# *Verteilte Systeme*

Vorlesung 2
Dr. Sebastian Iwanowski
FH Wedel

# Inhaltlicher Umfang dieser Vorlesung

## Inhaltliche Voraussetzungen:

Programmieren mit Java

Objektorientierte Programmierung im Allgemeinen

Vorlesung Rechnernetze

## Lernziele dieser Vorlesung:

Verständnis für **verteilte** Rechnerarchitekturen, Daten und Algorithmen (als Alternative zur zentralistischen Denkweise)

Nebenläufigkeitskonzepte (Transaktionskonzepte)

Grundlagen für Verteilten Systeme (soweit nicht schon in Vorlesung Rechnernetze gegeben): vor allem Synchronisation, Fehlertoleranz

Konkrete Programmiertechniken in Java

Kennenlernen geeigneter Beispiele aus der Praxis

# Literatur

**Allgemeine Lehrbücher:**

George Coulouris / Jean Dollimore / Tim Kindberg:
*Distributed Systems, Concepts and Design*,
Addison-Wesley 2001, ISBN 0201-61918-0
Deutsche Übersetzung auch erhältlich:
Pearson Studium 2002, ISBN 3-8273-7022-1

Andrew Tanenbaum / Marten van Steen:
*Verteilte Systeme, Grundlagen und Paradigmen,*
Pearson Studium 2003, ISBN 3-8273-7057-4

Ulrike Hammerschall:
*Verteilte Systeme und Anwendungen, Architekturkonzepte, Standards und Middleware-Technologien*,
Pearson Studium 2005, ISBN 3-8273-7096-5

# Literatur

**Java-Bücher für verteilte Systeme:**


Marko Boger:
*Java in verteilten Systemen, Nebenläufigkeit, Verteilung, Persistenz*,
dpunkt-Verlag 1999, ISBN 3-932588-32-0

David Chappell / Tyler Jewell:
*Java Web Services*,
O'Really 2002, ISBN 0-596-00269-6

Manfred Hein / Henner Zeller:
*Java Web Services, Entwicklung plattformübergreifender Dienste mit J2EE, XML und SOAP*,
Addison-Wesley 2003, ISBN 3-8273-2071-2
Neuauflage erscheint 27.04.2005, ISBN 3-8273-2231-6

Thomas Stark: *J2EE, Einstieg für Anspruchsvolle*, Pearson Studium 2005, ISBN 3-8273-2184-0

# Verteilte Systeme

1. Innovative Beispiele aus der Praxis
2. Allgemeine Anforderungen und Techniken verteilter Systeme
3. Die Client-Server-Beziehung und daraus entstehende Fragestellungen
4. Nebenläufigkeitstechniken in Java
5. Entfernte Aufrufe
6. Objektmigration
7. Agententechnologie
8. Dienstevermittlung
9. Synchronisation von Daten
10. Konzepte zur Erzielung von Fehlertoleranz
11. Web Services

# Definition Verteiltes System

**Andrew Tanenbaum / Maarten van Steen:**

> *Ein verteiltes System ist eine Menge unabhängiger Computer, die dem Benutzer wie ein einzelnes System erscheint.*

**Günther Bengel (FH Mannheim):**

> *Ein verteiltes System ist ein System, in dem eine Reihe einzelner Funktionseinheiten, die miteinander über ein Transportsystem verbunden sind, in Zusammenarbeit Anwendungen bewältigen.*

**Erweiterung (lw):**

> *Die Funktionseinheiten verarbeiten softwaretechnisch Daten und die korrekte Funktionalität des Transportsystems ist nicht gewährleistet.*

# Allgemeine Anforderungen an Verteilte Systeme

**Benutzerspezifische Anbindung an das System**

**Offenheit**

**Transparenz**

**Skalierbarkeit**

# Transparenzforderungen nach ISO (1995)

| **Transparenztyp** | **Beschreibung** |
|---|---|
| Zugriff (access) | Verberge, wie auf einzelne Ressource zugegriffen werden muss |
| Ort (location) | Verberge, wo einzelne Ressourcen liegen (von wo gefragt wird) |
| Persistenz | Verberge, ob sich Ressource im Hauptspeicher oder auf der Festplatte befindet |
| Migration | Verberge, dass Ressourcen verschoben werden können |
| Relokation | Verberge, dass Ressourcen verschoben werden können, *während sie benutzt werden* |
| Replikation | Verberge, wievielfach eine Ressource vorhanden ist |
| Konkurrenz | Verberge, wie viele Nutzer gleichzeitig auf die Ressource zugreifen |
| Ausfall (failure) | Verberge, welche Ressourcen nicht zur Verfügung stehen |

# Weitere Transparenzforderungen

| Transparenztyp | Beschreibung |
|---|---|
| Parallelität | Verberge, wie viele Prozesse gleichzeitig laufen |
| Leistung | Verberge, wie groß die Kapazitäten der einzelnen Rechner sind |
| Skalierung | Verberge, wie viele Teilnehmer und Funktionen das gesamte System verkraftet |

