Verteilte Systeme

Vorlesung 2 vom 15.04.2004
Dr. Sebastian Iwanowski
FH Wedel

Inhaltlicher Umfang dieser Vorlesung

Inhaltliche Voraussetzungen:

Programmieren, Grundkenntnisse Java (Syntax),

Vorteilhaft: Objektorientierte Programmierung

Lernziele dieser Vorlesung:

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

Nebenläufigkeitskonzepte (Transaktionskonzepte)

Grundlagen der Verteilten Systeme: Namensverwaltung, Synchronisation, Fehlertoleranz, Sicherheit

Konkrete Programmiertechniken ? (in Java)

Vorlesungsaufbau

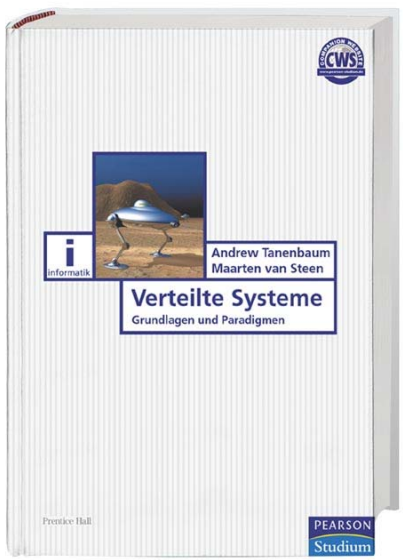
Vorlesungskonzept von Ralf Möller

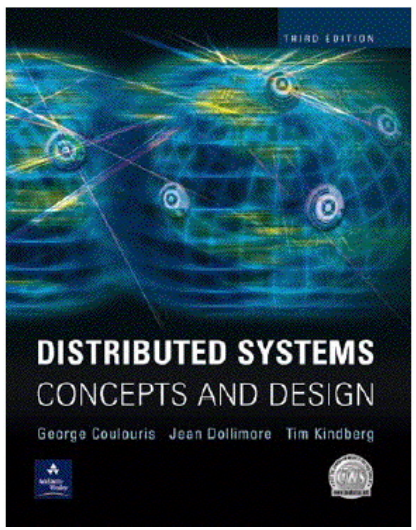
andere Reihenfolge, andere Schwerpunkte

Material zu dieser Vorlesung: www.fh-wedel.de/~iw/Lehrveranstaltungen/VS.html

Vorlesungsmaterial von Ralf Möller (www.fh-wedel.de/~mo/lectures/vsys-sose-03.html)

Literatur (vorerst):





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

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:

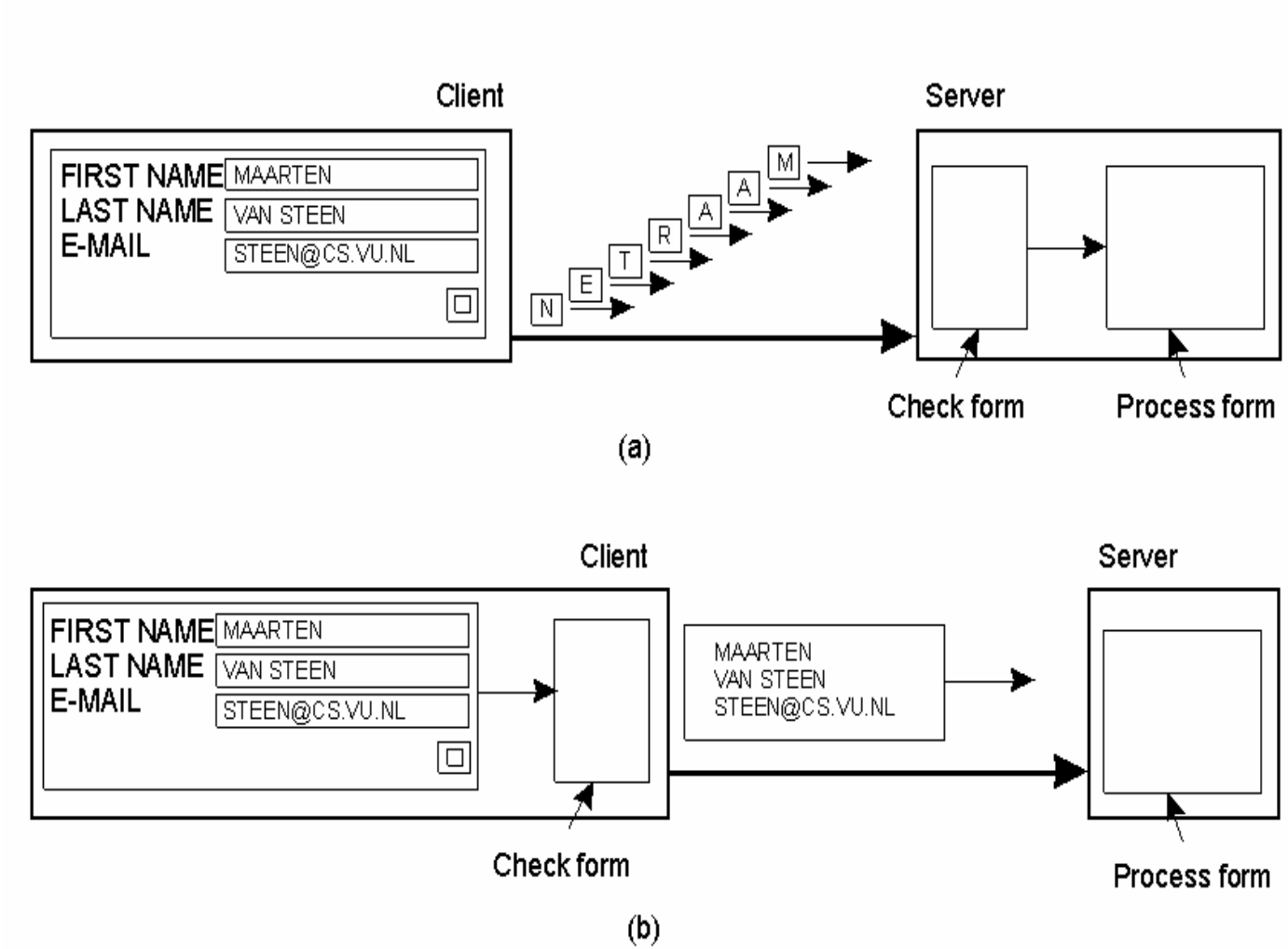
Zentrales Konzept	Beispiel
Zentraler Anbieter	Ein einzelner Server für alle
Zentrale Datenhaltung	Ein einzelnes on-line-Telephonbuch
Zentraler Algorithmus	Routingalgorithmus auf vollständigen Systemdaten

