Folienkopien zur Vorlesung

Verteilte Systeme

Friedemann Mattern

Departement Informatik, ETH Zürich

Wintersemester 2004/5 (5-stündig inkl. Übungen)

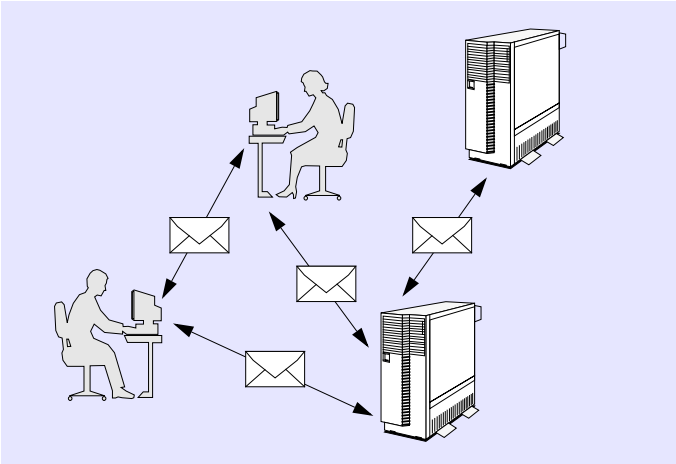
Achtung: Prüfungsrelevant ist der Inhalt der Vorlesung, nicht alleine der Text dieser Foliensammlung!

ETH Eidgenössische Technische Hochschule Zürich

Verteilte Systeme

Friedemann Mattern

Institut für Pervasive Computing
Departement Informatik
ETH Zürich



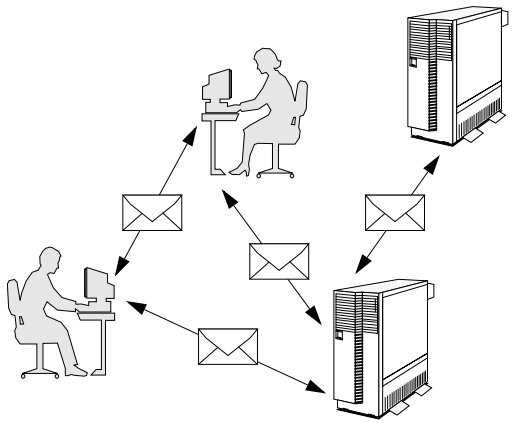
- Rechner, Personen, Prozesse, “Agenten” sind an *verschiedenen Orten*.
- Autonome Handlungsträger, die jedoch gelegentlich *kooperieren* (und dazu über Nachrichten *kommunizieren*).

V. Rencin

Wer bin ich? Wer sind wir?

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

Seit Herbst 1999 an der ETH Zürich



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

- Assistent(inn)en:
  - Jürgen Bohn
  - Vlad Coroama
  - Svetlana Domnitcheva
  - Christian Flörkemeier
  - Christian Frank
  - Oliver Kasten
  - Matthias Lampe
  - Marc Langheinrich
  - Matthias Ringwald
  - Kay Römer
  - Christof Roduner
  - Michael Rohs
  - Silvia Santini
  - Frank Siegemund
  - Harald Vogt

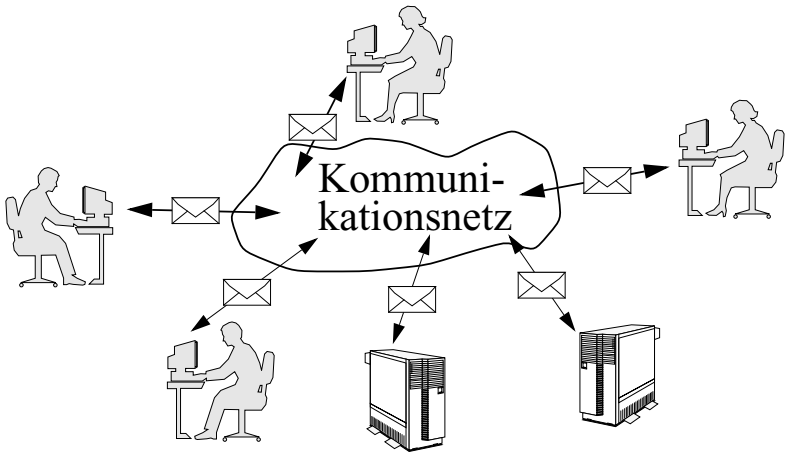
Mehr zu uns:
www.vs.inf.ethz.ch

Ansprechpersonen für organisatorische Aspekte (z.B. Übungsbetrieb)

“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?