**Grundsatz:**

*Es ist nicht immer sinnvoll, dass alle Transparenzforderungen erfüllt sind.*

*Es sollte aber immer klar sein, welche Transparenzforderungen erfüllt sind und welche nicht.*

# Skalierung

**Kriterien (nach Neumann):**

1. *Physische Kapazität des Gesamtsystems*
2. *Geographische Ausdehnung*
3. *Anzahl der unabhängigen Systemteile*

# Skalierung

## Probleme mit zentralen Konzepten:

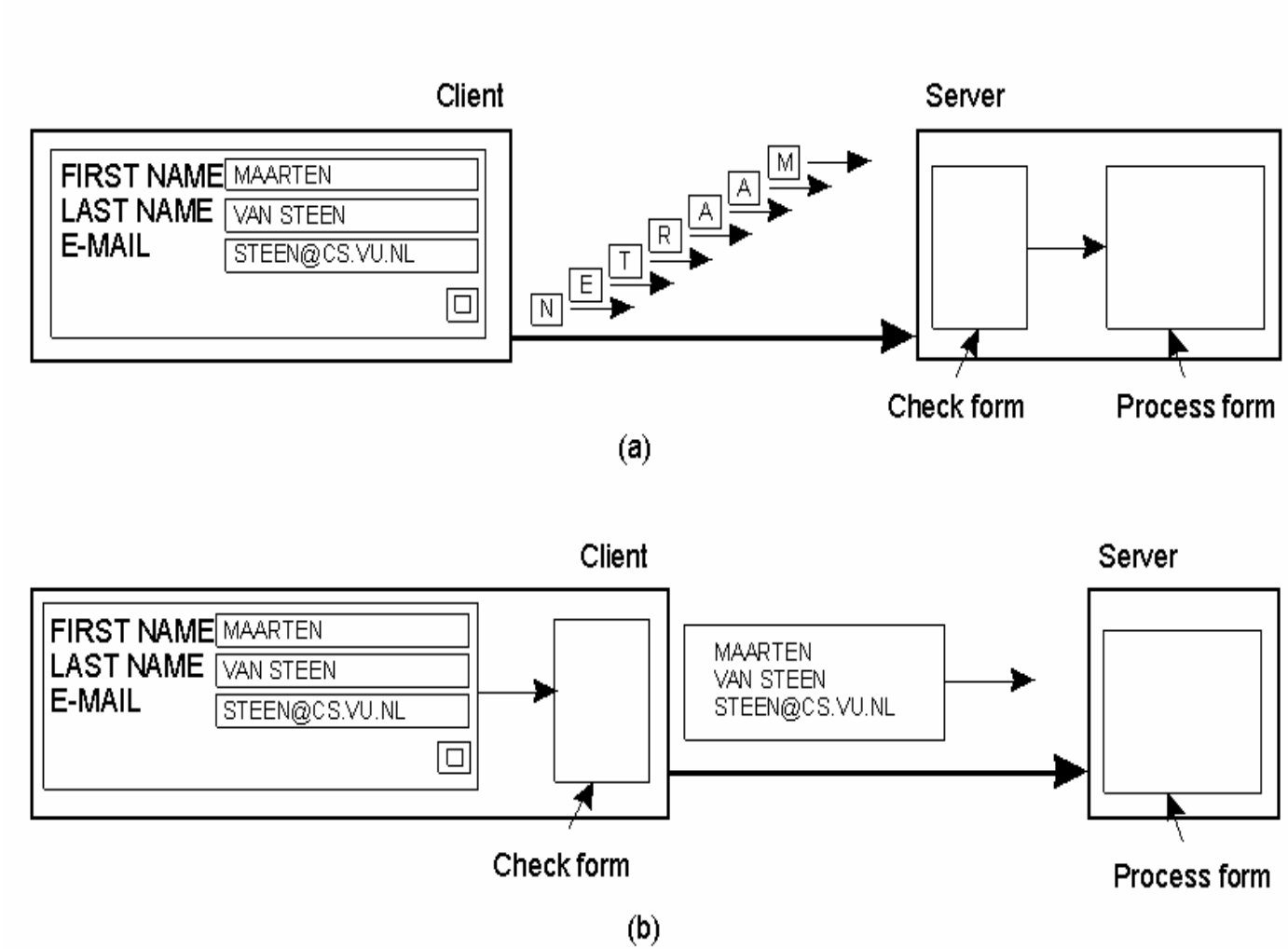
| Zentrales Konzept | Beispiel |
|---|---|
| Zentraler Anbieter | Ein einzelner Server für alle |
| Zentrale Datenhaltung | Ein einzelnes on-line-Telephonbuch |
| Zentraler Algorithmus | Routingalgorithmus auf vollständigen Systemdaten |

