten (Korrelation von Web-Seiten zur Realisierung von “Warenkörben” im WWW):

- IP-Adresse des Kunden an Auftrag anheften?
  - Problem: Proxy-Server → viele Kunden haben gleiche IP-Adresse
  - Problem: dynamische IP-Adressen → keine Langzeitwiedererkennung
- “URL rewriting” und dynamische Web-Seiten
  - Einstiegsseite eine eindeutige Nummer anheften, wenn der Kunde diese erstmalig aufruft
  - diese Nummer jedem link der Seite anheften und mit zurückübertragen
- Cookies
  - kleine Textdatei, die ein Server einem Browser (= Client) schickt und die im Browser gespeichert wird
  - nur der Sender des Cookies darf dieses später wieder lesen

Cookies



Auszug aus
http://home.netscape.com/newsref/std/cookie_spec.html:

Cookies are a general mechanism which server side connections (such as CGI scripts) can use to both store and retrieve information on the client side of the connection. The addition of a simple, persistent, client-side state significantly extends the capabilities of Web-based client/server applications.

A server, when returning an HTTP object to a client, may also send a piece of state information which the client will store... Any future HTTP requests made by the client... will include a transmittal of the current value of the state object from the client back to the server. The state object is called a cookie, for no compelling reason.

This simple mechanism provides a powerful new tool which enables a host of new types of applications to be written for web-based environments. Shopping applications can now store information about the currently selected items, for fee services can send back registration information and free the client from retyping a user-id on next connection, sites can store per-user preferences on the client, and have the client supply those preferences every time that site is connected to.

A cookie is introduced to the client by including a Set-Cookie header as part of an HTTP response... The expires attribute specifies a date string that defines the valid life time of that cookie. Once the expiration date has been reached, the cookie will no longer be stored or given out... A client may also delete a cookie before it’s expiration date arrives if the number of cookies exceeds its internal limits.

- Denkübung: Müssen Proxy-Server geeignete Massnahmen vorsehen?
- Übung: Man finde heraus, was doubleclick.net macht (und wie)

Cookies (2)

- Anwendung von cookies war und ist umstritten (Ausspionieren des Verhaltens); dazu kam eine gewisse Paranoia:

  http://www.cookiecentral.com/creport.htm

  http://www.ciac.org/ciac/bulletins/i-034.shtml

  The Energy Department's Computer Incident Advisory Capability (CIAC) recently issued a report on cookie technology and its use on the web....

  The report stressed that there's a sense of paranoia involved with cookies, cookies cannot harm your computer or pass on private information such as an email address without the user's intervention in the first place. Paranoia has recently been sparked by one rumour involving AOL's new software, it claimed that AOL were planning to use cookies to obtain private information from users hard drives.

- Problemlos ist das allerding nicht:

  http://www.cookiecentral.com/cookie5.htm

  Unfortunately, the original intent of the cookie has been subverted by some unscrupulous entities who have found a way to use this process to actually track your movements across the Web. They do this by surreptitiously planting their cookies and then retrieving them in such a way that allows them to build detailed profiles of your interests, spending habits, and lifestyle... it is rather scary to contemplate how such an intimate knowledge of our personal preferences and private activities might eventually be used to brand each of us as members of a particular group.