Organisatorisches zur Vorlesung

5-stündige Veranstaltung (Vorlesung inkl. Übungen)

Sinnvolle Vorkenntnisse:

- Betriebssysteme (Prozessbegriff, Synchronisation)...
- UNIX / C / Java
- Grundkenntnisse der Informatik und Mathematik (Vordiplom)

	Mo	Di	Mi	Do	Fr
08.00	**Vert Sys**	IFW A36		IFW A36	**Vert Sys**
10.00					
11.30					
13.30					
15.00					
17.00					

Mo 08:15 - 11:00, IFW A36
Fr 08:15 - 10:00, IFW A36
} *Vorlesung inkl. Übung*

Absicht!

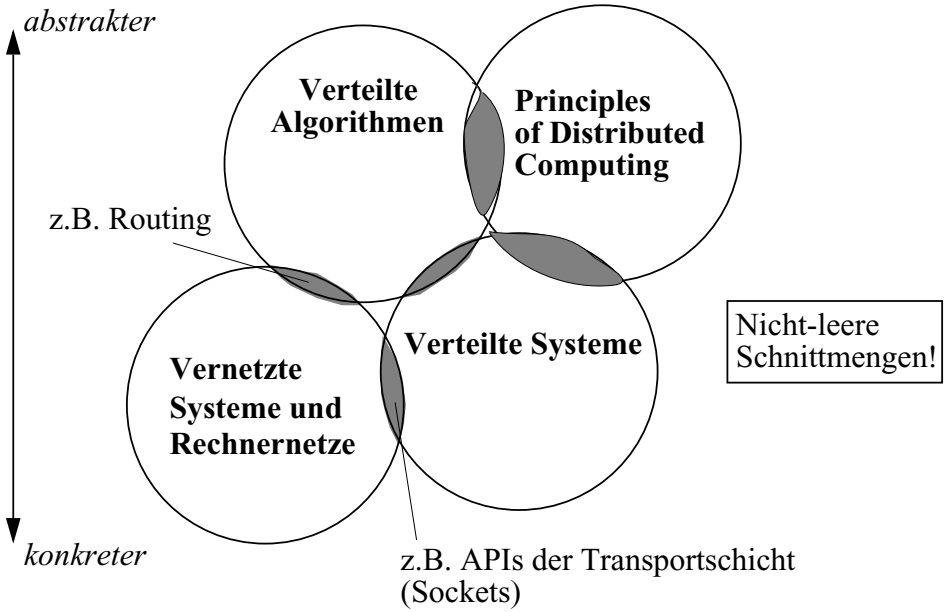
- Gelegentliche *Denkaufgaben* in der Vorlesung
- Praktische Übungen korrelieren gelegentlich nur schwach mit dem Inhalt der Vorlesung *(komplementieren* die Vorlesung)
- Gelegentliche *Übungsstunden* (zu den “Vorlesungsterminen)” zur Besprechung der Aufgaben und Vertiefung des Stoffes