# Skalierung

## Lösungen mit verteilten Ansätzen:

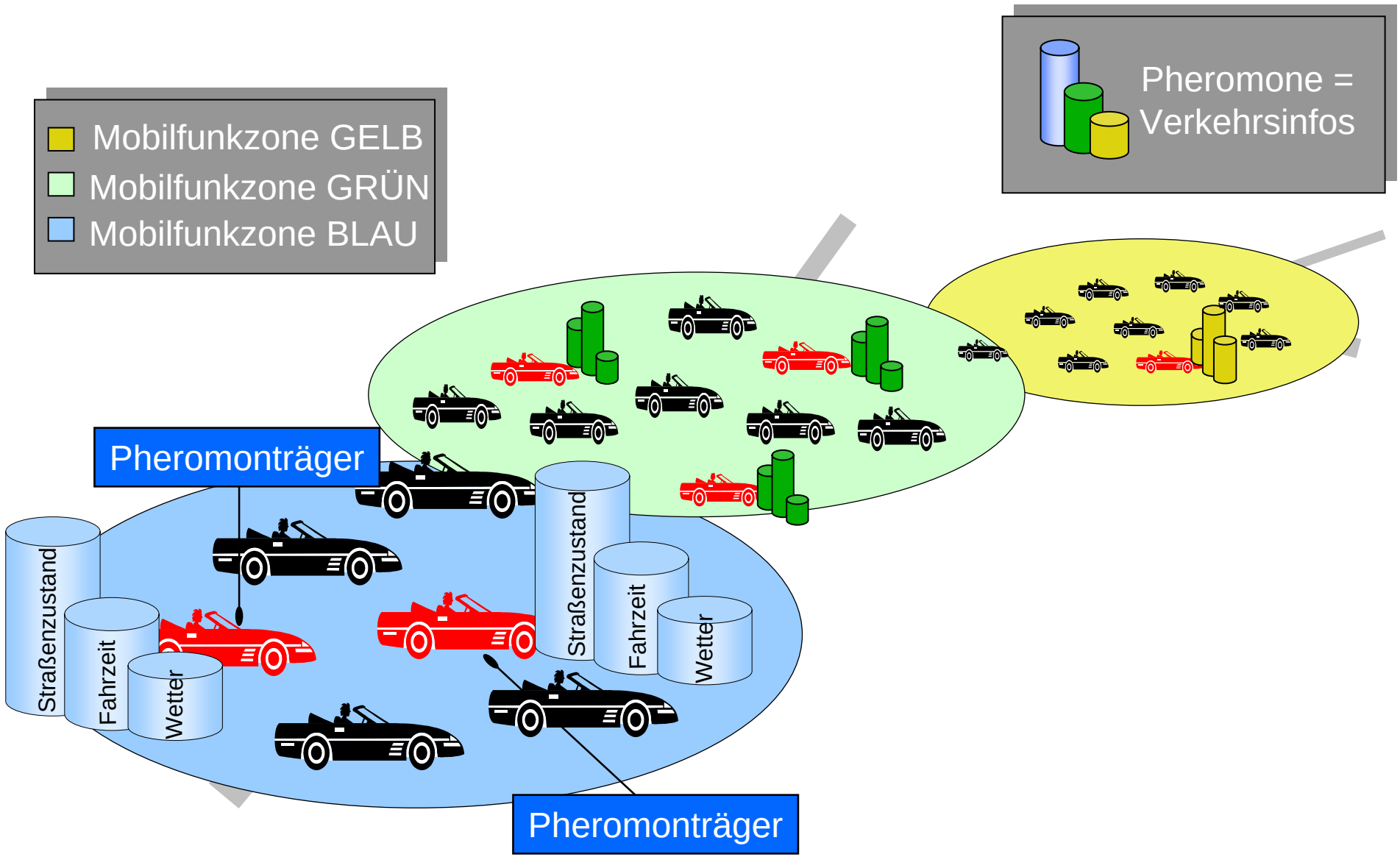
| **Ansatz** | **Beispiel** |
|---|---|
| Halte Software so lokal wie möglich | Formularausfüllung |
| Hierarchisch aufgebaute Namensverwaltung | Internet-DNS (Domain Naming System) |
| Verteilte Algorithmen | Car Swarm Intelligence |