Skalierung

Lösungen mit verteilten Ansätzen:

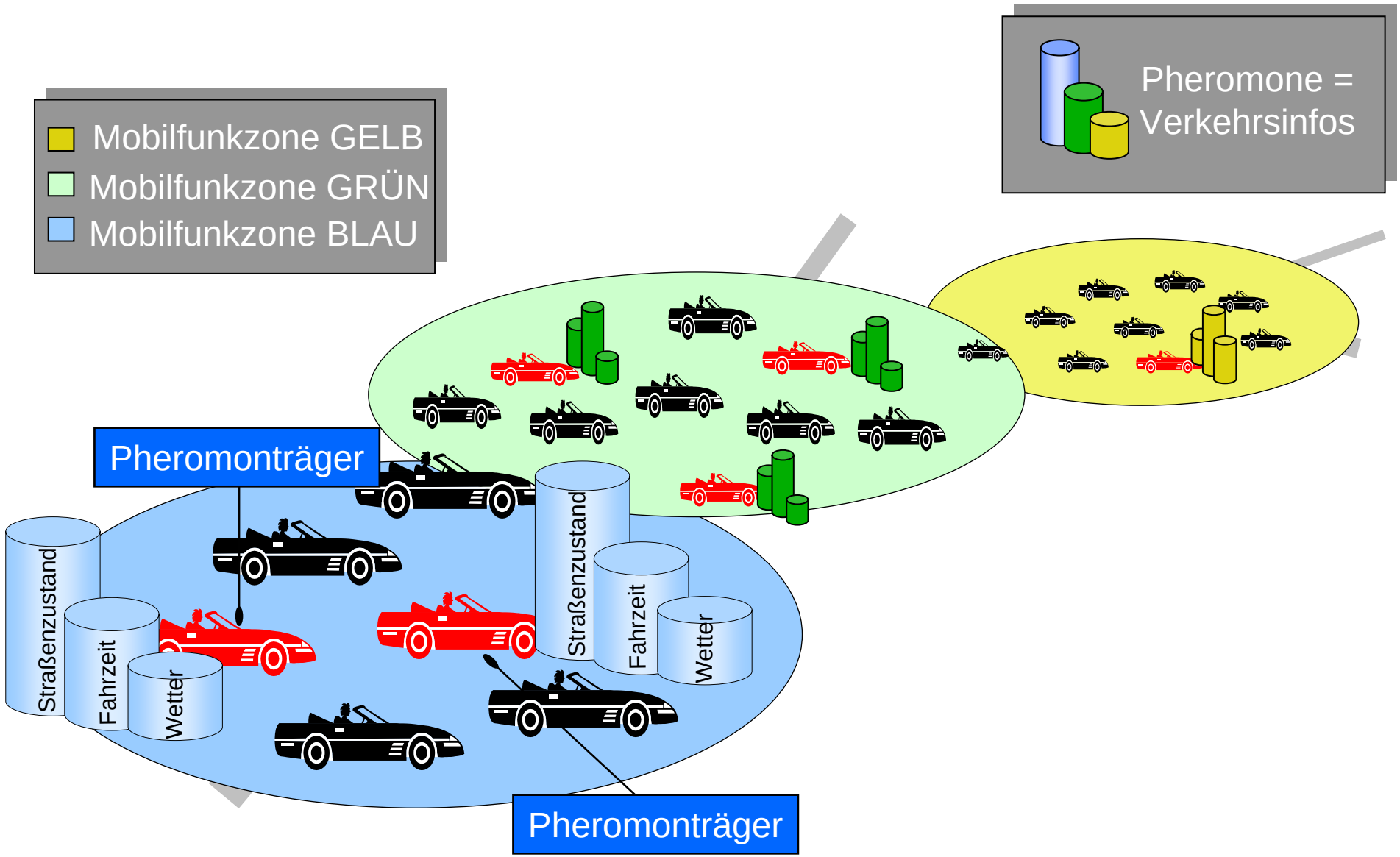
Ansatz	**Beispiel**
Halte Software so lokal wie möglich	Formularausfüllung
Hierarchisch aufgebaute Namensverwaltung	Internet-DNS (Domain Naming System)
Verteilte Algorithmen	Car Swarm Intelligence

