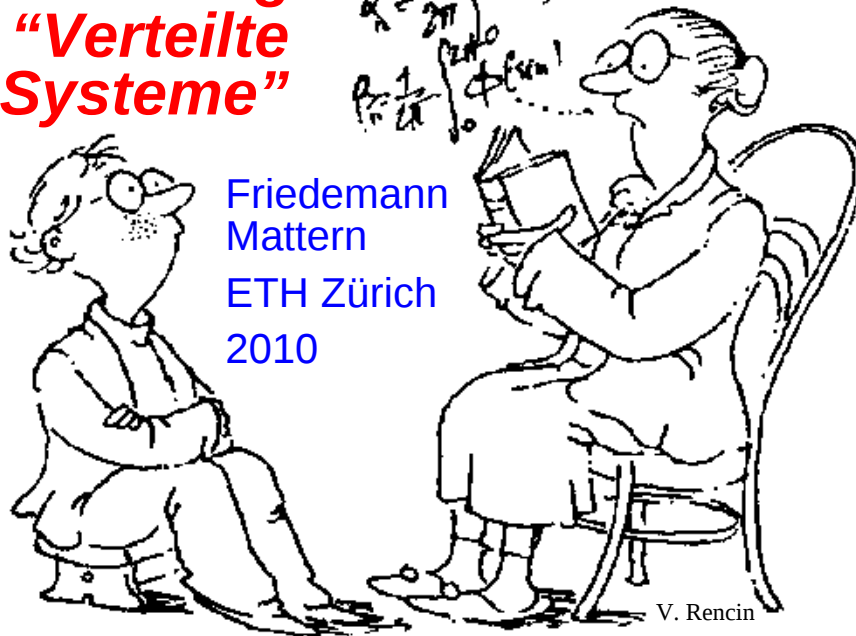


Vorlesung "Verteilte Systeme"

Friedemann
Mattern
ETH Zürich
2010

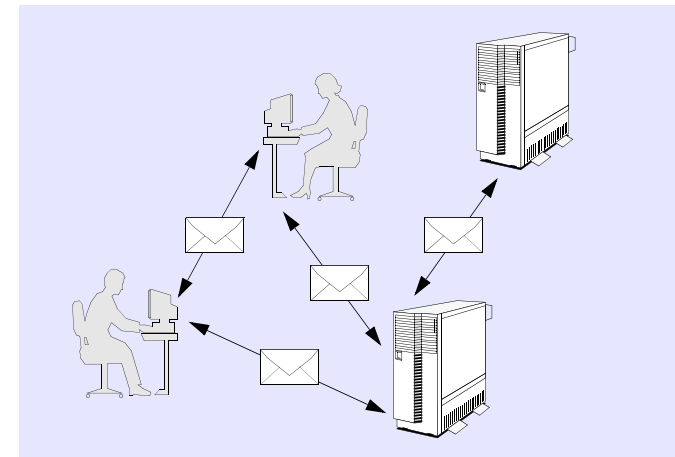


V. Rencin

Verteilte Systeme Teil 1

Friedemann Mattern

Institut für Pervasive Computing
Departement Informatik
ETH Zürich



- Rechner, Personen, Prozesse, "Agenten" sind an *verschiedenen Orten*.
- Autonome Handlungsträger, die jedoch gelegentlich *kooperieren* (und dazu über Nachrichten *kommunizieren*).

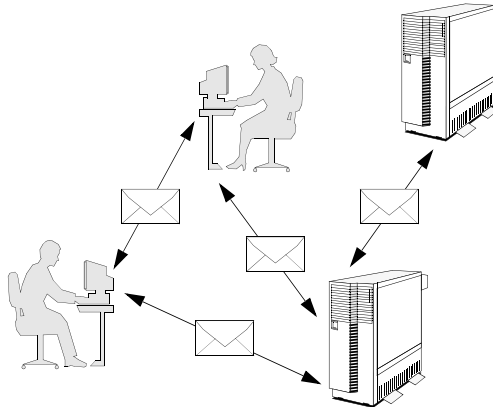
Wer bin ich? Wer sind wir?

Fachgebiet “**Verteilte Systeme**” im Departement Informatik, Institut für Pervasive Computing

- 15 (Ober)assistent(inn)en
- Forschungsprojekte

Mehr zu uns:
www.vs.inf.ethz.ch

- Infrastruktur für verteilte Systeme
- Internet der Dinge
- Ubiquitous Computing
- Sensornetze
- Verteilte Anwendungen und Algorithmen



Organisatorisches zur Vorlesung

Format: 6G+1A: Vorlesung, Praktikum, Übungen

Sinnvolle Vorkenntnisse (Grundlagen):

- 4 Semester der Bachelorstufe Informatik (inkl. Mathematik-Anteil)
- Computernetze
- Grundkenntnisse Betriebssysteme (z.B. Prozessbegriff, Synchronisation)
- UNIX, Java ist hilfreich

Mo 9:15 - 12:00, NO C 6

Fr 9:15 - 12:00, NO C 6

} Vorlesung, Praktikum, Übung

- Praktikum ist *komplementär* zur Vorlesung (mobile, kommunizierende Plattformen: Android, HTC Desire)
- Gelegentliche *Denkaufgaben* (ohne Lösung...) in der Vorlesung
- Gelegentliche *Übungsstunden* (zu den “Vorlesungsterminen”) zur Besprechung der Aufgaben und Vertiefung des Stoffes

Absicht!

- *Folienkopien* jeweils einige Tage nach der Vorlesung

- im .pdf-Format bei www.vs.inf.ethz.ch/edu

- Vorlesung ab November: *Prof. Roger Wattenhofer*

- Prüfung schriftlich

- bewertete Praktikumsaufgaben gehen zum Teil in die Prüfungsnote ein

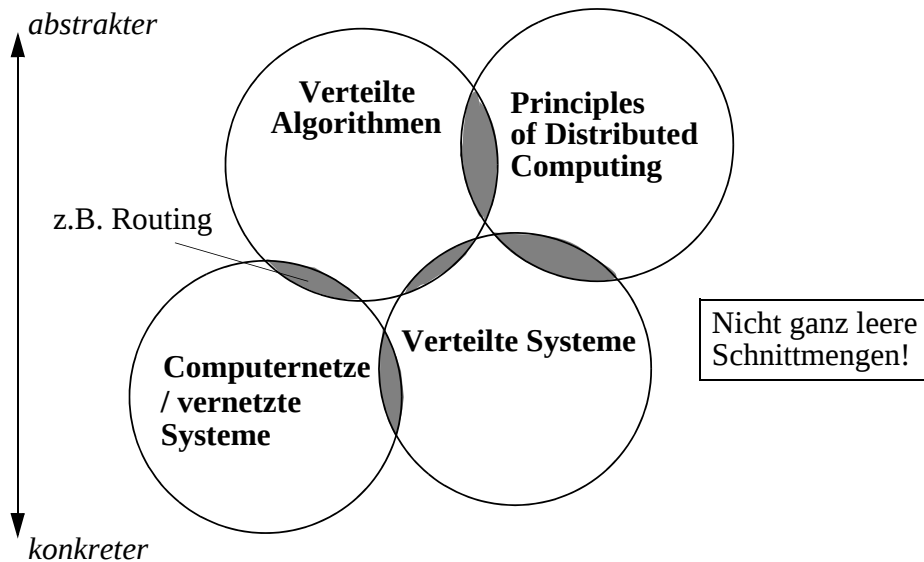
- Ansprechperson für organisatorische Aspekte:

- Matthias Kovatsch, kovatsch@inf.ethz.ch

Thematisch verwandte Veranstaltungen im Masterstudium

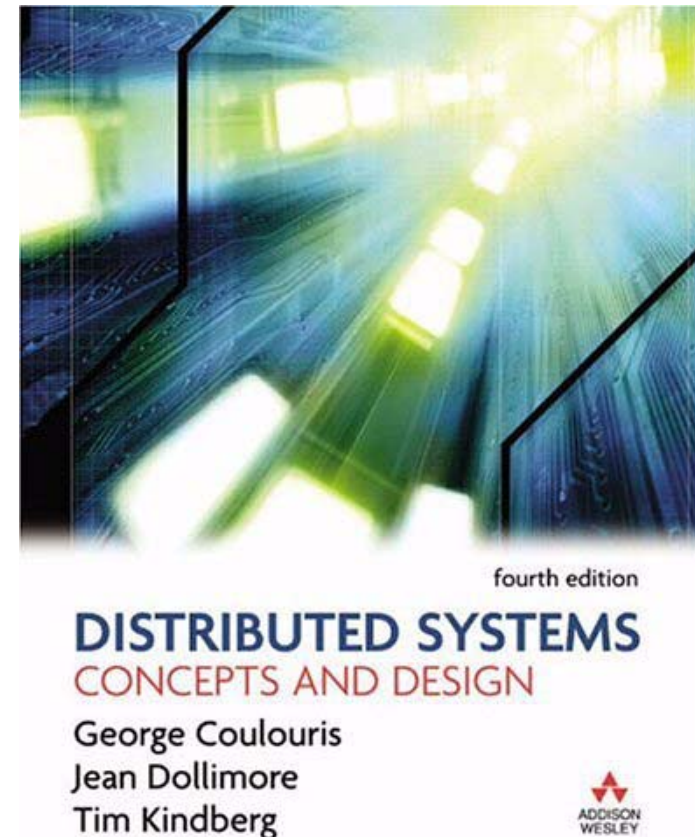
- Ubiquitous Computing
- Enterprise Application Integration - Middleware
- Web Services and Service Oriented Architectures
- Verteilte Algorithmen
- Principles of Distributed Computing
- ...

- Einschlägige Seminare
- Semester- und Masterarbeit
- Praktikum ("Lab")



Literatur

G. Coulouris, J. Dollimore, T. Kindberg: *Distributed Systems: Concepts and Design (4th ed.)*. Addison-Wesley, 2005



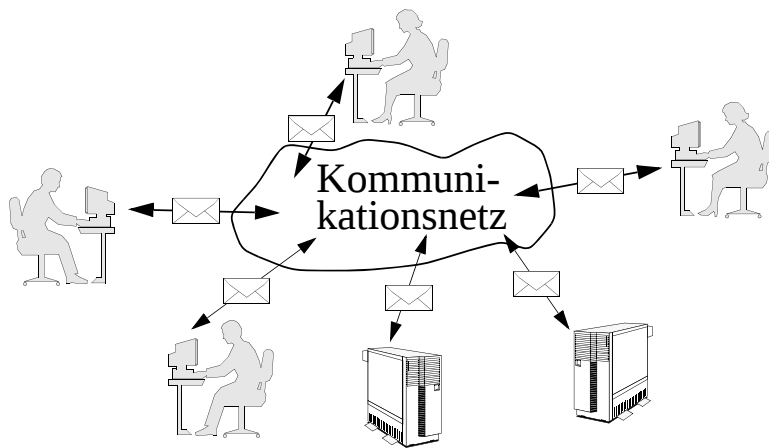
A. Tanenbaum, M. van Steen: *Distributed Systems: Principles and Paradigm (2nd ed.)*. Prentice-Hall, 2007

Oliver Haase: *Kommunikation in verteilten Anwendungen (2. Auflage)*. R. Oldenbourg Verlag, 2008

“Verteiltes System” - zwei Definitionen

A distributed computing system consists of multiple autonomous processors that do not share primary memory, but cooperate by sending messages over a communication network.

-- H. Bal

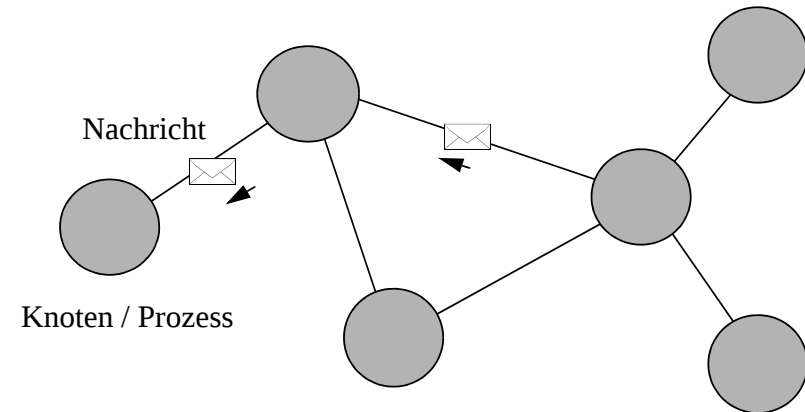


A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

-- Leslie Lamport

- welche Problemaspekte stecken hinter Lamports Charakterisierung?

“Verteiltes System”



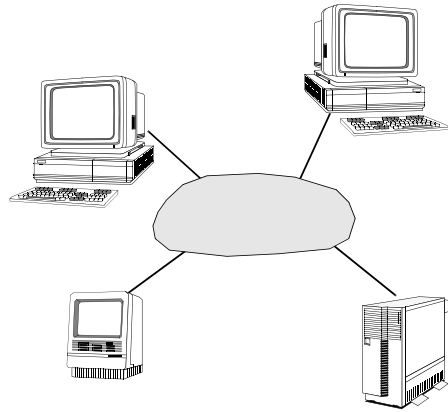
Physisch verteiltes System:

Mehrrechnersystem ... Rechnernetze

Logisch verteiltes System: Prozesse (Objekte, Agenten)

- Verteilung des Zustandes (keine globale Sicht)
- Keine gemeinsame Zeit (globale, genaue Uhr)

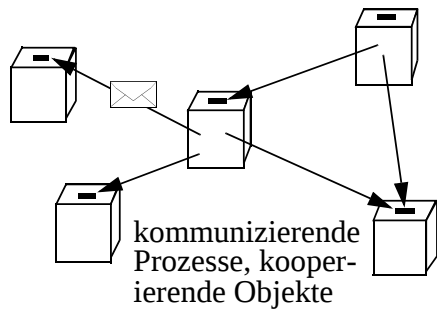
Sichten verteilter Systeme



Computernetz mit “Rechenknoten”:

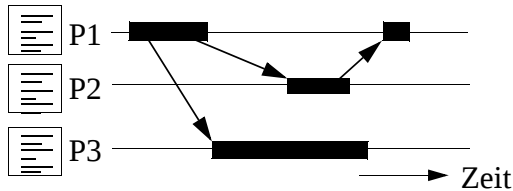
- Compute-Cluster
- Local Area Network
- Internet

⇒ Routing, Adressierung,...



Objekte in
*Betriebssystemen,
Middleware,
Programmiersprachen*

⇒ “Programmiersicht”
(Client, Server,...)