# Beispiel: Formularausfüllung



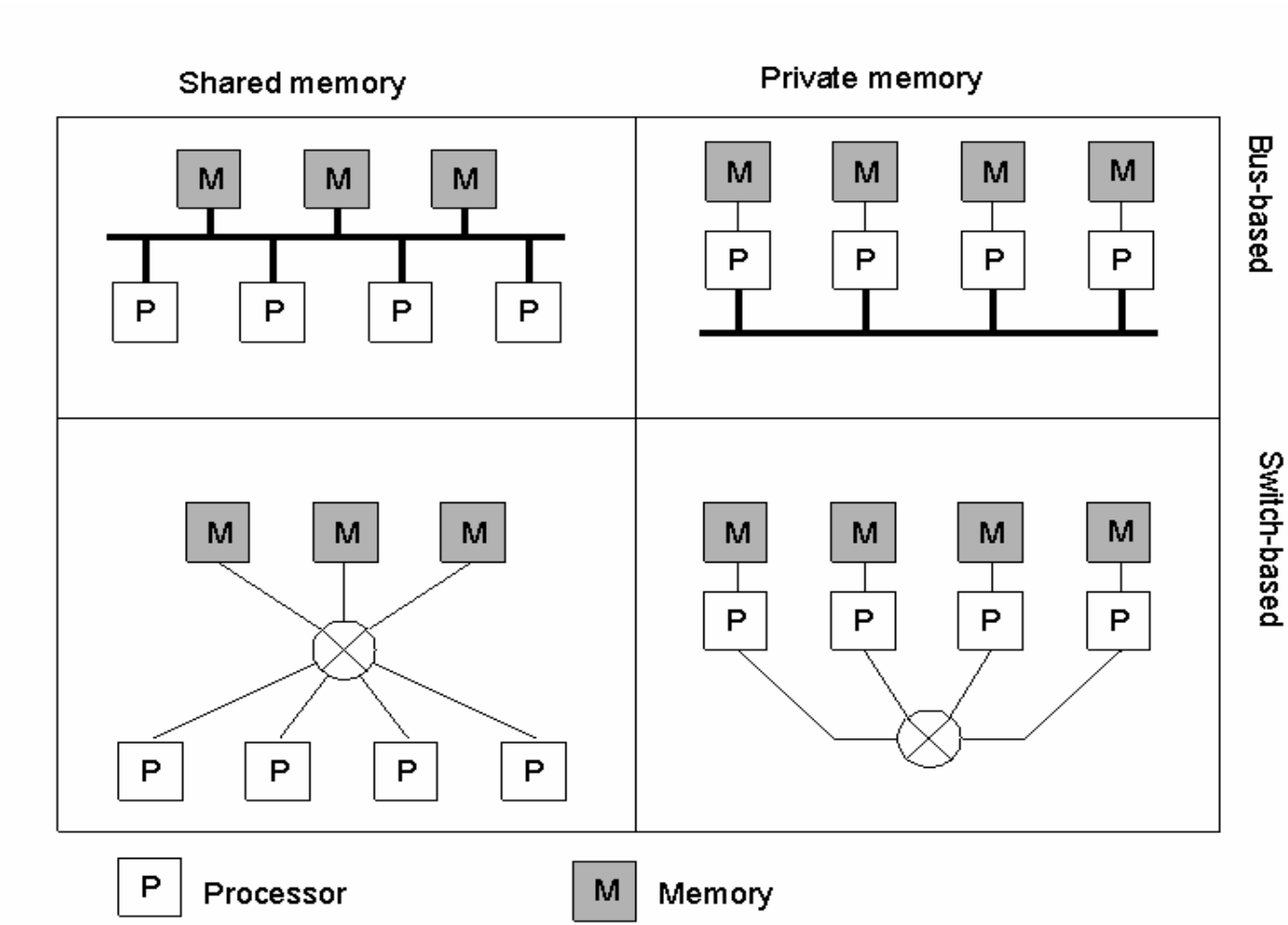

aus Tanenbaum / van Steen: S. 31

# Beispiel: Car Swarm Intelligence



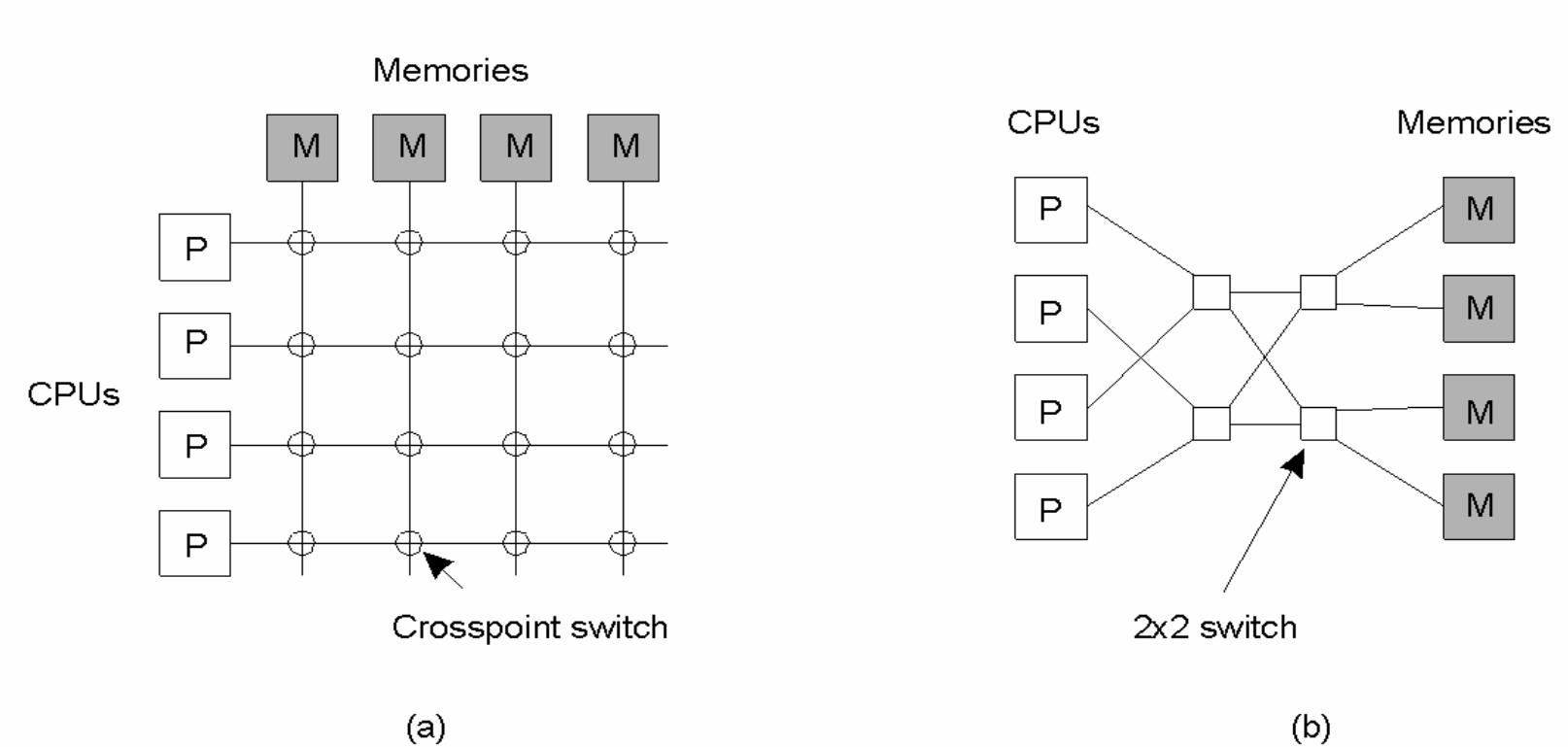
# *Exkurs: Konzepte verteilter Hardware*

# Verschiedene Hardwareverbindungskonzepte




aus Tanenbaum / van Steen: S. 33