Beispiel: Formularausfüllung



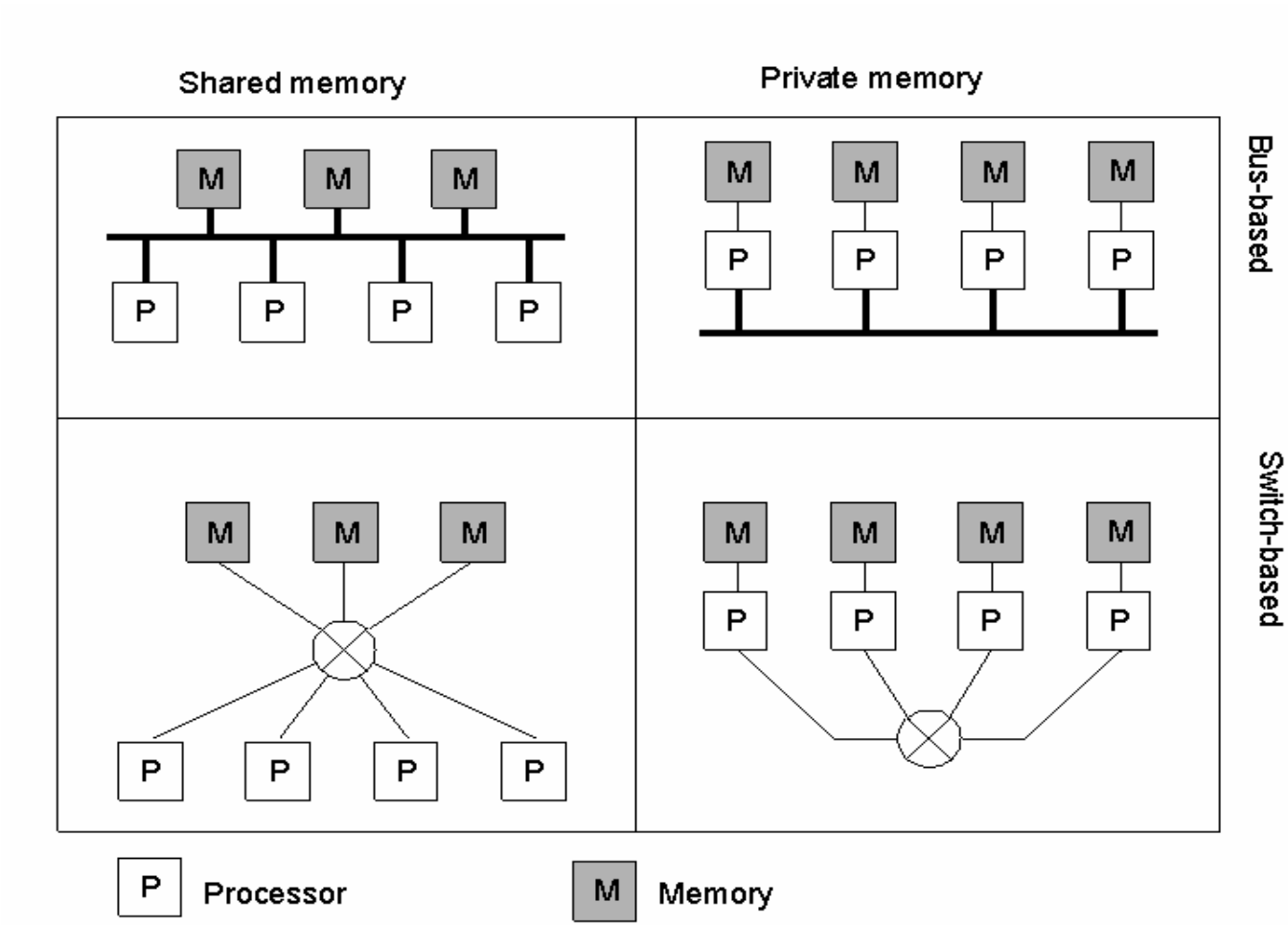
aus Tanenbaum / van Steen: S. 31

Beispiel: Car Swarm Intelligence



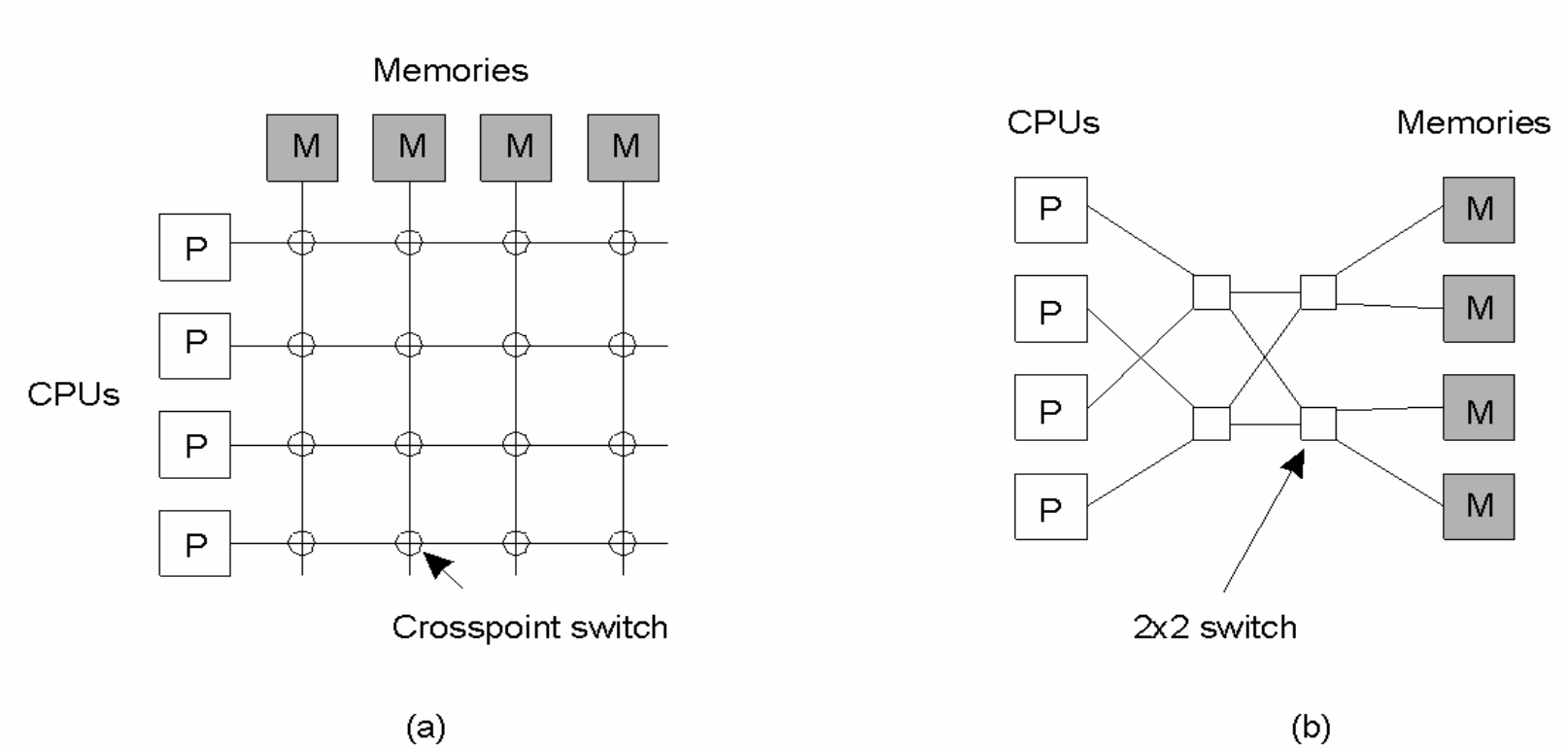
Konzepte verteilter Hardware

Verschiedene Hardwareverbindungskonzepte



aus Tanenbaum / van Steen: S. 33