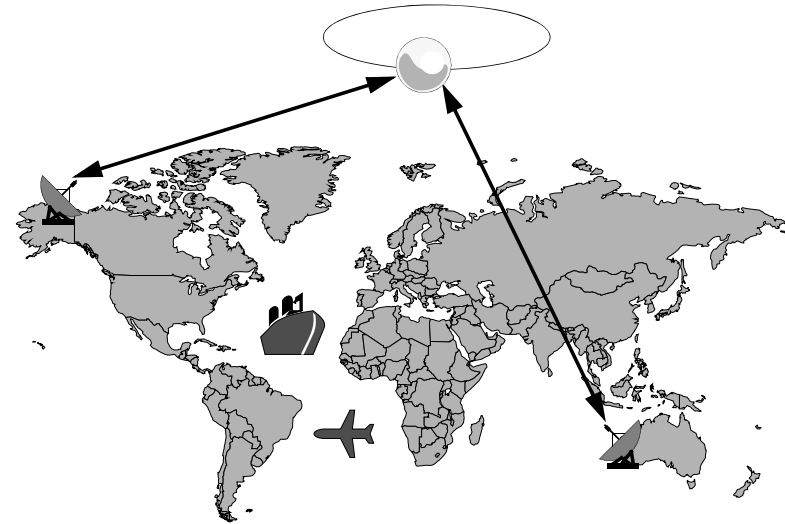


Algorithmen-
und Protokoll-
ebene

- Aktionen, Ereignisfolgen
- Konsistenz, Korrektheit

zunehmende Abstraktion

Die verteilte Welt



Auch die “reale Welt” ist ein verteiltes System:

- Viele gleichzeitige (“parallele”) Aktivitäten
- Exakte globale Zeit nicht erfahrbar / vorhanden
- Keine konsistente Sicht des Gesamtzustandes
- Kooperation durch explizite Kommunikation
- *Ursache* und *Wirkung* zeitlich (und räumlich) getrennt

Warum verteilte Systeme?

- Es gibt *inhärent geographisch verteilte* Systeme
 - z.B. Zweigstellennetz einer Bank; Steuerung einer Fabrik (Zusammenführen / Verteilen von Information)
 - *Electronic commerce*
 - kooperative Informationsverarbeitung räumlich getrennter Institutionen (z.B. Reisebüros, Kreditkarten,...)
 - Mensch-Mensch-*Telekommunikation*
 - E-Mail, Diskussionsforen, Blogs, digitale soz. Netze, IP-Telefonie,...
 - *Globalisierung* von Diensten
 - Skaleneffekte, Outsourcing,...
-
- Wirtschaftliche Aspekte
 - Cluster von einfachen Computern oder Netz von PCs manchmal besseres Preis-Leistungsverhältnis als Grosscomputer
 - Outsourcing von Diensten manchmal wirtschaftlicher
 - vgl.: Virtualisierung, dynamische Lastverteilung, Cloud Computing

Verteilte Systeme als “Verbunde”

- Verteilte Systeme *verbinden* räumlich (oder logisch) getrennte Komponenten zu einem bestimmten *Zweck*
 - *Systemverbund*
 - gemeinsame Nutzung von Betriebsmitteln, Geräten,....
 - einfache inkrementelle Erweiterbarkeit
 - *Funktionsverbund*
 - Kooperation bzgl. Nutzung jeweils spezifischer Eigenschaften
 - *Lastverbund*
 - Zusammenfassung der Kapazitäten
 - *Datenverbund*
 - allgemeine Bereitstellung von Daten
 - *Überlebensverbund*
 - Redundanz durch Replikation
 - dann i.Allg. nur Ausfall von Teilfunktionalität

Historische System-Entwicklung

Rechner-zu-Rechner-Kommunikation

- Zugriff auf entfernte Daten (“Datenfernübertragung”, DFÜ)
- dezentrale Informationsverarbeitung war zunächst ökonomisch nicht sinnvoll (zu teuer, Fachpersonal nötig)
→ Master-Slave-Beziehung (“Remote Job Entry”, Terminals)

ARPA-Netz (Prototyp des Internet)

- “symmetrische” Kommunikationsbeziehung (“peer to peer”)
- Internet-Protokollfamilie (TCP/IP,...)
- file transfer (ftp), remote login, E-Mail

Workstation-Netze (LAN)

- bahnbrechende, frühe Ideen bei XEROX-PARC
(XEROX-Star als erste Workstation, Desktop-Benutzerinterface, Ethernet, RPC, verteilte Dateisysteme,...)

Kommerzielle Pionierprojekte als Treiber

- z.B. Reservierungssysteme, Banken, Kreditkarten

WWW (und Internet) als Plattform

- für electronic commerce etc.
- XML, web services, peer to peer,...
- neue, darauf aufbauende Dienste

Mobile Geräte (Smartphones etc.)

Internet der Dinge?

“Historische” Konzepte

- Concurrency, Synchronisation,...
 - war bereits klassisches Thema bei Datenbanken und Betriebssystemen
- Programmiersprachen
 - kommunizierende Objekte
- Physische Parallelität
 - z.B. Multiprozessoren
- Parallele und verteilte Algorithmen
- Semantik
 - mathematische Modelle für Verteiltheit (z.B. CCS, Petri-Netze)
- Abstraktionsprinzipien
 - Schichten, Dienstprimitive,...
- Verständnis grundlegender Phänomene der Verteiltheit
 - Konsistenz, Zeit, Zustand,...

Entwicklung “guter” Konzepte, Modelle, Abstraktionen etc. zum Verständnis der Phänomene dauert oft lange

- notwendige Ordnung und Sichtung des verfügbaren Gedankenguts

Diese sind jedoch für die Lösung praktischer Probleme hilfreich, oft sogar notwendig!

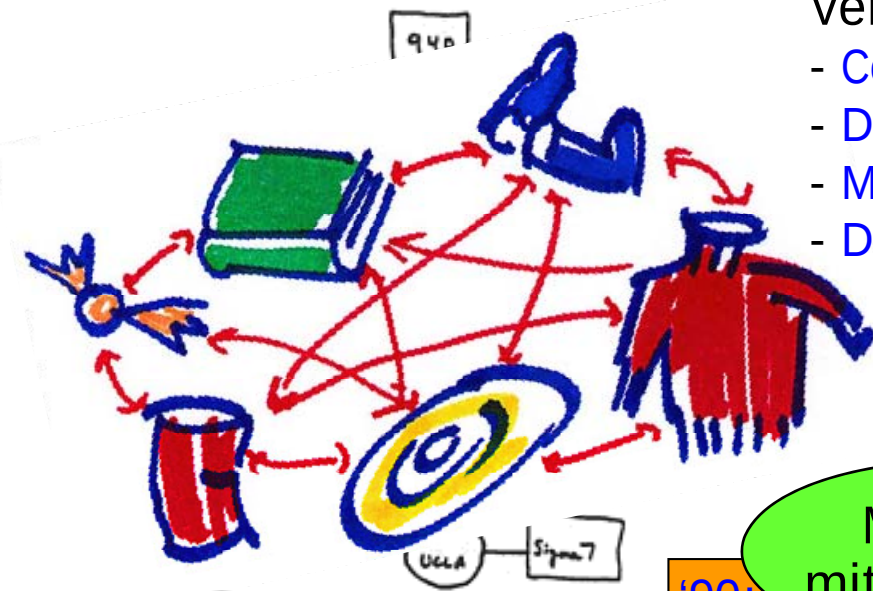
Software-Infrastruktur für Internet-basierte Anwendungen?

- Phänomen: das Internet verbreitet sich immer weiter
 - mehr Nutzer, Popularisierung
 - bis an die Person (Handy)
 - immer “exotischere” Endgeräte (TV, Kühlschrank, Chipkarte)
 - bald enthalten vielleicht auch viele Supermarktprodukte, Kleidungsstücke etc. kommunikationsfähige Chips
 - Sensoren
 - Mobile Geräte, dynamische Umgebungen
 - Es entstehen neue Dienste im Netz
 - Dienste interagieren miteinander
 - Kompatibilität, Standards, Protokolle, offene Schnittstellen,...
 - Markt erfordert sehr schnelle Reaktion
 - schnelle Implementierung neuer Dienste
 - Update über das Netz
 - Anschluss neuer Geräte muss “von selbst” erfolgen
 - Integration in eine Infrastruktur und Umgebung von Ressourcen
-
- Kann man eine Infrastruktur schaffen, die das unterstützt?
 - Welche Systemarchitektur ist hierfür geeignet?

Änderung der Vernetzungs“qualität“

Vernetzung von:

- Computern (ftp)
- Dokumenten (WWW)
- Menschen (soz. Netze)
- Dingen (Internet der Dinge)



Mensch
mit Mensch

E-Mail

TCP

Mensch
mit Maschine

WWW

http

'00:

Mobiler
Zugang

Web 2.0

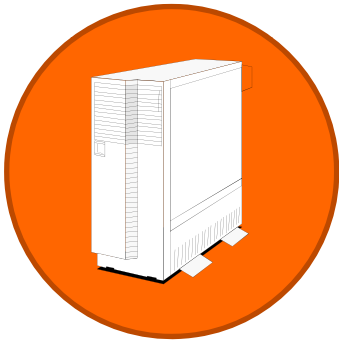
Mensch
mit Ding

Eingebettete
Internet-
Dienste

2010

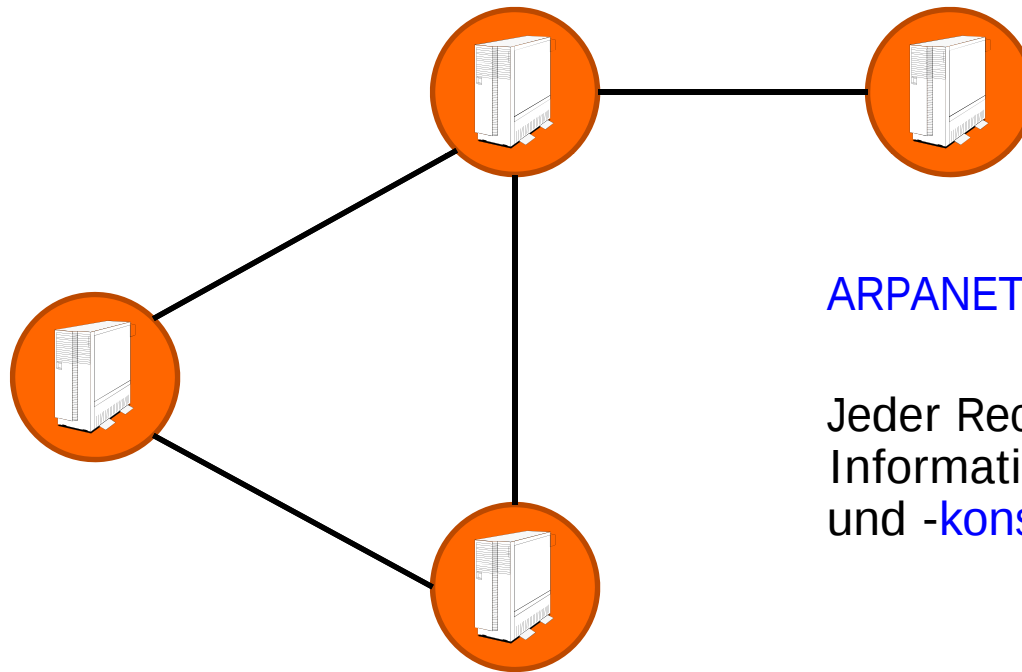
Architekturen verteilter Systeme

- Zu Anfang waren Systeme **monolithisch**



- Nicht verteilt / vernetzt
- **Mainframes**
- **Terminals** als angeschlossene „Datensichtgeräte“ („Datenendgerät“: Fernschreiber, ASCII)

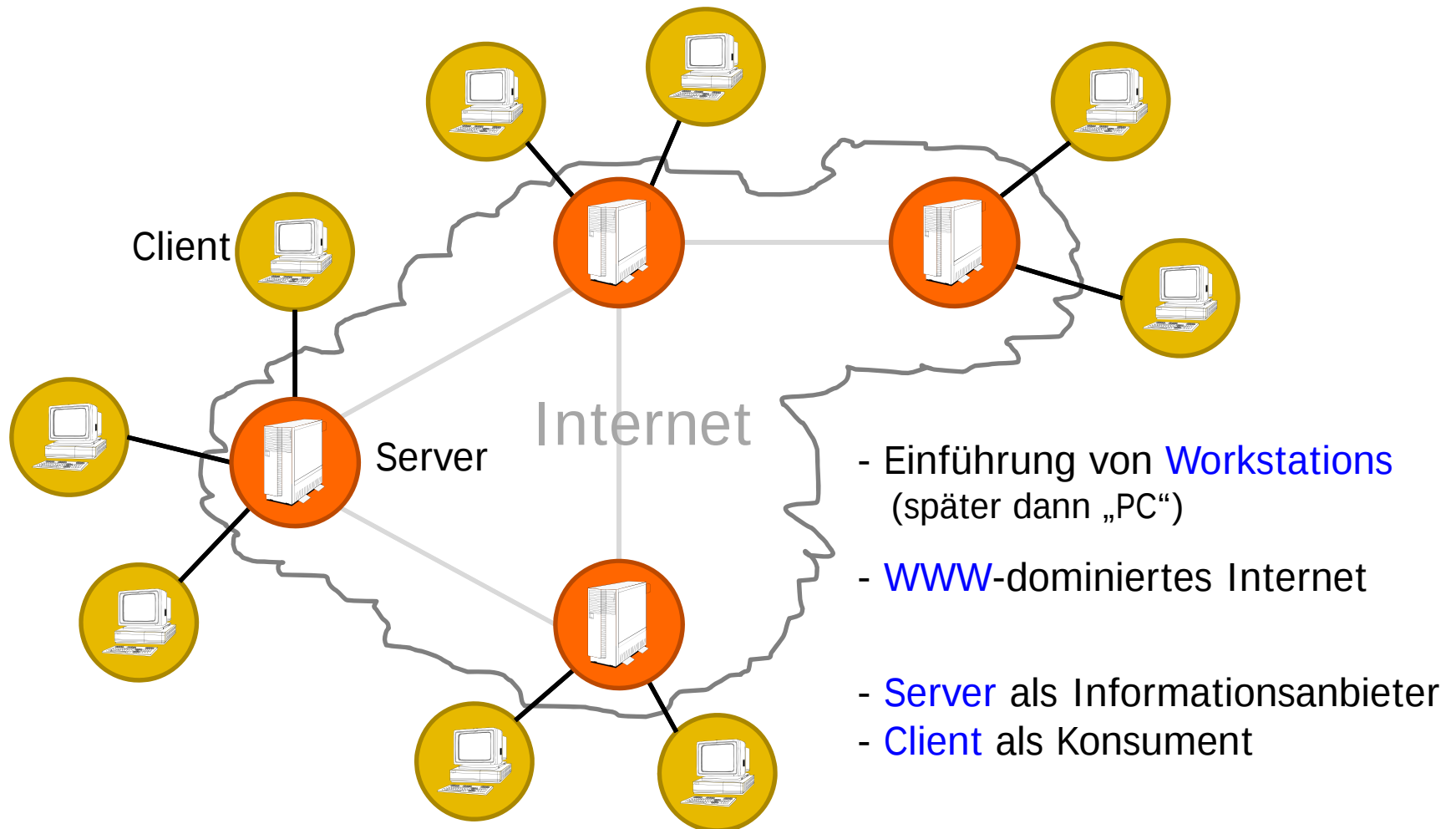
Architekturen verteilter Systeme: **Peer-to-Peer**