# Hardwareverbindungen mit Schaltern

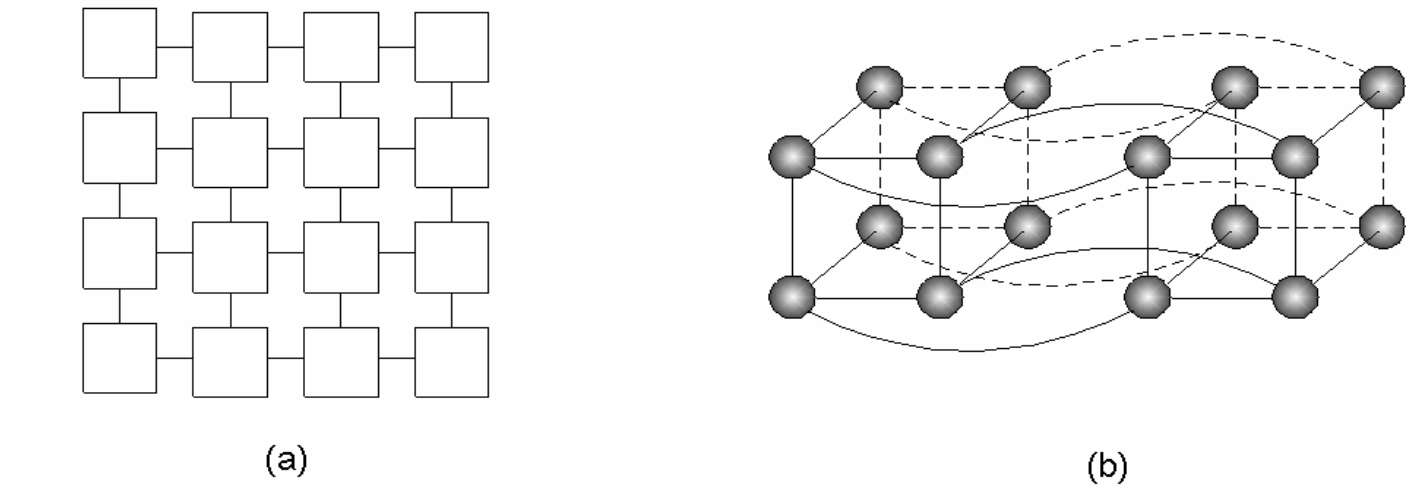



aus Tanenbaum / van Steen: S. 36

**Komplexitätsreduktion durch intelligentes Design**

# Multicomputersysteme

## Homogene Multicomputersysteme (angenehmer Spezialfall)




Gitter (grid)

Mehrdimensionaler Würfel (hypercube)