Hardwareverbindungen mit Schaltern

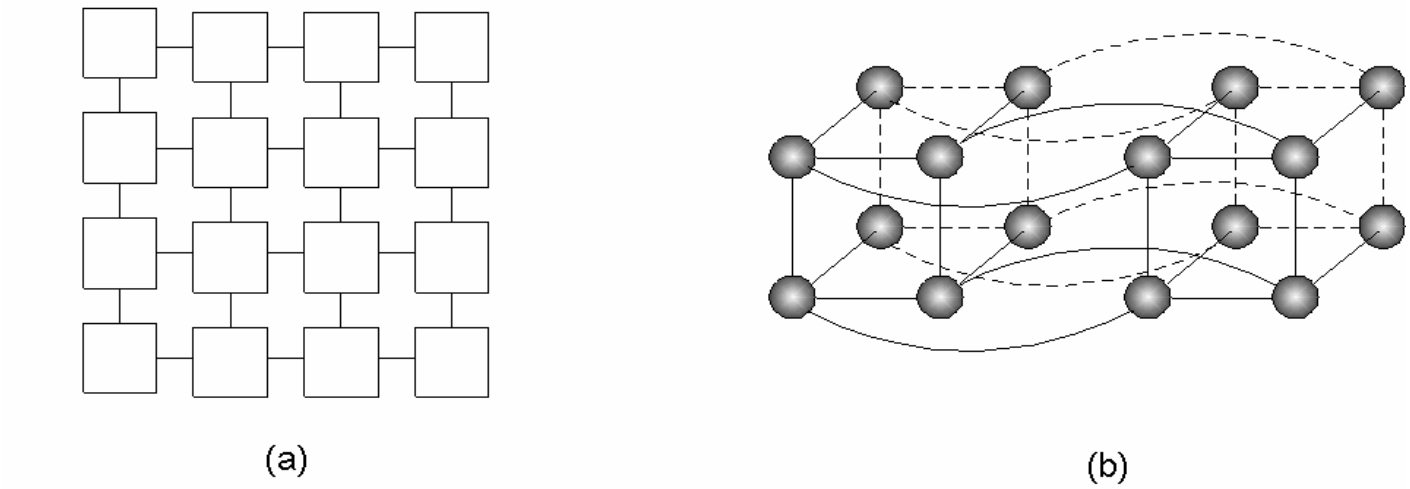


aus Tanenbaum / van Steen: S. 36

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

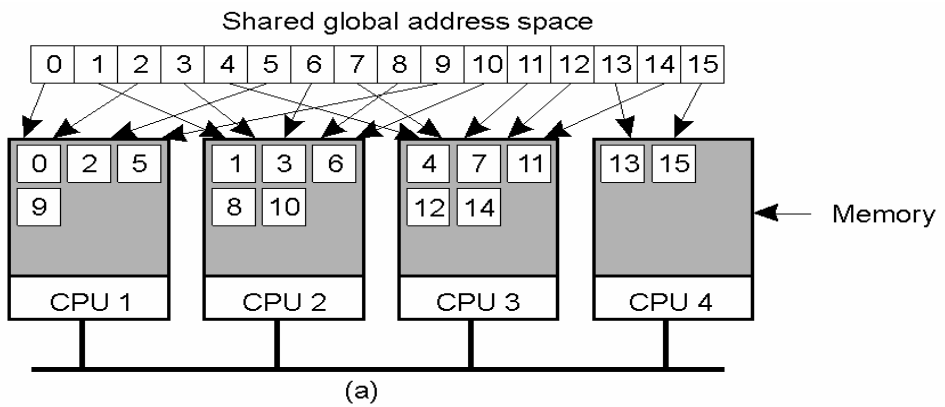
Systeme, deren Verteilung durch die Problemstellung gegeben ist **(Normalfall)**