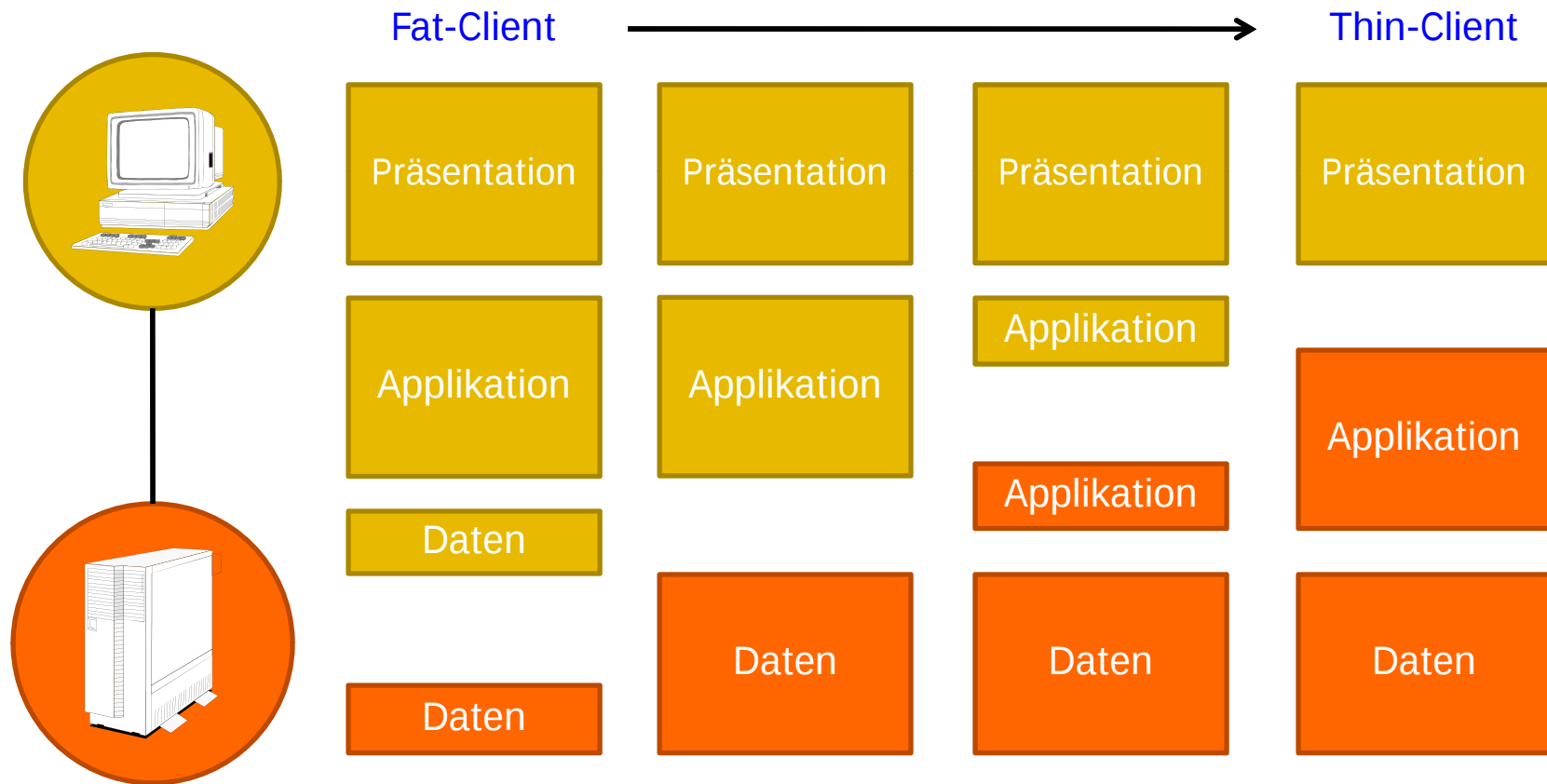
ARPANET 1969

Jeder Rechner **gleichzeitig**
Informations**anbieter**
und -**konsument**

Architekturen verteilter Systeme: **Client-Server**



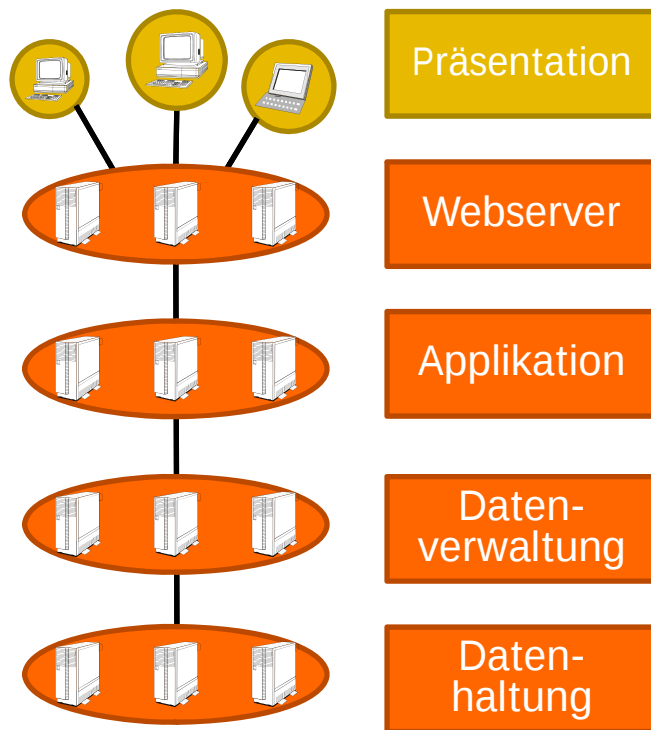
Architekturen verteilter Systeme: **Fat-** und **Thin-Client**



Architekturen verteilter Systeme: 3-Tier



Architekturen verteilter Systeme: **Multi-Tier**

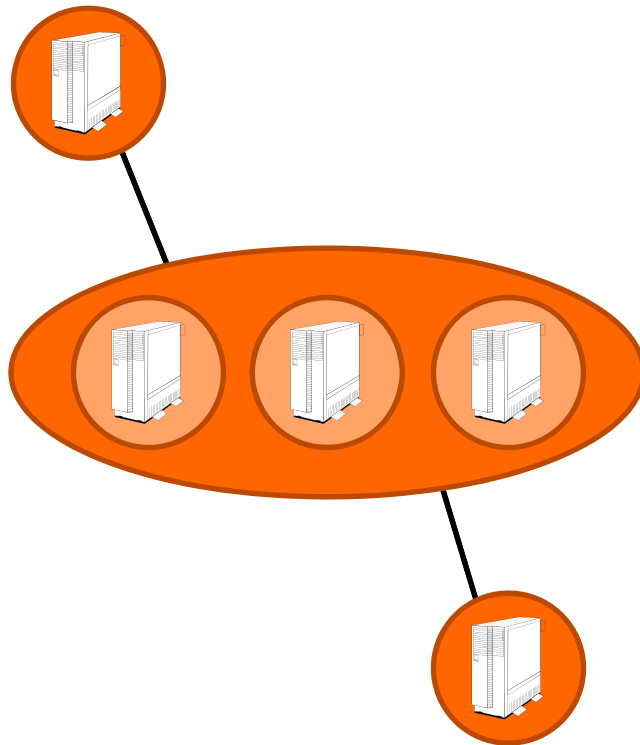


Weitere Schichten sowie mehrere physikalische Einheiten pro Schicht erhöhen die **Skalierbarkeit** und **Flexibilität**

Mehrere Webserver ermöglichen z.B. **Lastverteilung**

Verteilte Datenbanken in der Datenhaltungsschicht bietet Sicherheit durch Replikation und hohen Durchsatz

Architekturen verteilter Systeme: Service-Oriented Architecture (SOA)



Eine **Unterteilung** der Applikation in einzelne, unabhängige Abläufe innerhalb eines **Geschäftsprozesses** erhöht die Flexibilität weiter

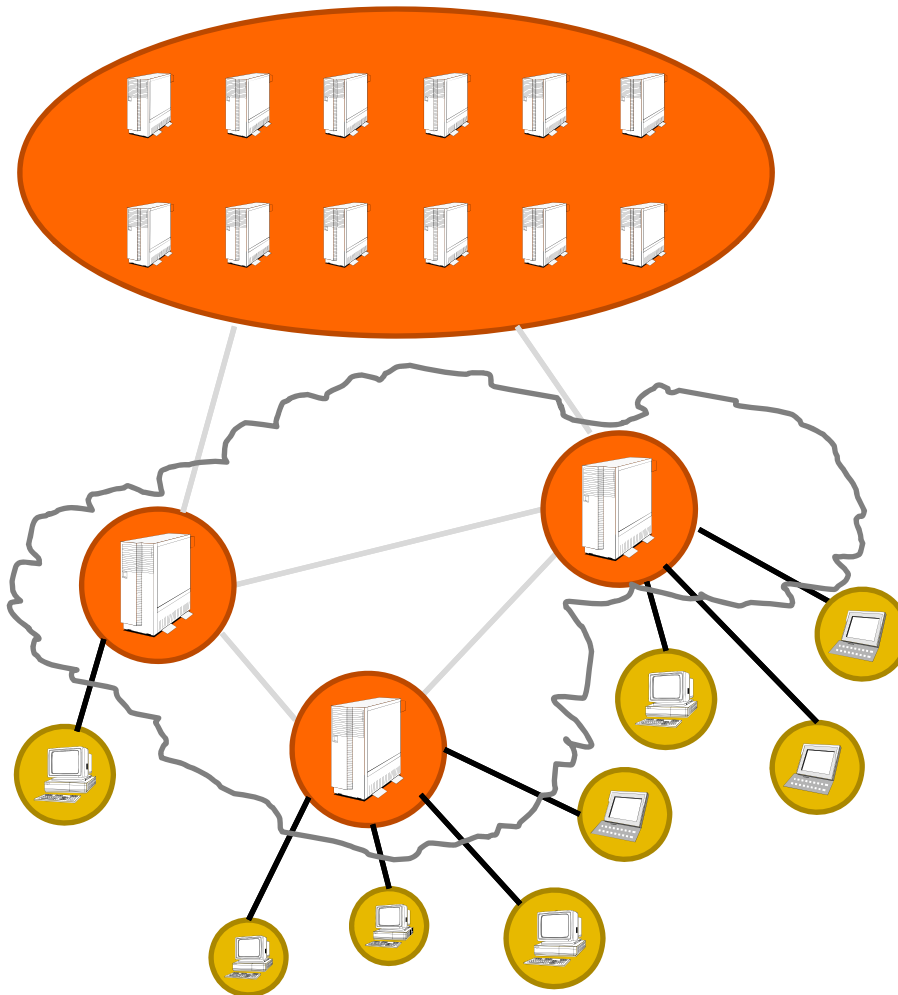
Lose Kopplung zwischen Services über Nachrichten und events (statt RPC)

Services können bei Änderungen der Prozesse **einfach neu zusammengestellt** werden („development by composition“)

Services können auch von **externen Anbietern** bezogen werden

Oft in Zusammenhang mit **Web-Services**

Architekturen verteilter Systeme: **Cloud-Computing**

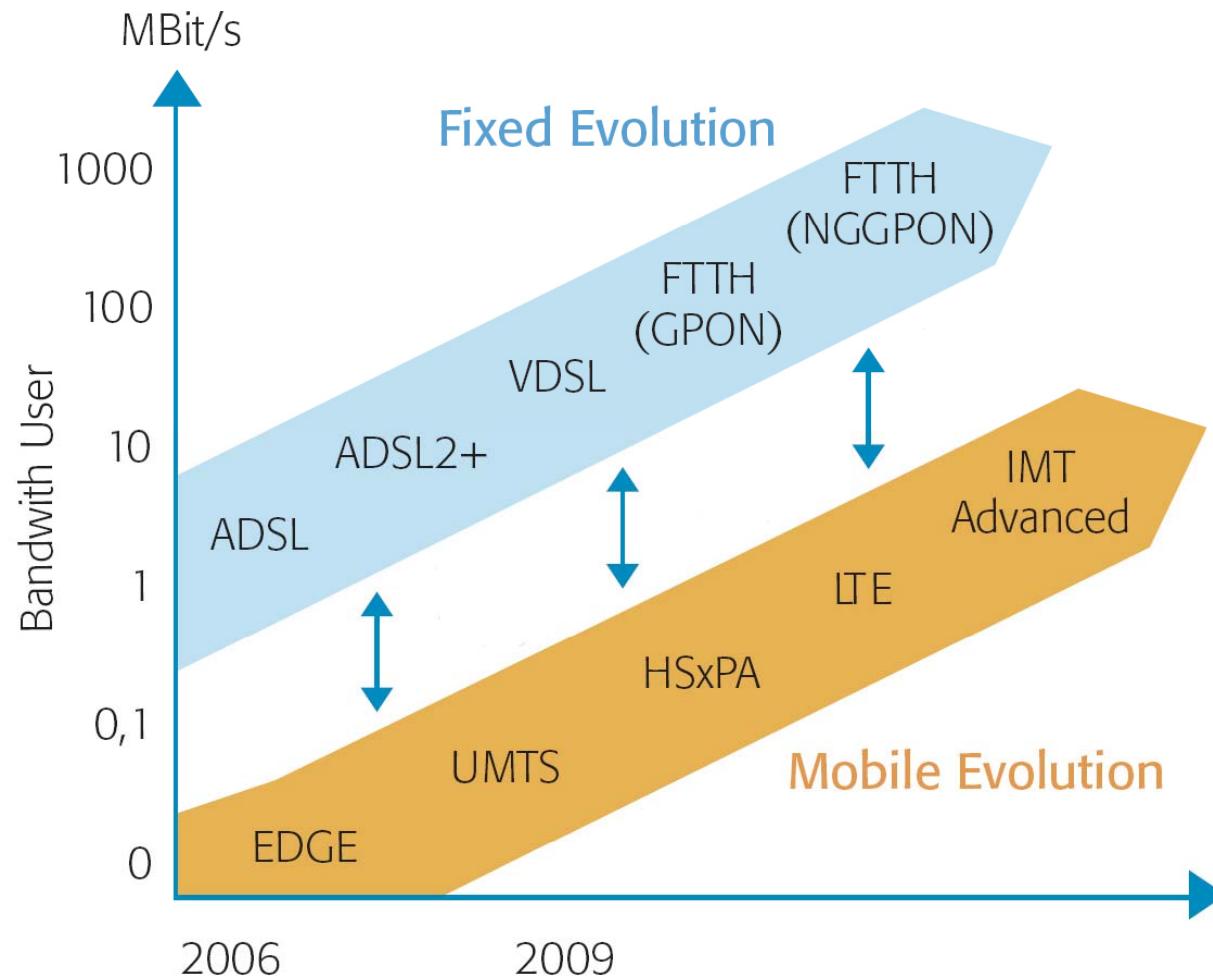


Massive Bündelung der
Rechenleistung an zentraler Stelle

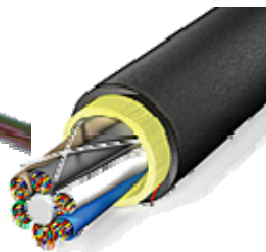
Internet als Vermittlungsinstanz



Motivierender Trend: Stetige Erhöhung der Bandbreite für Endnutzer



Hochgeschwindigkeit ins Haus



- Konvergenz TV, Telekommunikation und Internet
- Technologiewechsel → erhebliche Investitionen
 - wirtschaftliche Faktoren und Bedingungen

- Telefondraht → Internet → TV
 - TV-Kabel → Telefon → Internet
- } → Glasfaser (x1, x5, x10, ...?)
- „Tripleplay“ (Sprache, Daten, Video)

Cloud-Computing



E-Mail wird beim
Provider gespeichert



Cloud-Computing



Fotos werden bei
flickr gespeichert

Cloud-Computing



Videos bei **You**Tube

Cloud-Computing



Private Dokumente
werden bei einem
Storage Provider
abgelegt

Das Tagebuch wird
öffentlich „im Netz“
als Blog geführt

Cloud-Computing



Informieren tut man
sich im Netz

Google



WIKIPEDIA
Die freie Enzyklopädie

Cloud-Computing



Vernetzen tut man
sich bei „social net-
works“ oder „digital
communities“ im Netz



Cloud-Computing



Plattformen im Netz
nutzt man zum

- Kaufen
- Spielen
- Kommunizieren

- ...