## Unterscheidung nach Aufgabenstellung:

*Systeme, die mit dem Zweck der besseren Lösbarkeit des Problems verteilt werden*

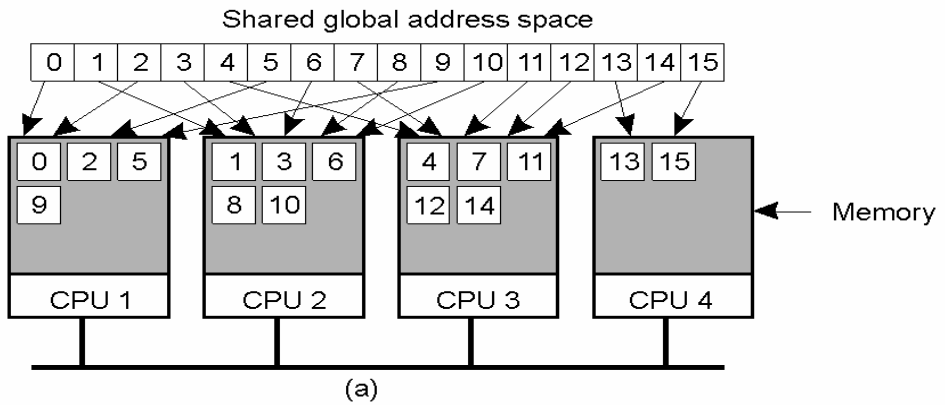
➔ häufig homogene Multicomputersysteme

*Systeme, deren Verteilung durch die Problemstellung gegeben ist* **(Normalfall)**

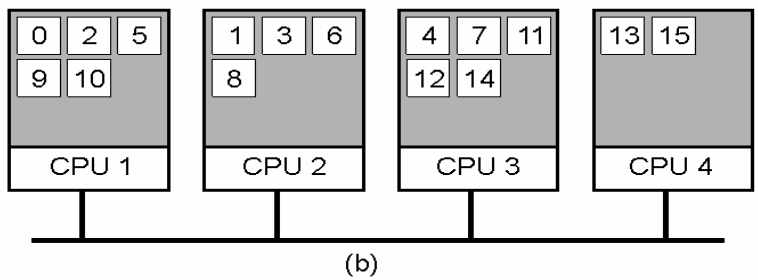
➔ immer heterogene Multicomputersysteme

# Mehrprozessorsysteme mit gemeinsamem Adressraum

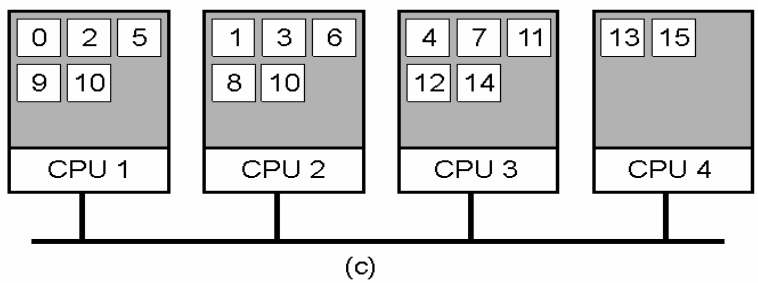
Ausgangssituation:
Daten sind in Seiten zusammengefasst
und auf die Prozessoren verteilt