➔ immer heterogene Multicomputersysteme

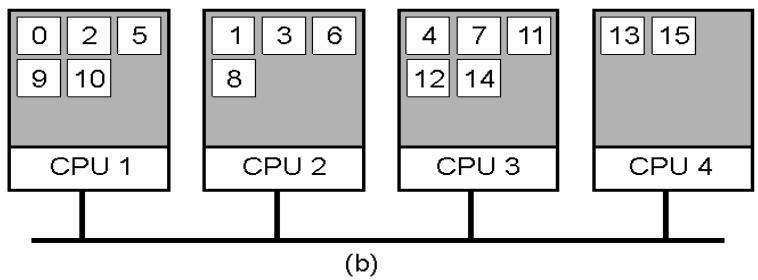
Grundlegende Konzepte verteilter Software

Verteilte Prozessoren mit gemeinsamem Adressraum

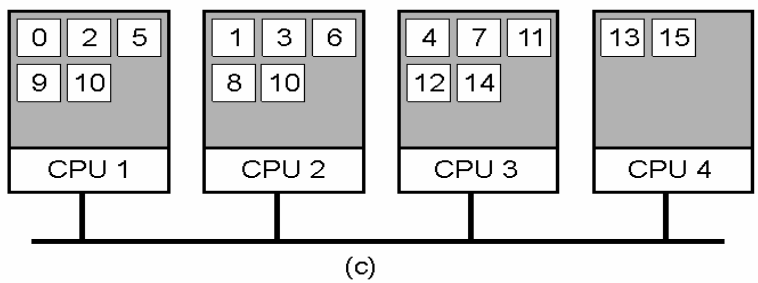
Ausgangssituation:
Daten sind in Seiten zusammengefasst und auf die Prozessoren verteilt