GoogleTM
Romance BETA

Cloud-Computing



Vorteile für Nutzer:

- von überall zugreifbar
- keine Datensicherung
- keine Softwarepflege

Kein PC, sondern
billiges Web-Terminal,
Smartphone etc.

Wie Wasserleitungen einst den eigenen Brunnen überflüssig machten...

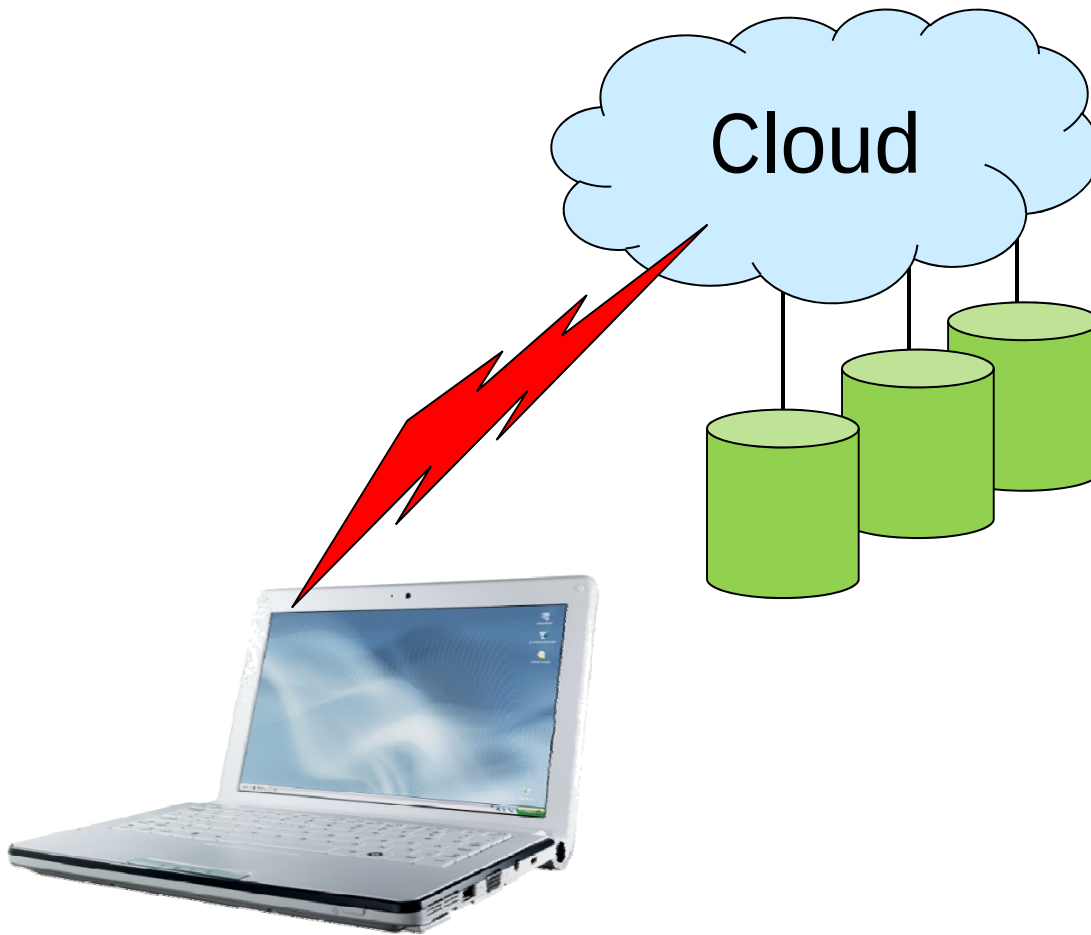
Cloud-Computing



Voraussetzungen?

- Überall Breitband (fest & mobil)
- Netz-Verlässlichkeit (Versorgungssicherheit, Datenschutz,...)
- Wirtschaftlichkeit

Verlässlichkeit?



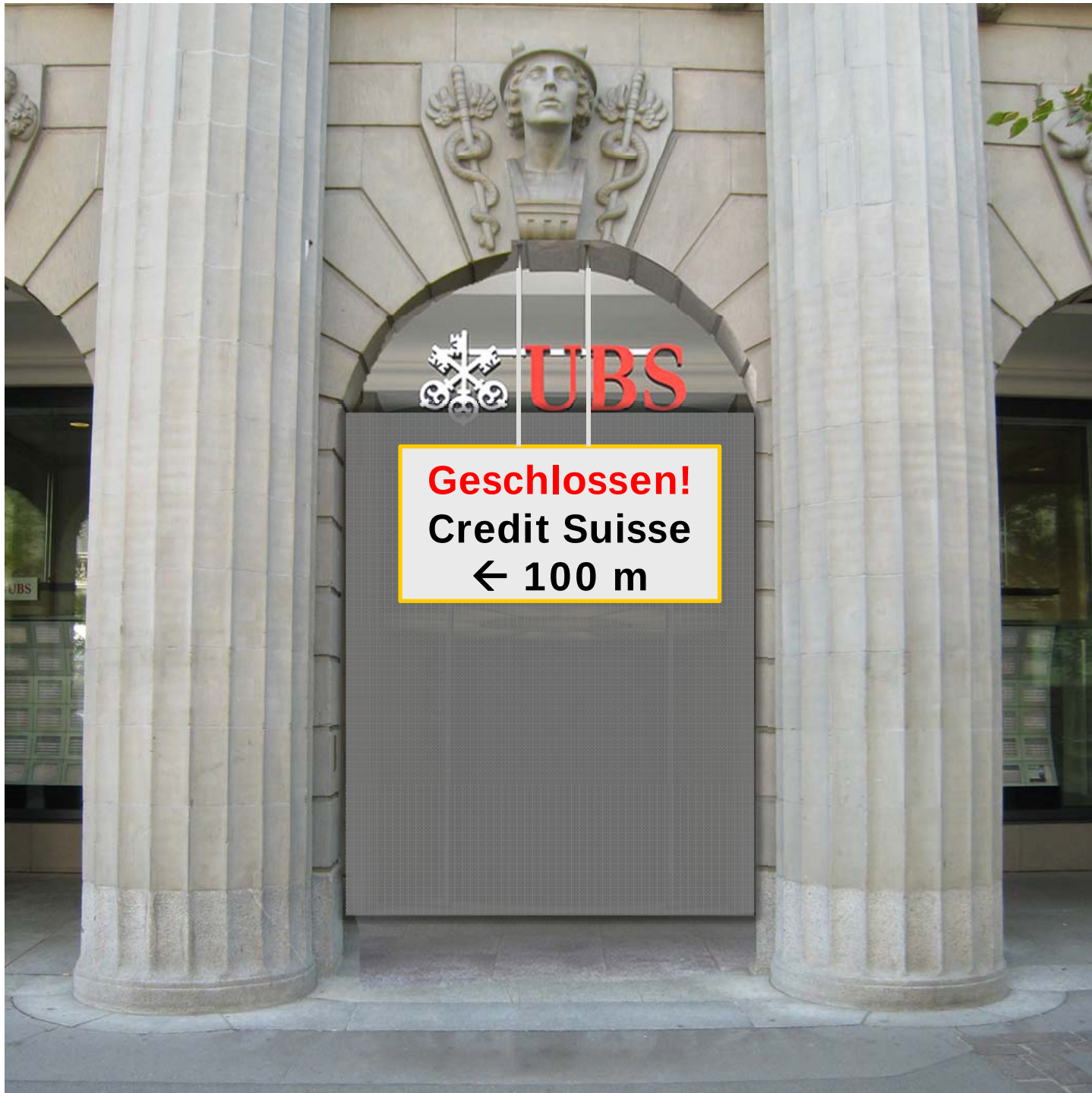
Voraussetzungen?

- Überall Breitband (fest & mobil)
- Netz-Verlässlichkeit (Versorgungssicherheit, Datenschutz,...)
- Wirtschaftlichkeit

The Xdrive service is closed.

Thank you for having been an Xdrive user.





Cloud-Computing



Plattformen:

- Wer betreibt sie? Wo?
- Wer verdient daran?
- Wer bestimmt?
- Wer kontrolliert?
- Welche Nationen profitieren davon?

Beispiel: Google-Datenzentren



- Jedes Datenzentrum hat 10 000 – 100 000 Computer
- Kostet über 500 Mio \$ (Bau, Infrastruktur, Computer)
- Verbraucht 50 – 100 MW Energie (Strom, Kühlung)
- Neben Google weitere (z.B. Amazon, Microsoft, Ebay,...)