Situation, nachdem CPU 1 schreibenden
Zugriff auf Seite 10 angefordert hat




Situation, nachdem CPU 1 lesenden
Zugriff auf Seite 10 angefordert hat



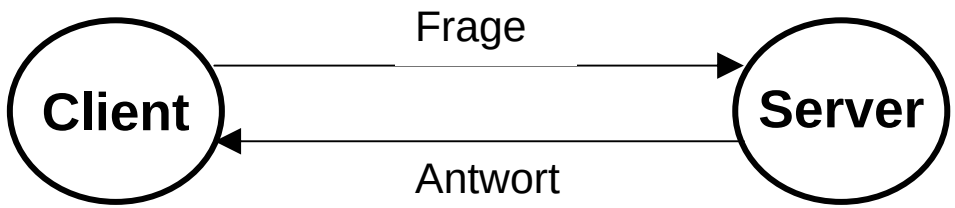

**Allgemeine Fragestellung:**

**Wer hat wann Zugriff auf die Daten ?**

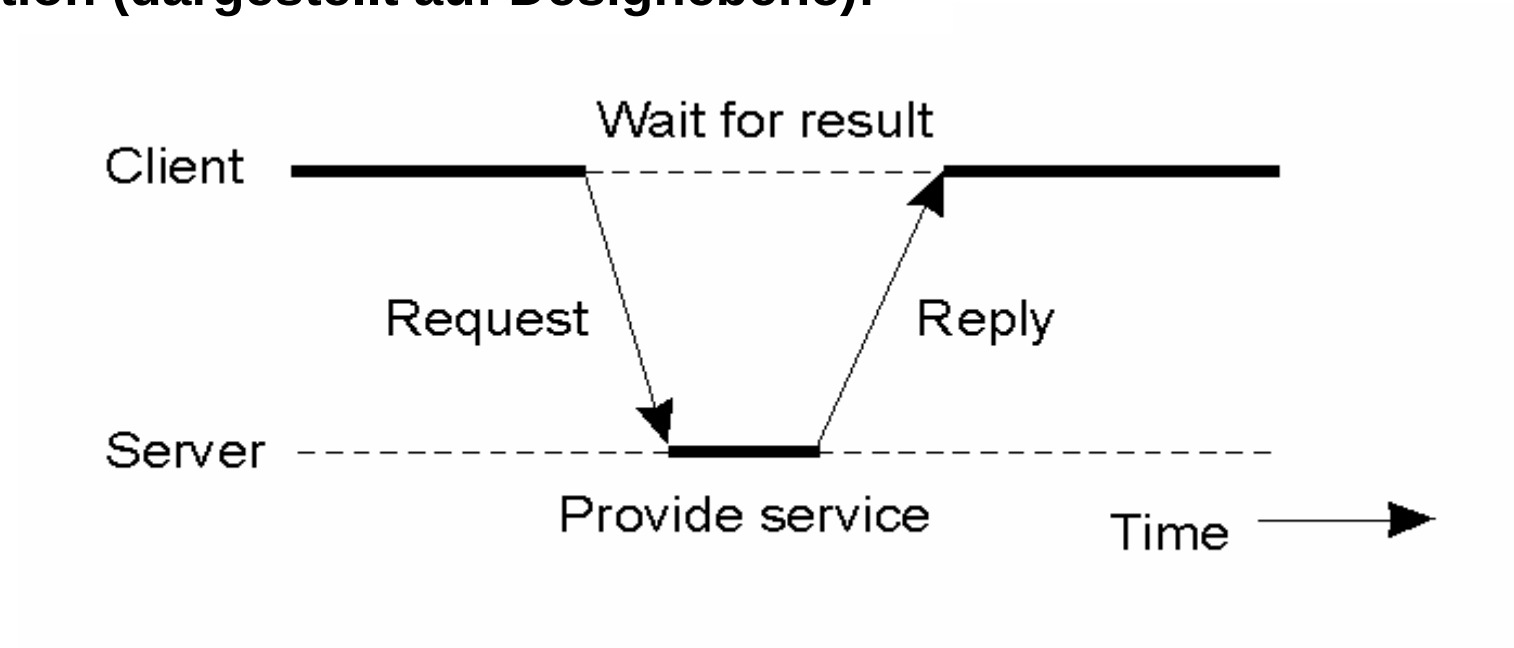
# Verteilte Systeme

1. Innovative Beispiele aus der Praxis
2. Allgemeine Anforderungen und Techniken verteilter Systeme
3. Die Client-Server-Beziehung und daraus entstehende Fragestellungen
4. Nebenläufigkeitstechniken in Java
5. Entfernte Aufrufe
6. Objektmigration
7. Agententechnologie
8. Dienstevermittlung
9. Synchronisation von Daten
10. Konzepte zur Erzielung von Fehlertoleranz
11. Web Services