Situation, nachdem CPU schreibenden Zugriff auf Seite 10 angefordert hat



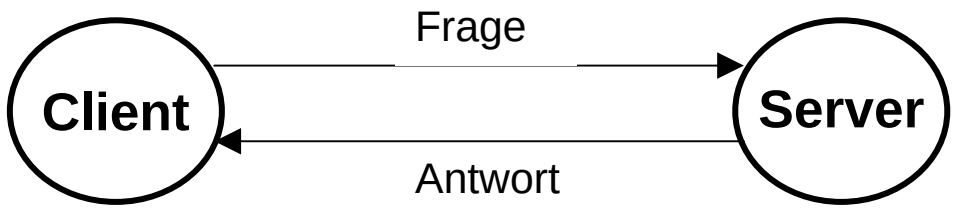
Situation, nachdem CPU lesenden Zugriff auf Seite 10 angefordert hat



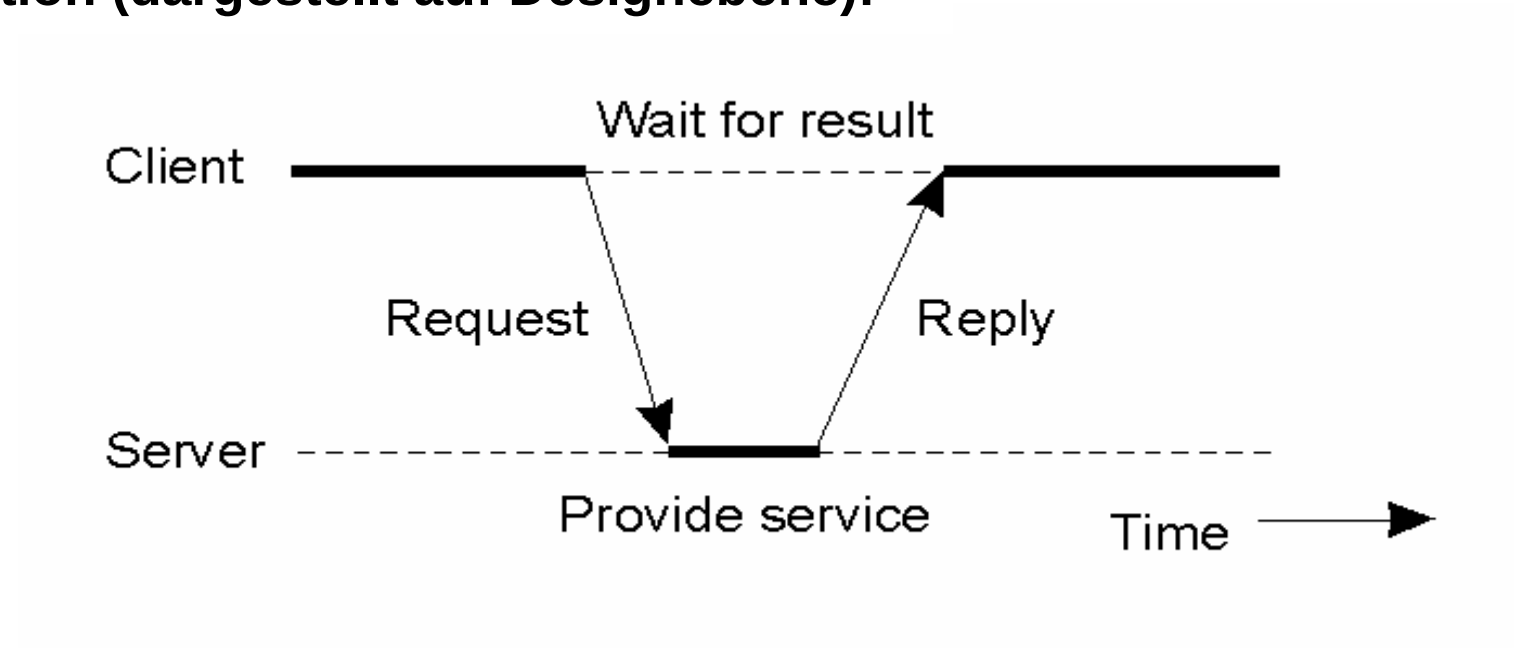
Allgemeine Fragestellung:

Wer hat wann Zugriff auf die Daten ?