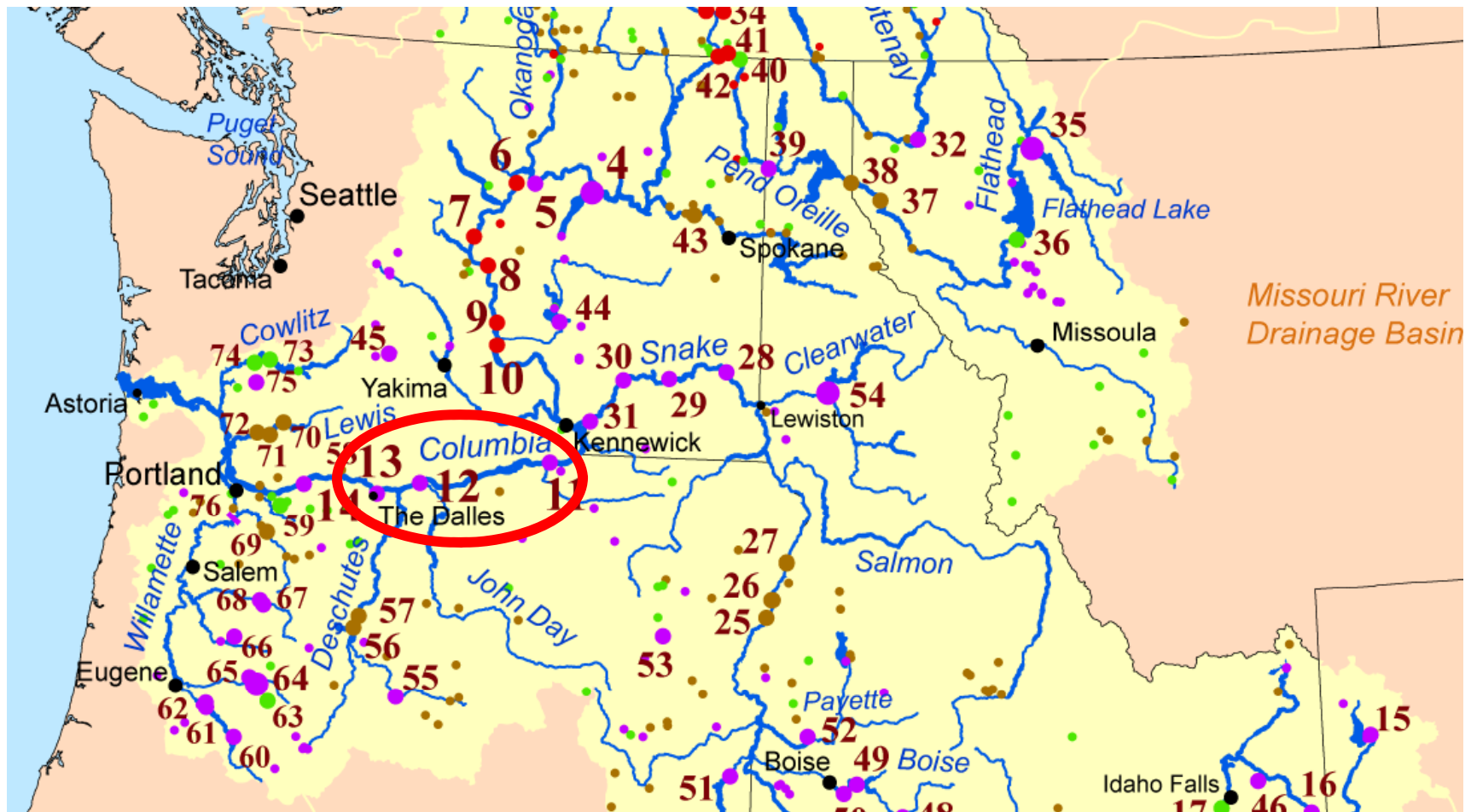
Google Data Center Groningen



Google Data Center Columbia River



The Dalles, OR, Columbia River

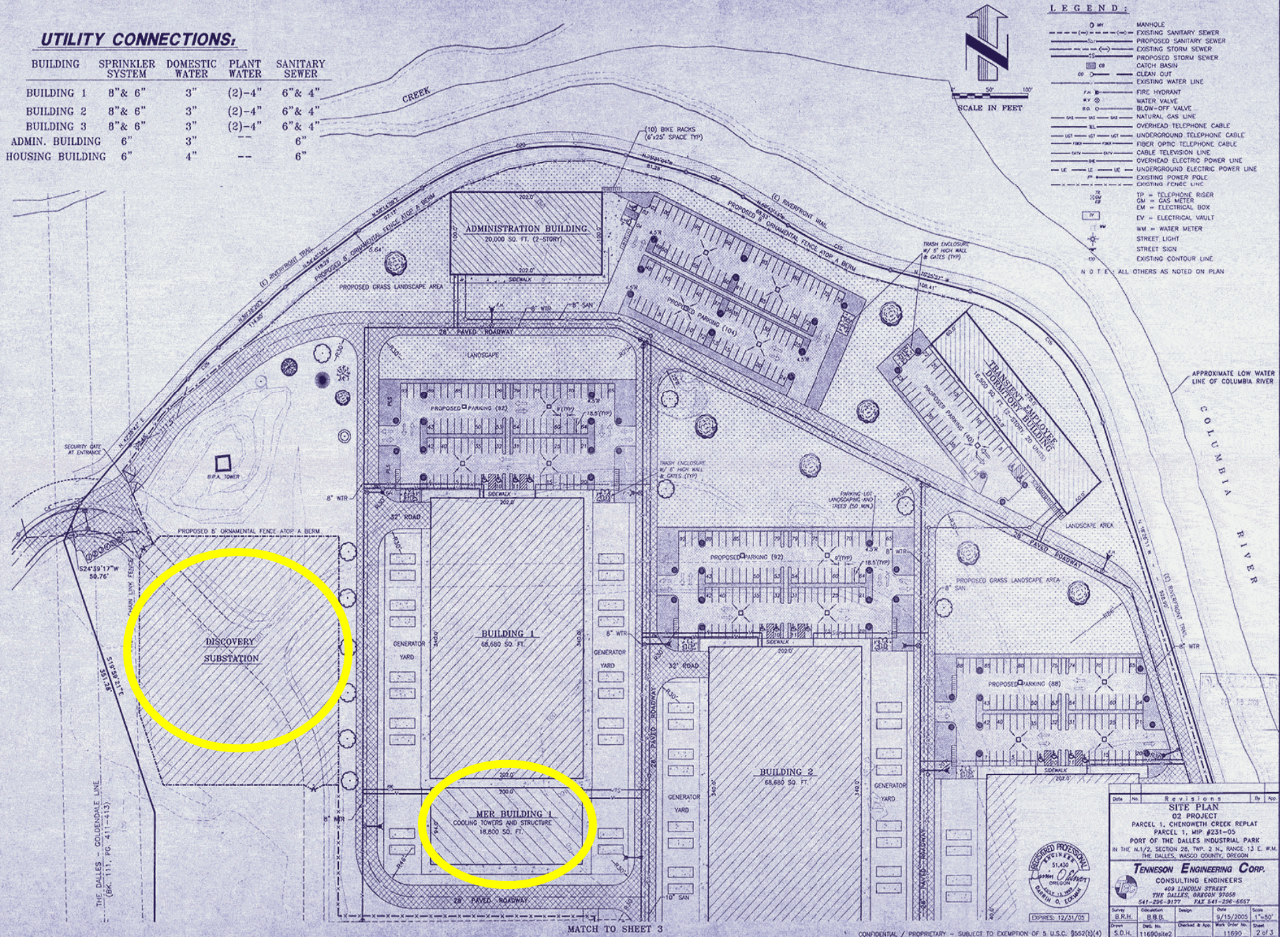


Google Data Center Columbia River



UTILITY CONNECTIONS:

BUILDING	SPRINKLER SYSTEM	DOMESTIC WATER	PLANT WATER	SANITARY SEWER
BUILDING 1	8" & 6"	3"	(2)-4"	6" & 4"
BUILDING 2	8" & 6"	3"	(2)-4"	6" & 4"
BUILDING 3	8" & 6"	3"	(2)-4"	6" & 4"
ADMIN. BUILDING	6"	3"	--	6"
HOUSING BUILDING	6"	4"	--	6"



MATCH TO SHEET 3

CONFIDENTIAL / PROPRIETARY - SUBJECT TO EXEMPTION OF 5 U.S.C. §552(b)(4)



Revisions			
Date	No.	By	App.
SITE PLAN 02 PROJECT PARCEL 1, CHEROKEE CREEK REPLAY PARCEL 1, MIP #231-05 PORT OF THE DALLES INDUSTRIAL PARK IN THE N.1/2, SECTION 28, TWP. 2 N., RANGE 13 E. W.M. THE DALLES, WASCO COUNTY, OREGON			
TENNESSEE ENGINEERING CORP. CONSULTING ENGINEERS 409 LINCOLN STREET THE DALLES, OREGON 97058 541-296-9177 FAX 541-296-6657			
Survey	B.R.H.	Design	9/15/2005
Drawn	S.O.H.	Checked & App.	11/6/2005
Scale	1"=50'	Sheet	2 of 3

Across the street from the Google data center is an idle Northwest Aluminum smelter that once used 85 megawatts. In 2000, when energy prices soared, it was decommissioned, and it now is being

Across the street from the Google data center is an idle Northwest Aluminum smelter that once used 85 megawatts. In 2000, when energy prices soared, it was decommissioned, and it now is being

Across the street from the Google data center is an idle Northwest Aluminum smelter that once used 85 megawatts. In 2000, when energy prices soared, it was decommissioned, and it now is being

Energiezufuhr „Discovery Substation”: 115 kV / 13.8 kV

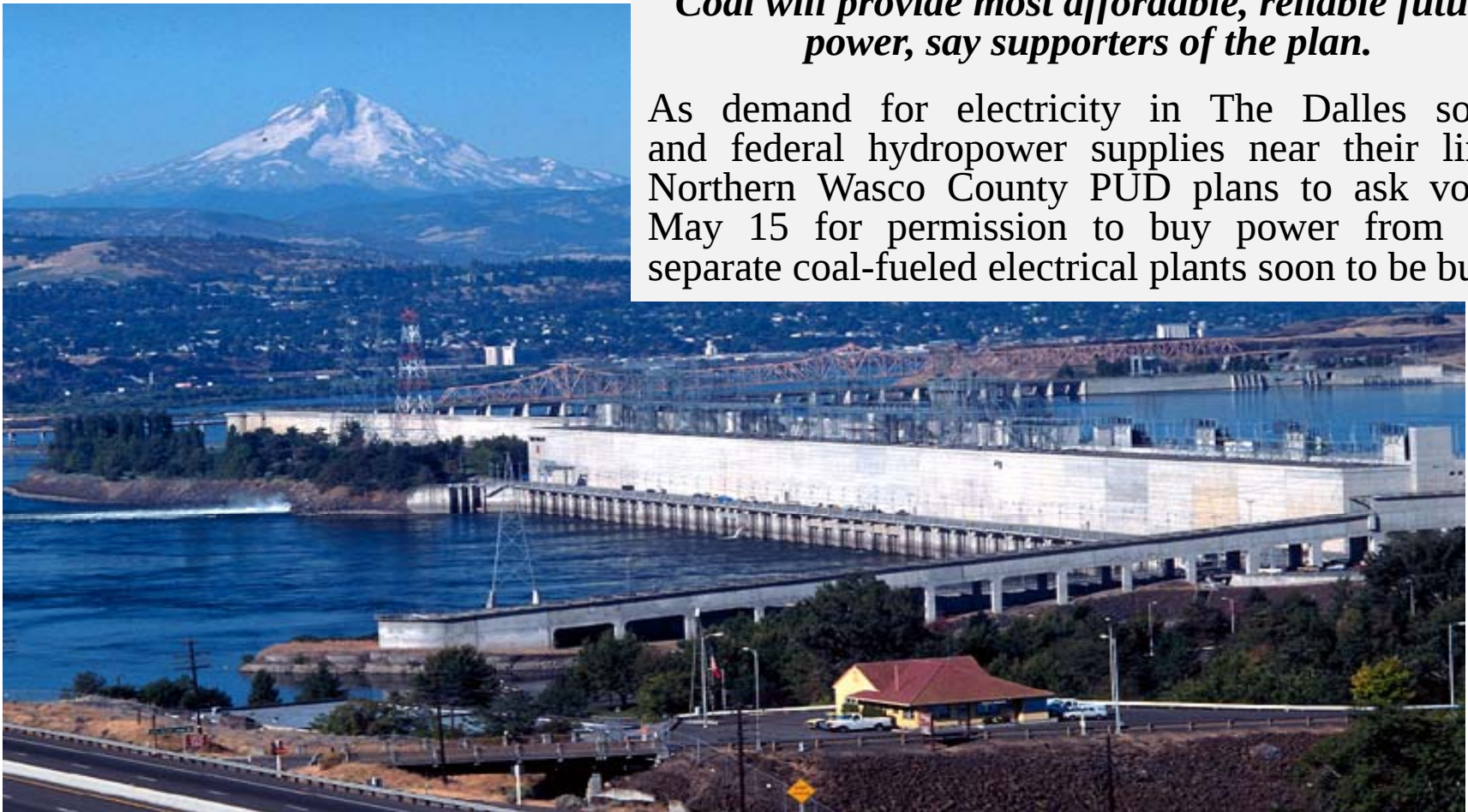


Nahes Kraftwerk: Dalles Dam Power Station, Columbia River

The Chronicle, March 1, 2007 –
PUD to seek vote on coal power

Coal will provide most affordable, reliable future power, say supporters of the plan.

As demand for electricity in The Dalles soars, and federal hydropower supplies near their limit, Northern Wasco County PUD plans to ask voters May 15 for permission to buy power from two separate coal-fueled electrical plants soon to be built.



Innenansicht eines Cloud-Zentrums

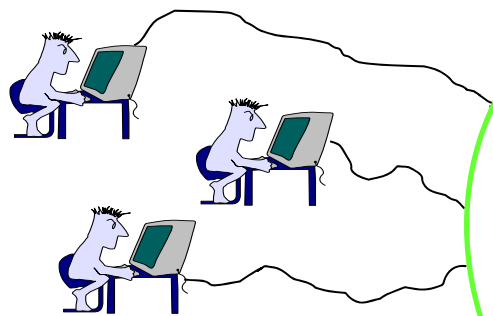


- Effizient wie Fabriken
 - Produkt: Internet-Dienste
- Kostenvorteil durch Skaleneffekt
 - Faktor 5 – 7 gegenüber traditionellen „kleinen“ Rechenzentren
- Angebot nicht benötigter Leistung auf einem Spot-Markt

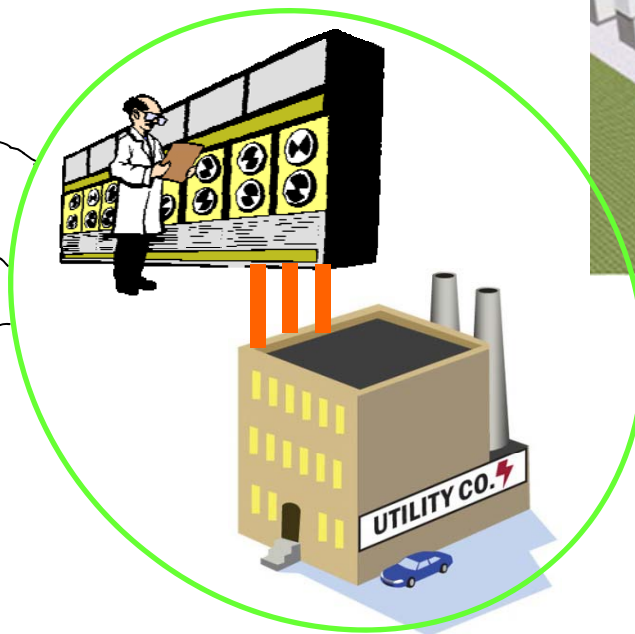
Das entwickelt sich zum
eigentlichen Geschäft!

Zukünftige Container-Datenzentren

- Hunderte von **Containern** aus je einigen tausend Compute-Servern
 - mit Anschlüssen für Strom und Kühlung
- Nahe an **Kraftwerken**
 - Transport von Daten billiger als Strom



Thin clients?



Cloud-Computing für die Industrie und Wirtschaft

- **Spontanes Outsourcen** von IT inklusive Geschäftsprozesse
 - Datenverarbeitung als Commodity
 - Software und Datenspeicher als Service
- Keine Bindung von Eigenkapital
 - Kosten nach „Verbrauch“
- **Elastizität**: Sofortiges Hinzufügen weiterer Ressourcen bei Bedarf
 - virtualisierte Hardware



Markt für „utility computing“
2010: ca. 95 Milliarden EUR

Charakteristika und “praktische” Probleme verteilter Systeme

- Räumliche Separation, autonome Komponenten

→ Zwang zur Kommunikation per Nachrichtenaustausch

→ neue Probleme:

- partielles Fehlverhalten (statt totaler “Absturz”)
- fehlender globaler Zustand / exakt synchronisierte Zeit
- Inkonsistenzen, z.B. zwischen Datei und Verzeichnis
- konkurrender Zugriff, Replikate, Cache,...

Eingesetzt zur Realisierung von Leistungs- und Ausfalltransparenz

- Heterogenität

- ist in gewachsenen Informationsumgebungen eine Tatsache
- findet sich in Hard- und Software

- Dynamik, Offenheit

- Gewährleistung von “Interoperabilität” ist nicht einfach

- Komplexität

- Abstraktion als Mittel zur Beherrschung der Komplexität wichtig:
 - a) Schichten (Kapselung, virtuelle Maschinen)
 - b) Modularisierung (z.B. Services)
 - c) “Transparenz”-Prinzip

- Sicherheit

- Vertraulichkeit, Authentizität, Integrität, Verfügbarkeit,...
- notwendiger als in klassischen Einzelsystemen
- aber schwieriger zu gewährleisten (mehr Angriffspunkte)

Aspekte verteilter Systeme

im Vergleich zu *sequentiellen* Systemen:

- Grösse und Komplexität

→ jede(r) ist anders

- Heterogenität

- Nebenläufigkeit

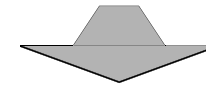
→ vieles gleichzeitig

- Nichtdeterminismus

→ morgen anders als heute

- Zustandsverteilung

→ niemand weiss alles



- Programmierung *komplexer*

- Test und Verifikation *aufwändiger*

- Verständnis der Phänomene *schwieriger*

⇒ gute Werkzeuge (“Tools”) und Methoden

- z.B. Middleware als Software-Infrastruktur

⇒ adäquate Modelle, Algorithmen, Konzepte

- zur Beherrschung der “neuen” Phänomene

Ziel: Verständnis der grundlegenden Phänomene, Kenntnis der geeigneten Konzepte und Verfahren

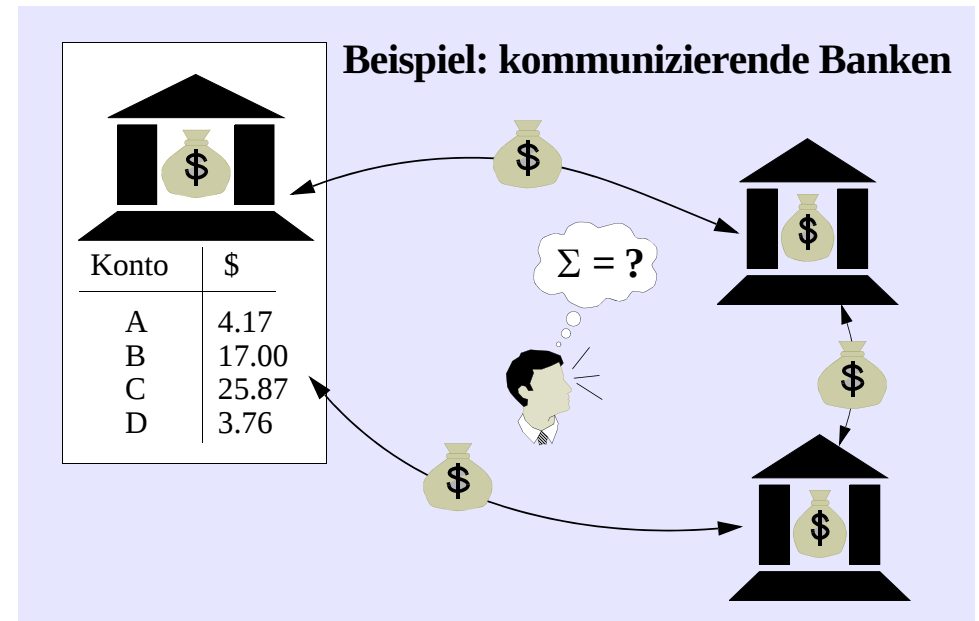
Einige *konzeptionelle* Probleme und Phänomene verteilter Systeme

- 1) Schnappschussproblem
- 2) Phantom-Deadlocks
- 3) Uhrensynchronisation
- 4) Kausaltreue Beobachtungen
- 5) Geheimnisvereinbarung über unsichere Kanäle

-
- Dies sind einige einfach zu erläuternde Probleme und Phänomene
 - Es gibt noch viel mehr und viel komplexere Probleme
 - konzeptioneller Art
 - praktischer Art
 - Achtung: Manches davon wird nicht hier, sondern in der Vorlesung “Verteilte Algorithmen” eingehender behandelt!