# Client-Server-Architektur



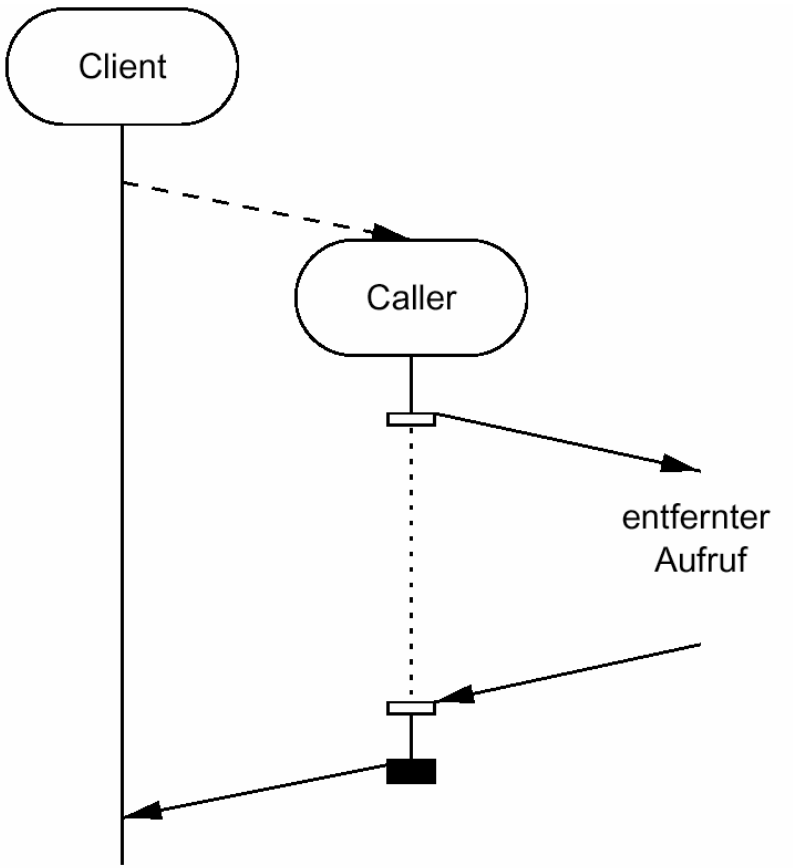

**Situation (dargestellt auf Designebene):**



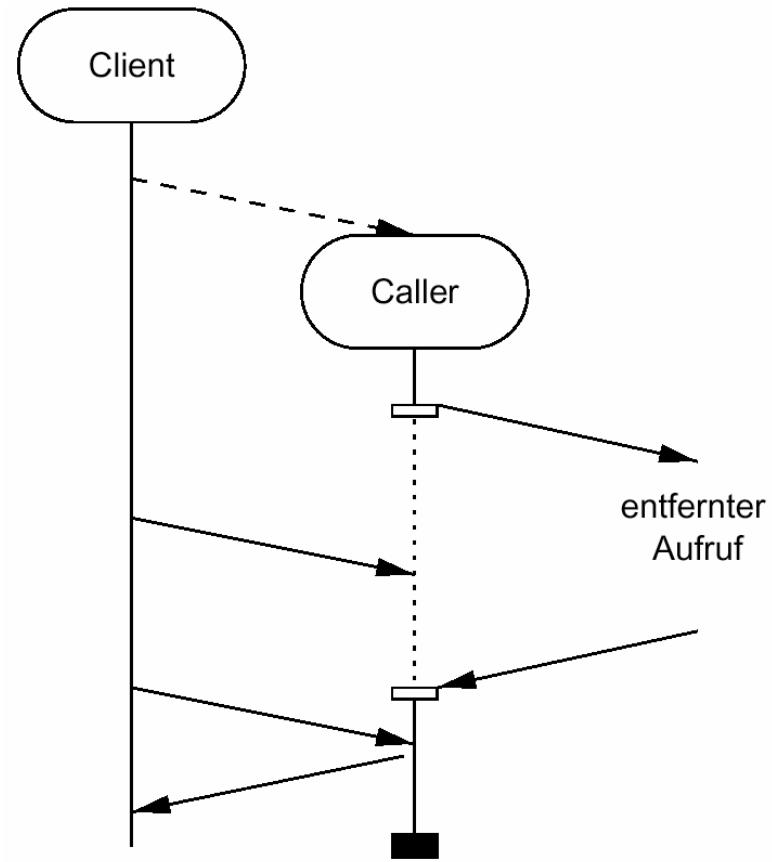

**Frage: Wie wird das auf Prozessebene realisiert ?**

# Client-Server-Architektur

**Callback:**




**Polling:**




**Problem: Die Kommunikationskanäle sind unzuverlässig !**

# Client-Server-Architektur

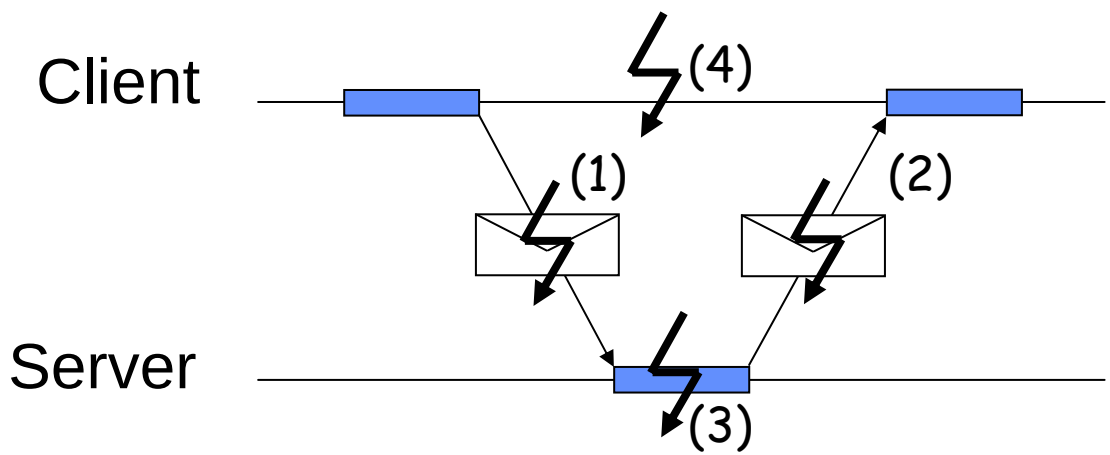
## Fehlermöglichkeiten:

Verlust der Auftragsnachricht (1)

Verlust der Ergebnisnachricht (2)

Ausfall des Servers (3)

Ausfall des Klienten (4)



# Beim nächsten Mal:
# Client-Server-Beziehung im Detail