Client-Server-Architektur



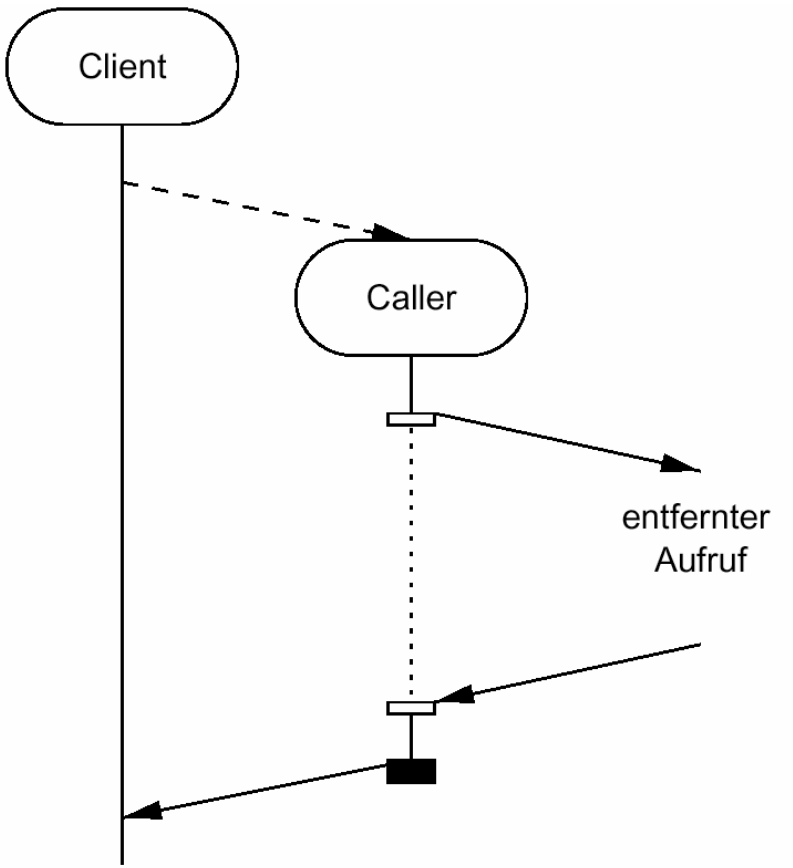
Situation (dargestellt auf Designebene):



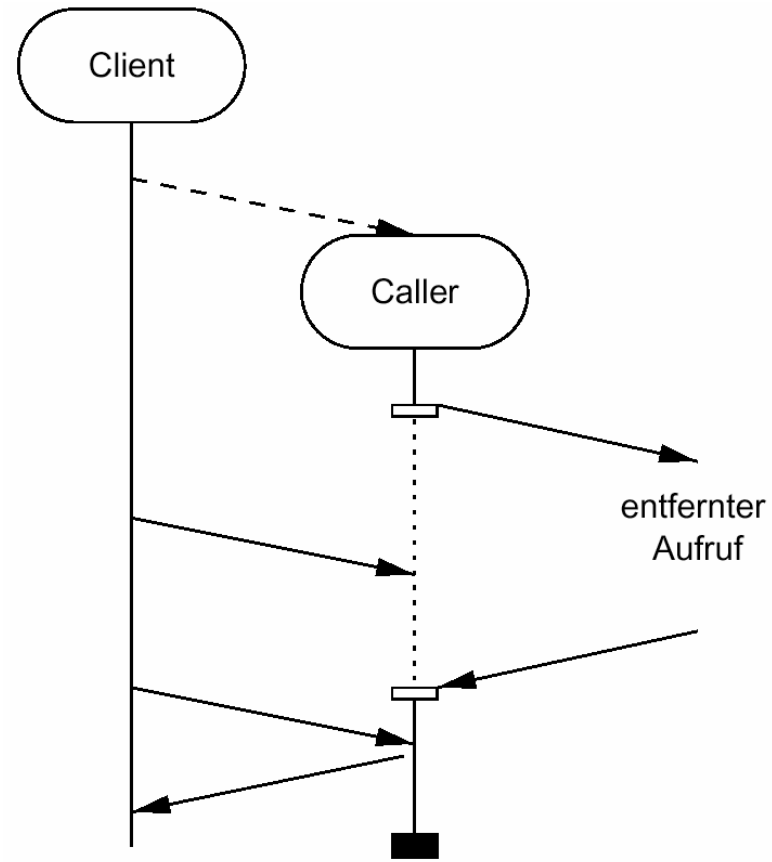
Frage: Wie wird das auf Prozessebene realisiert ?

Client-Server-Architektur

Callback:



Polling:



Problem: Die Kommunikationskanäle sind unzuverlässig !

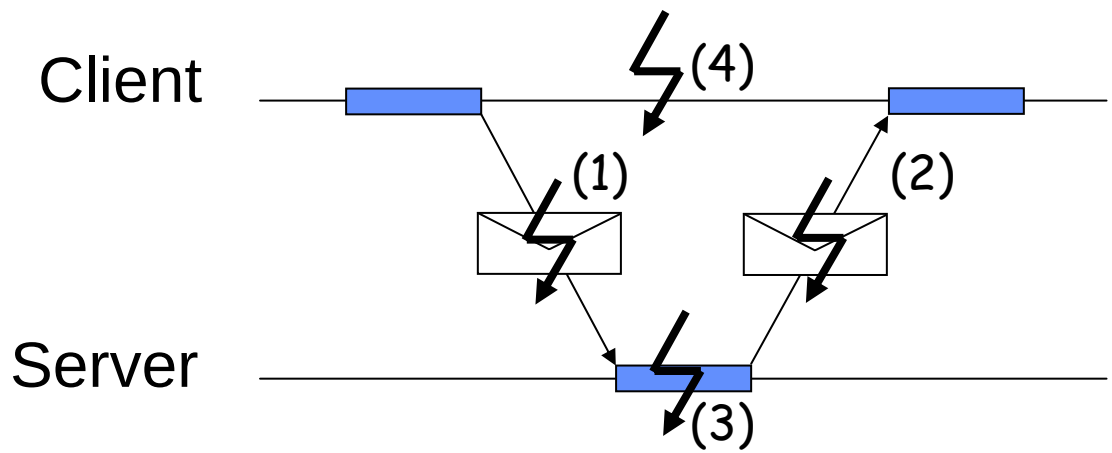
Fehlermöglichkeiten bei der Client-Server-Kommunikation