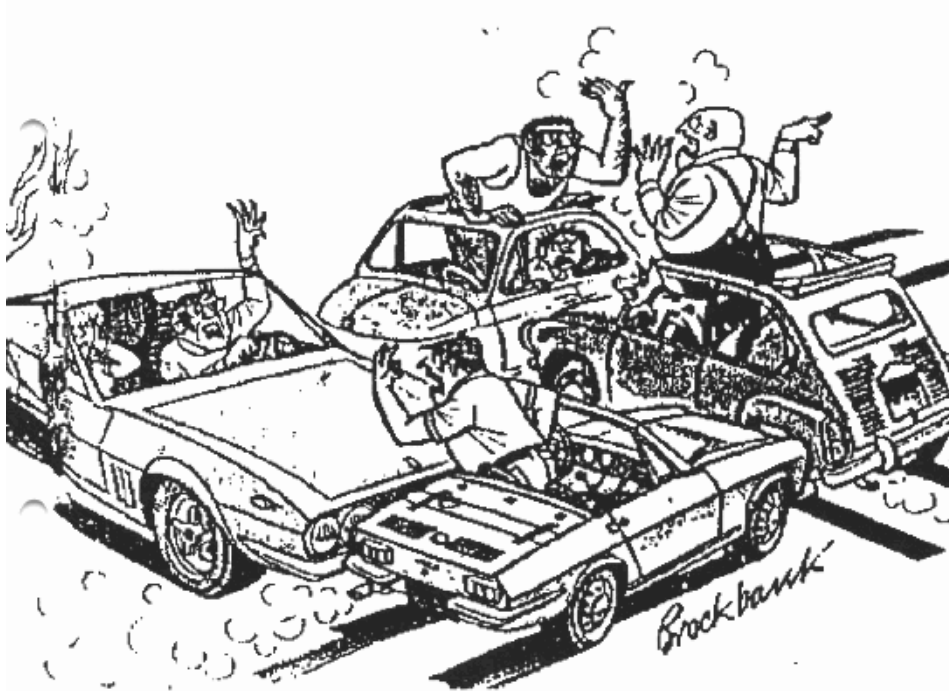
Ein erstes Beispiel: Wieviel Geld ist in Umlauf?

- **konstante** Geldmenge, oder
- **monotone** Inflation (→ Untergrenze)



- **Modellierung:**
 - verteilte Geldkonten
 - **ständige Transfers** zwischen den Konten
- **Erschwerte Bedingungen:**
 - niemand hat eine **globale Sicht**
 - es gibt keine **gemeinsame Zeit** (“Stichtag”)
- **Anwendung:** z.B. verteilte DB-Sicherungspunkte

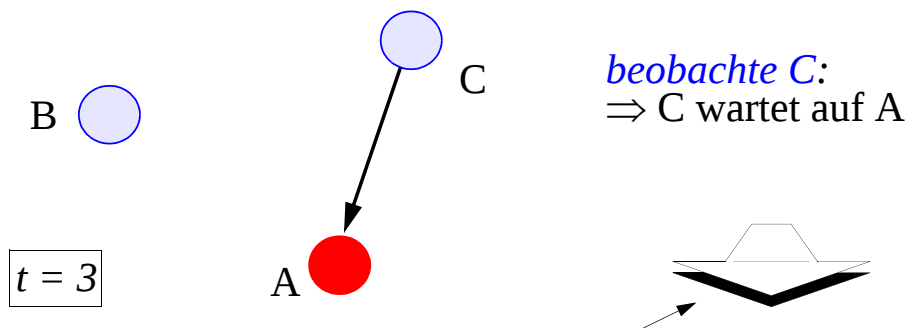
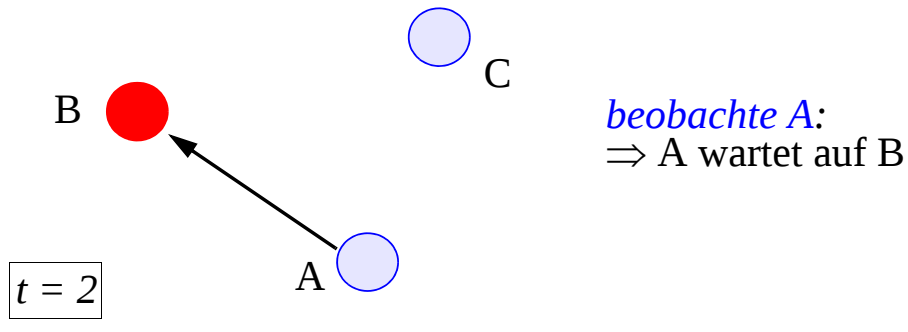
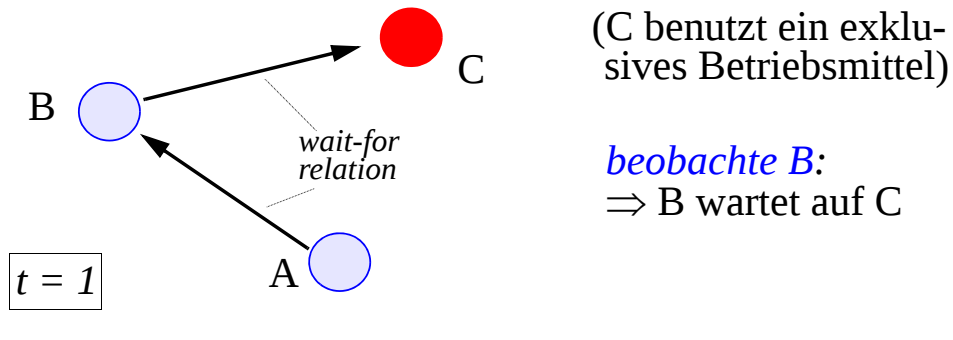
Ein zweites Beispiel: Das Deadlock-Problem



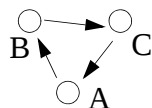
Das Deadlock-Problem



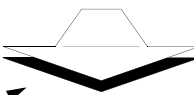
Phantom-Deadlocks



Keine exakte globale Zeit!

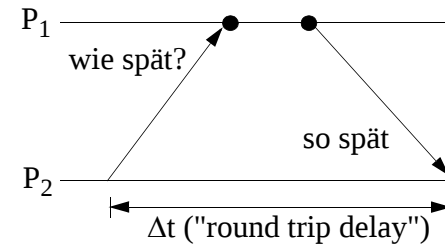


falscher Schluss!

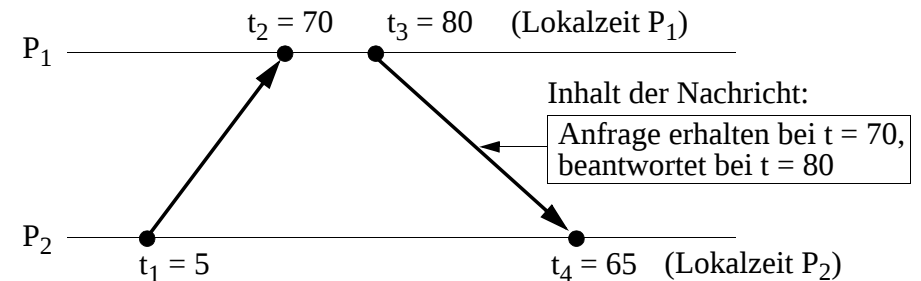


Deadlock!

Ein drittes Problem: Uhrensynchronisation



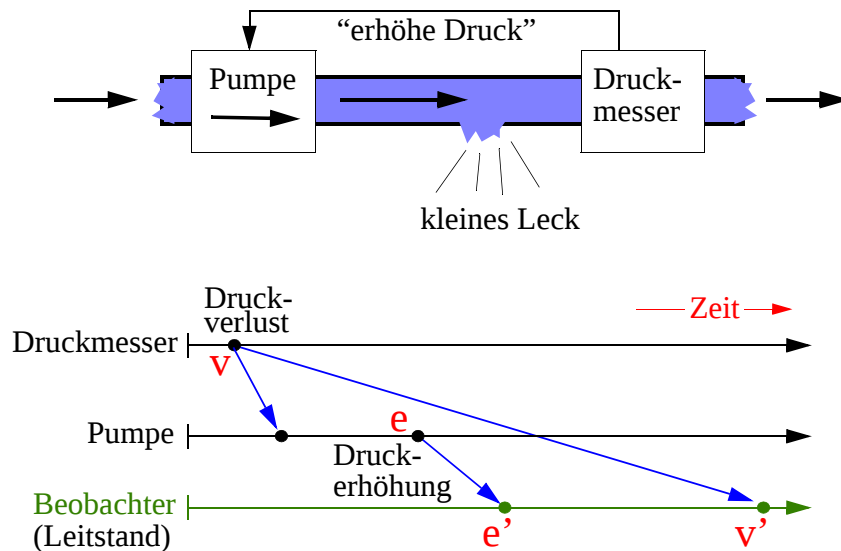
- Lastabhängige Laufzeiten von Nachrichten
- Unsymmetrische Laufzeiten
- Wie erfährt man die Laufzeit?



- Uhren gehen nicht unbedingt gleich schnell!
 (wenigstens "Beschleunigung ≈ 0 ", d.h. konstanter Drift gerechtfertigt?)
- Wie kann man den Offset der Uhren ermitteln oder zumindest approximieren?

Ein viertes Problem: (nicht) kausaltreue Beobachtungen

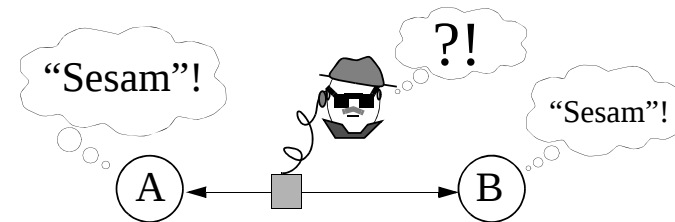
- Gewünscht: Eine **Ursache** stets vor ihrer (u.U. indirekter) **Wirkung** beobachten



Falsche Schlussfolgerung des Beobachters:

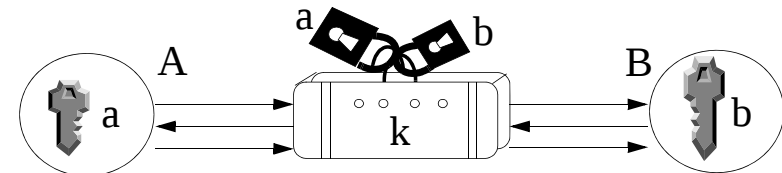
Es erhöhte sich der Druck (aufgrund einer unbegründeten Aktivität der Pumpe), es kam zu einem Leck, was durch den abfallenden Druck angezeigt wird.

Und noch ein Problem: Verteilte Geheimnisvereinbarung



- Problem: A und B wollen sich über einen unsicheren Kanal auf ein gemeinsames geheimes Passwort einigen.

- Idee: Vorhängeschlösser um eine sichere Truhe:



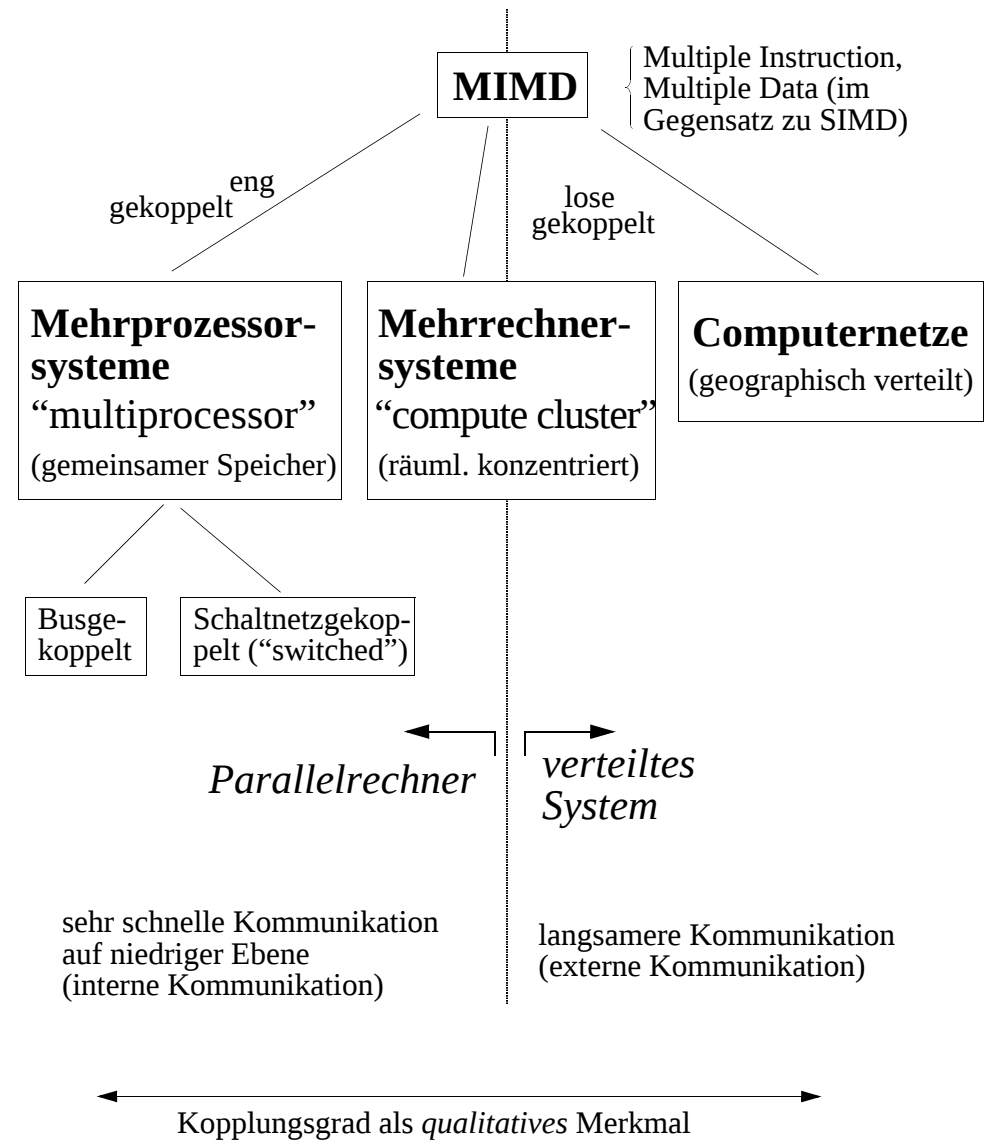
1. A denkt sich Passwort k aus und tut es in die Truhe.
2. A verschliesst die Truhe mit einem Schloss a.
3. A sendet die so verschlossene Truhe an B.
4. B umschliesst das ganze mit seinem Schloss b.
5. B sendet alles doppelt verschlossen an A zurück.
6. A entfernt Schloss a.
7. A sendet die mit b verschlossene Truhe wieder an B.
8. B entfernt sein Schloss b.