Verlust der Auftragsnachricht (1)

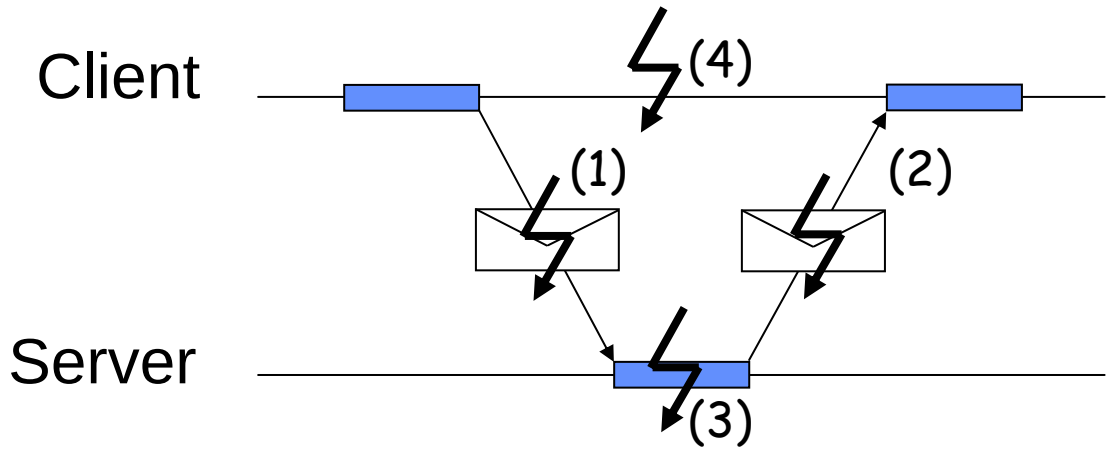
Verlust der Ergebnisnachricht (2)

Ausfall des Servers (3)

Ausfall des Klienten (4)



Einfache Fehlerbehebung und neue Probleme



Client wartet und versucht...

... nach Timeout ein erneutes Senden,

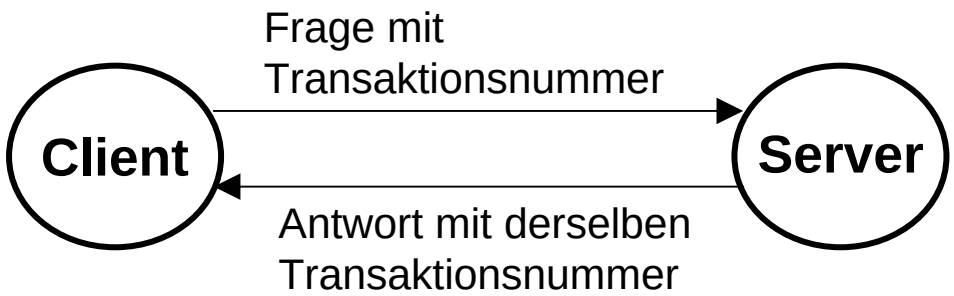
kann aber nicht zwischen verschiedenen Fehlersituationen unterscheiden.

Erneutes Senden führt zur erneuten Ausführung.

Client antizipiert eventuell neuen Zustand nicht.

Problem: **Wie erkennt der Server, dass der Client dieselbe Anfrage noch einmal gestellt hat und nicht eine neue gestellt hat ?**

Lösungsansatz: Protokolle / Transaktionskonzept



wichtig:

- **Transaktionsnummer ist im ganzen Netzwerk eindeutig !**

Gewisse Aktionen dürfen nur zusammen oder gar nicht ausgeführt werden:

- **Ein Protokoll definiert, welche Aktionen zu einer Transaktion zusammengefasst werden**
- **Jede Aktion hat dieselbe Transaktionsnummer und eine Ausführungsnummer, die seine Stellung innerhalb des Protokolls beschreibt**
- **Wenn eine Transaktion nicht ordnungsgemäß zu Ende geführt wurde, werden alle Aktionen dieser Transaktion rückgängig gemacht.**

Zusammenfassung:

Überblick über das Vorlesungskonzept
Allgemeine Anforderungen an Verteilte Systeme
Konzepte verteilter Hardware
Client-Server-Architektur (zweischichtig)