- Problem: Lässt sich das so softwaretechnisch realisieren?

Wie wäre es damit?: k sei eine Zahl. "Verschliessen" und "aufschliessen" eines Schlosses entspricht dem Hinzuaddieren oder Subtrahieren einer beliebig ausgedachten (geheimgehaltenen) Zahl a bzw. b.

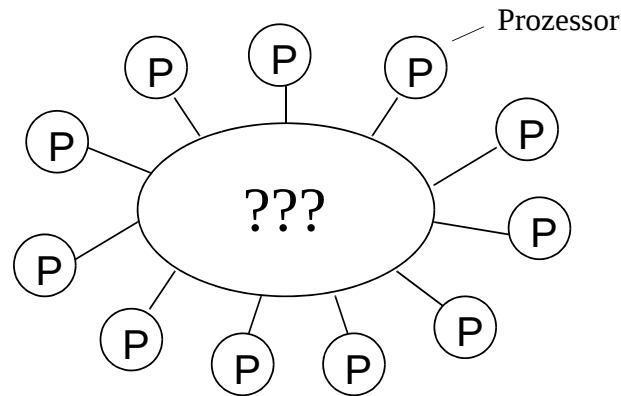
Multiprozessoren und Compute-Cluster

Abgrenzung Parallelrechner



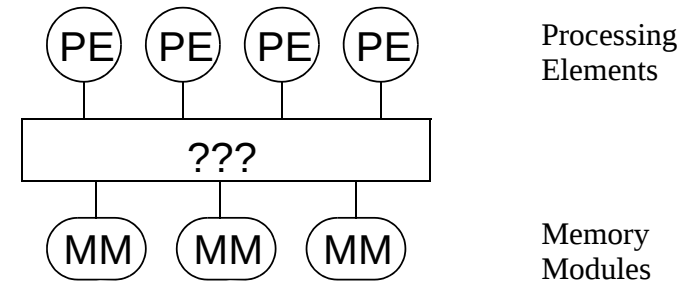
Prozessorverbund

- Autonome Prozessoren + Kommunikationsnetz
- Je nach Kopplungsgrad und Grad der Autonomie ergibt sich daraus ein
 - Mehrprozessorsystem
 - Compute Cluster
 - Computernetz



Speicherkopplung

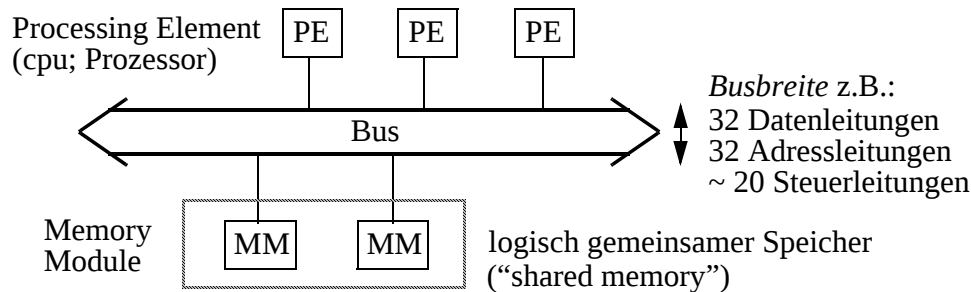
- Shared Memory
 - Kommunikation über gemeinsamen Speicher



- n Processing Elements teilen sich k Memory Modules
- Kopplung zwischen PE und MM, z.B.
 - Bus
 - Schaltnetz
 - Permutationsnetz
- UMA-Architektur (Uniform Memory Access) oder NUMA (Non-Uniform Memory Access)

wenn es "nahe" und "ferne" Speicher gibt: z.B. schneller Zugriff auf den eigenen Speicher, langsamer auf fremden

Busgekoppelte Multiprozessoren

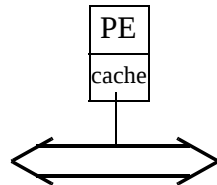


Problem:

Bus i.Allg. bereits bei wenigen (3 - 5) PEs überlastet

Lösung:

Lokale Caches
zwischen PE und Bus:



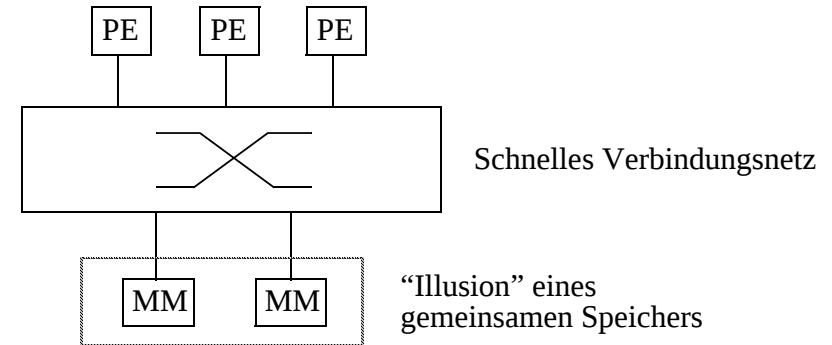
Cache gross genug (relativ zur Hauptspeichergrösse)
wählen, um Hitraten > 90% erzielen

Probleme:

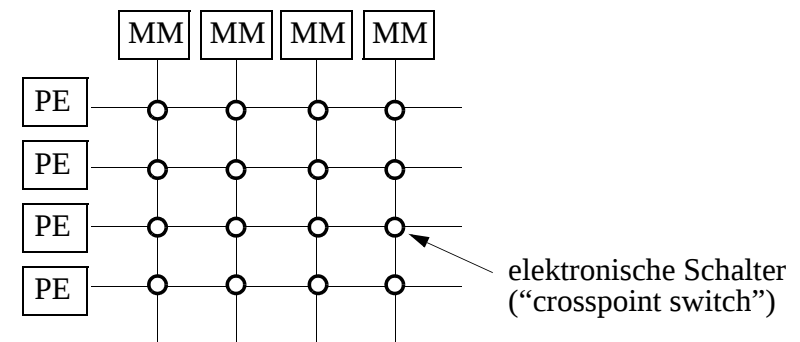
- 1) Kohärenz der caches
- 2) Damit Überlastungsproblem nur verschoben
(nicht wesentlich mehr Prozessoren möglich)

Generell: Busgekoppelte Systeme schlecht skalierbar!
(Übertragungsbandbreite bleibt "konstant" bei Erweiterung um Knoten)

Schaltnetzgekoppelte Multiprozessoren



Z.B. Crossbar-switch (Kreuzschienenverteiler):

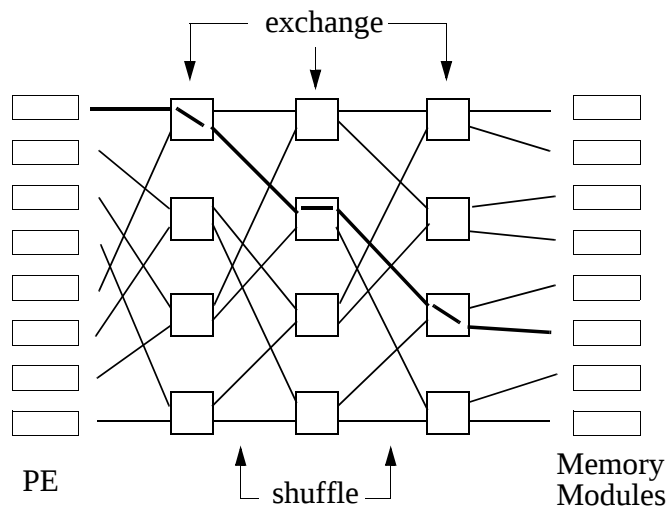
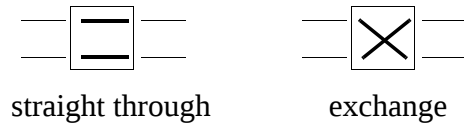


- Gleichzeitiger Zugriff von PEs auf Speichermodule (MM) zum Teil möglich
- Schlecht skalierbar (quadratisch viele Schalter)

Permutationsnetze

Mehrere Stufen von Schaltelementen ermöglichen die Verbindung jeden Einganges zu jedem Ausgang

Schaltelement ("interchange box") kann zwei Zustände annehmen (durch ein Bit ansteuerbar):



Beispiel:
Shuffle-Exchange-Netz
(Omega-Netz)

Hier: $\log n$ (identische!) Stufen mit je $n/2$ Schaltern

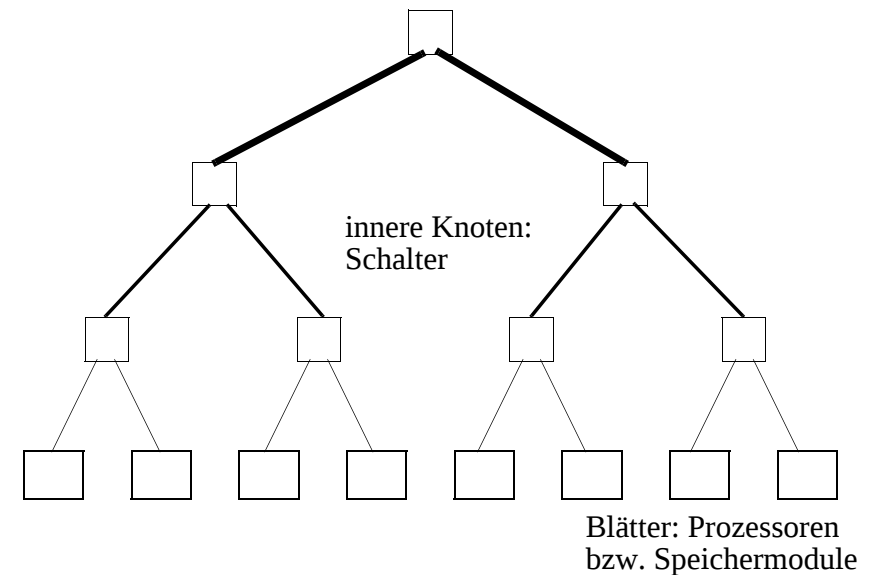
Es gibt weitere ähnliche dynamisch schaltbare Netze

Designkriterien:

- wenig Stufen ("delay")
- Parallele Zugriffe; Vermeidung von Blockaden

z.B. Butterfly-Netze

Fat-Tree-Netze



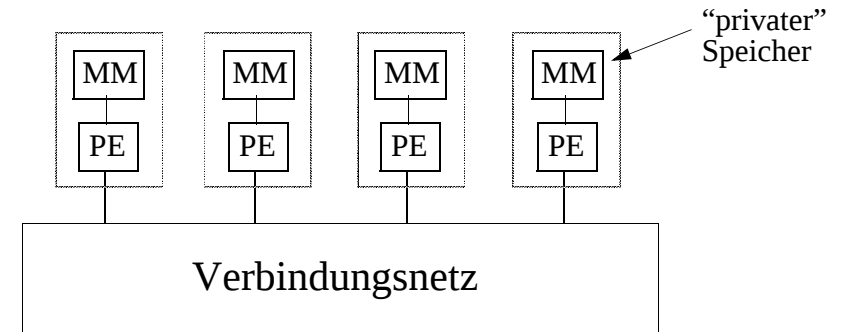
Verbindungsleitungen höherer Bandbreite bzw. mehrere parallele Leitungen auf Niveaus, die näher an der Wurzel liegen

Multiprozessoren - Fazit

- Gemeinsamer Speicher, über den die Prozessoren Information austauschen (d.h. kommunizieren) können
 - Prozessoren müssen mit dem Speicher (bzw. den einzelnen Speichermodulen) gekoppelt werden
- Speicherkopplung begrenzt Skalierbarkeit und räumliche Ausdehnung
 - Untergliederung des Speichers in mehrere Module (Parallelität)
 - leistungsfähiges Kommunikationsnetz
- Bewertungskriterien für Verbindungsnetze
 - Realisierungsaufwand (Grösse, Kosten)
 - Skalierbarkeit (mit wachsender Anzahl PEs und MMs)
 - innere Blockadefreiheit (parallele Kommunikationsvorgänge?)
 - Anzahl der Stufen (Verzögerung)
 - Eingangsgrad und Ausgangsgrad der Bauelemente

Mehrrechnersysteme ("Compute Cluster")

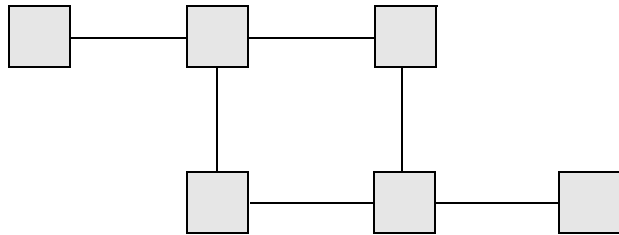
Vernetzung vollständiger Einzelrechner:



Zugriff auf andere Rechner (bzw. deren private Speicher) nur indirekt über *Nachrichten*

- kein globaler Speicher
- NORMA-Architektur (NO Remote Memory Access)

Verbindungstopologien für Mehrrechnersysteme



Zusammenhängender Graph mit

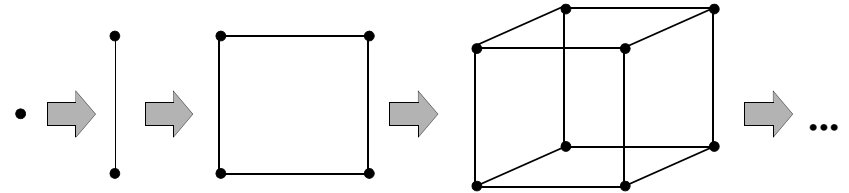
Knoten = Rechner (Prozessor mit privatem Speicher)

Kante = dedizierte Kommunikationsleitung

Ausdehnung: i.Allg. nur wenige Meter

Hypercube

- Hypercube = “Würfel der Dimension d ”



← Draufsicht von der Seite liefert jeweils niedrigere Dimension

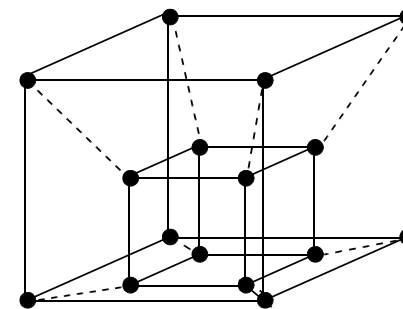
→ Entsprechend: Herausdrehen des Objektes aus der Blickebene zeigt, dass es sich “eigentlich” um ein Objekt der Dimension $n+1$ handelt!

- Rekursives Konstruktionsprinzip

- Hypercube der Dimension 0: Einzelrechner

- Hypercube der Dimension $d+1$:

„Nimm zwei Würfel der Dimension d
und verbinde korrespondierende Ecken“



4-dimensionaler
Würfel

Bewertungskriterien:

- Gesamtzahl der Verbindungen (bei n Knoten)
- maximale bzw. durchschnittliche Entfernung zweier Knoten
- Anzahl der Nachbarn eines Knotens (“fan out”)
- Skalierbarkeit
- Routingkomplexität
- Zahl der alternativ bzw. parallel verfügbaren Wege

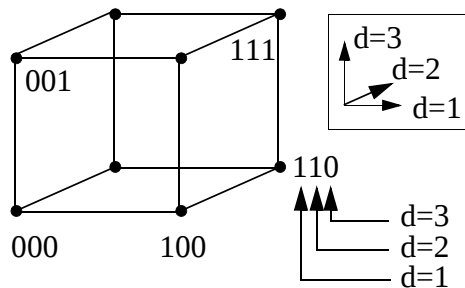
Technologische Faktoren:

- Geschwindigkeit, Durchsatz, Verzögerung, spezifische Kommunikationsprozessoren / Switches,...

Frage: Wieso kommuniziert man nicht einfach über Funk (indem z.B. jeder Knoten seine spezifische Empfangsfrequenz hat)?

Hypercube der Dimension d

- $n = 2^d$ Knoten
- Anzahl der Nachbarn eines Knotens = d
(Anzahl der “ports” in der Hardware)
- Gesamtzahl der Kanten (= Verbindungen): $d \cdot 2^{d-1}$
(Ordnung $O(n \log n)$)
- Einfaches Routing:
 - Knoten systematisch (entspr. rekursivem Aufbau) numerieren
 - Zieladresse bitweise xor mit Absenderadresse
 - Wo sich eine “1” findet, in diese Dimension muss gewechselt werden



- Maximale Weglänge: d
- Durchschnittliche Weglänge = $d/2$
(Induktionsbeweis als Übung!)

-Vorteile Hypercube:

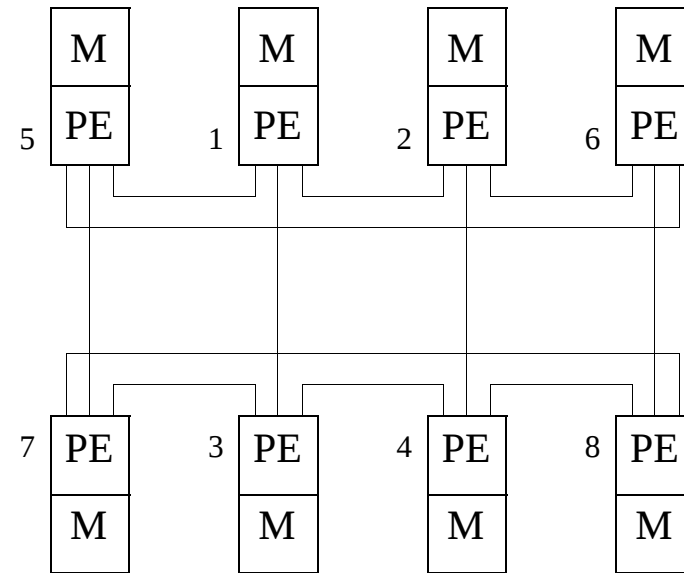
- kurze Weglängen (max. $\log n$)
- einfaches Routing
- viele Wegalternativen (Fehlertoleranz, Parallelität!)

-Nachteile:

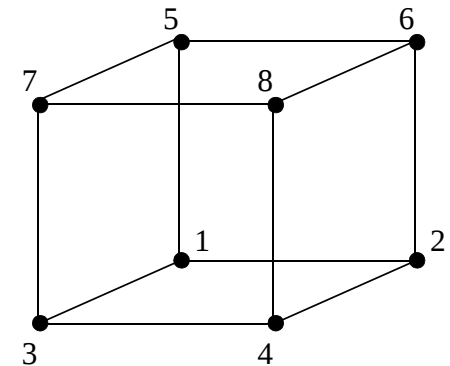
- Anzahl der Nachbarn eines Knotens wächst mit der Dimension d
- insgesamt relativ viele Verbindungen (“Kanten”): $O(n \log n)$
(eigentlich genügen $n-1$)

wieviele verschiedene Wege der Länge k gibt es insgesamt?

Layout eines Hypercube

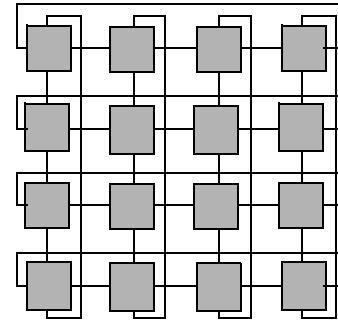
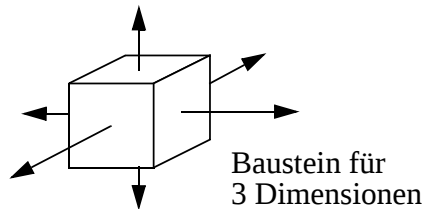
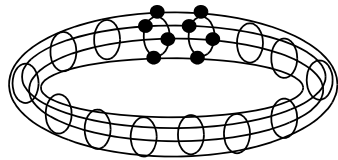


Obiger Topologie sieht man nicht unmittelbar an, dass es sich dabei um einen 3-dimensionalen Würfel handelt!



Eine andere Verbindungstopologie: der d-dimensionale Torus

= d-dimensionales Gitter mit “wrap-around”



- Rekursives Konstruktionsprinzip: „Nimm w_d gleiche Exemplare eines Torus der Dimension $d-1$ und verbinde korrespondierende Elemente zu einem Ring“

- Bei Ausdehnung w_i in Dimension i :

$$n = w_1 \times w_2 \times \dots \times w_d \text{ Knoten;}$$

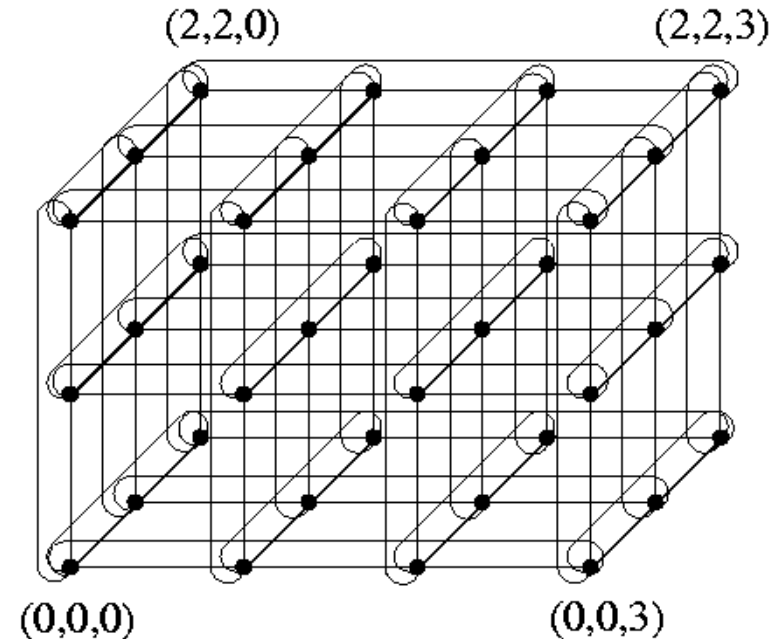
$$\text{mittlere Entfernung zw. 2 Knoten: } \Delta \approx \frac{1}{4} \sum w_i$$

- *Ring* als Sonderfall $d = 1$!

- *Hypercube* der Dimension d ist d-dimensionaler Torus mit $w_i = 2$ für alle Dimensionen!

$$\rightarrow \Delta = \frac{1}{4} \sum_d 2 = \frac{1}{4} (2d) = \frac{d}{2} = \frac{1}{2} \log_2 n$$

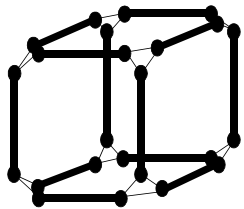
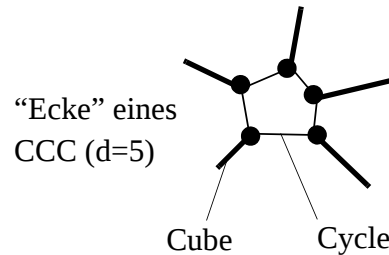
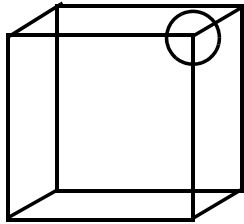
3-dimensionaler Torus



Mit $w_1 = 4$, $w_2 = 3$, $w_3 = 3$

Cube Connected Cycles (CCC)

CCC = d-dimensionaler Hypercube mit aufgeschnittenen Ecken, die durch Gruppen von d ringförmig verbundenen Knoten ersetzt sind



Beachte: Jeder Knoten hat immer *drei* Anschlüsse!

Bei Dimension d: $n = d \cdot 2^d$

Maximale / mittlerer Weglänge? ←

Denkübung!
(nicht ganz
einfach)

Anzahl der Verbindungen = $3n / 2$
(statt $O(n \log n)$ wie beim Hypercube)

Es gibt viele weitere Verbindungstopologien
(wollen wir hier aber nicht betrachten)

Kommunikation

Kommunikation

- Prozesse sollen kooperieren, daher untereinander Information austauschen können; mittels
 - *gemeinsamer Daten* in einem globalen Speicher (dieser kann physisch oder evtl. nur logisch vorhanden sein: “virtual shared memory”)
 - oder *Nachrichten*: Daten an eine entfernte Stelle kopieren

Notwendig, damit die Kommunikation klappt, ist jedenfalls:

- 1) dazwischenliegendes *physikalisches Medium*
 - z.B. elektrische Signale in Kupferkabeln
- 2) einheitliche *Verhaltensregeln*
 - Kommunikationsprotokolle
- 3) gemeinsame *Sprache* und gemeinsame *Semantik*
 - gleiches Verständnis der Bedeutung von Kommunikationskonstrukten und -Regeln

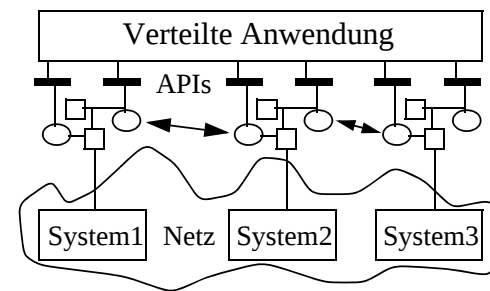
Also trotz Verteiltheit gewisse gemeinsame Aspekte!

- Nachrichtenbasierte Kommunikation:
 - send → receive
 - Implizite Synchronisation: Senden *vor* Empfangen
 - Empfänger erfährt, wie weit der Sender mindestens ist
 - Nachrichten sind dynamische Betriebsmittel
 - verursachen Aufwand und müssen verwaltet werden

Message Passing System

als einfache Form von “Middleware”

- Organisiert den Nachrichtentransport
- Bietet Kommunikationsprimitive (als APIs) an
 - z.B. *send (...)* bzw. *receive (...)*
 - evtl. auch ganze Bibliothek unterschiedlicher Kommunikationsdienste
 - verwendbar mit gängigen Programmiersprachen (oft zumindest C)

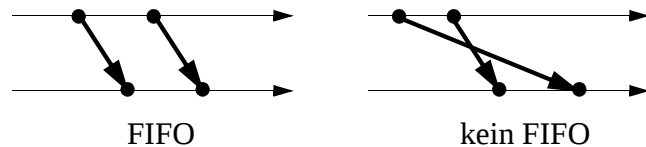


- Besteht aus Hilfsprozessen, Pufferobjekten,...
- Verbirgt Details des zugrundeliegenden Netzes

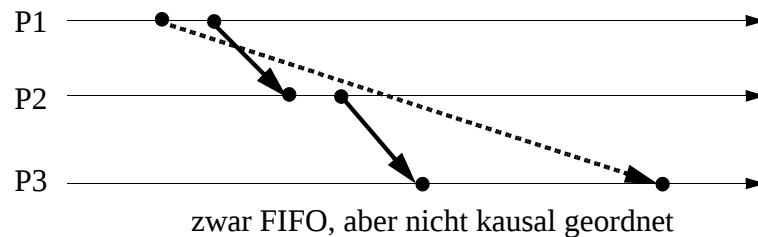
- Verwendet vorhandene Netzprotokolle und implementiert damit neue, “höhere” Protokolle
- Garantiert (je nach “Semantik”) gewisse Eigenschaften
 - z.B. Reihenfolgeerhalt oder Prioritäten von Nachrichten
- Abstrahiert von Implementierungsaspekten
 - wie z.B. Geräteadressen, Längenbeschränkung von Nachrichten etc.
- Maskiert gewisse Fehler
 - mit typischen Techniken zur Erhöhung des Zuverlässigkeitsgrades: Timeouts, Quittungen, Sequenznummern, Wiederholungen, Prüfsummen, fehlerkorrigierende Codes,...
- Verbirgt Heterogenität unterschiedlicher Rechner- bzw. Betriebssystemplattformen
 - erleichtert Portabilität von Anwendungen

Ordnungserhalt von Nachrichten?

- Manchmal werden vom Kommunikationssystem Garantien bzgl. Nachrichtenreihenfolgen gegeben
- Eine mögliche Garantie stellt FIFO (First-In-First-Out) dar: Nachrichten zwischen zwei Prozessen überholen sich nicht: Empfangsreihenfolge = Sendereihenfolge



- FIFO verbietet allerdings nicht, dass Nachrichten evtl. indirekt (über eine Kette anderer Nachrichten) überholt werden



- Möchte man auch dies haben, so muss die Kommunikation “kausal geordnet” sein (Anwendungszweck?)
 - “Dreiecksungleichung”: Keine Information erreicht Empfänger auf Umwegen schneller als auf direktem Wege
 - entspricht einer “Globalisierung” von FIFO auf mehrere Prozesse
 - Denkübung: wie garantiert (d.h. implementiert) man kausale Ordnung auf einem System ohne Ordnungsgarantie?