---

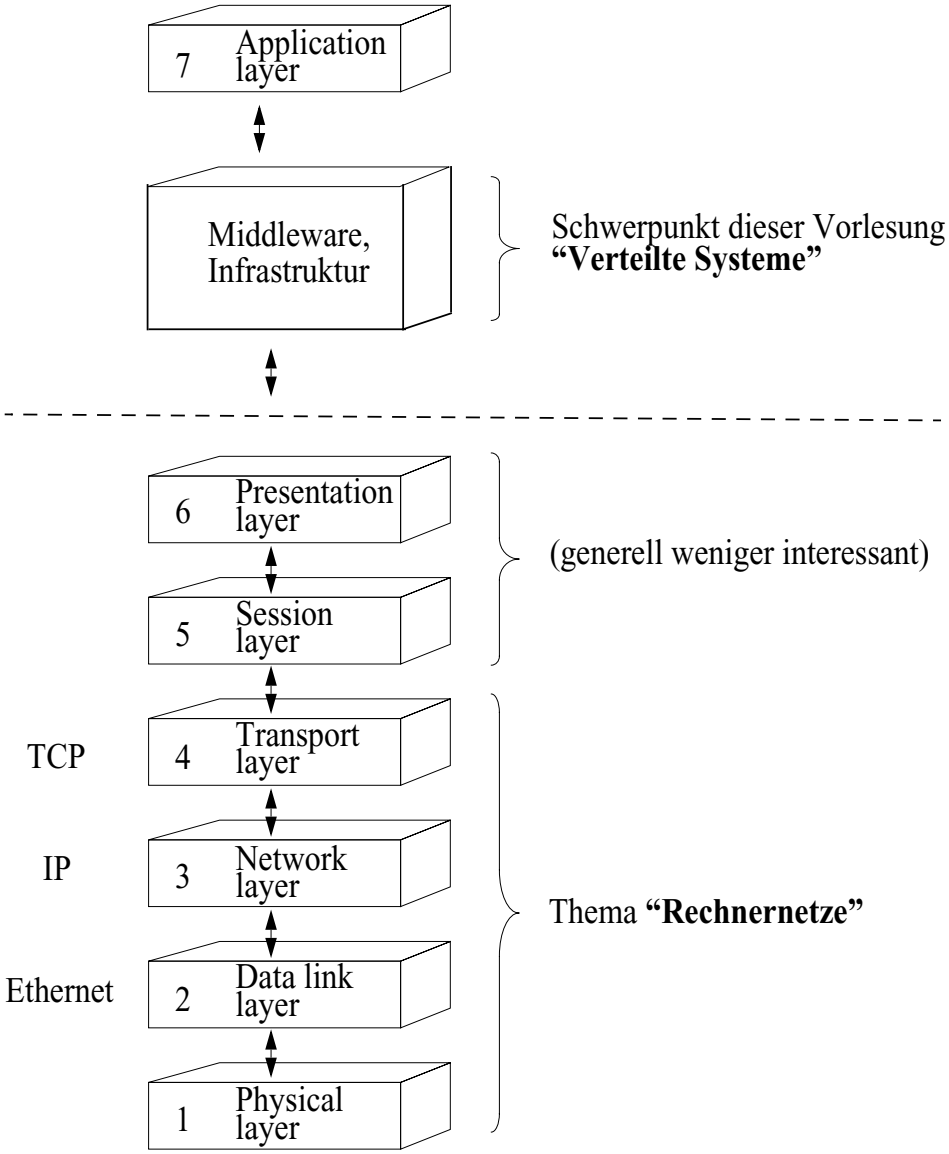
- *Folienkopien* jeweils einige Tage nach der Vorlesung im Web im .pdf-Format: www.vs.inf.ethz.ch/edu

Thematisch verwandte Veranstaltungen im Master-/Fachstudium

- *Principles of Distributed Computing*
- *Ubiquitous Computing*
- *Mobile Computing*
- Einschlägige *Seminare*
- *Semester- und Diplomarbeit / Masterarbeit*
- *Praktikum ("Labor")*
- ...



Netze, Anwendungen, Verteilte Systeme



Literatur

! G. Coulouris, J. Dollimore, T. Kindberg: *Distributed Systems: Concepts and Design (3rd ed.).* Addison-Wesley, 2001

! A. Tanenbaum, M. van Steen: Distributed Systems: Principles and Paradigm, Prentice-Hall, 2001

M. Weber: *Verteilte Systeme*. Spektrum Hochschultaschenbuch, 1998

S. Mullender (Hg.): *Distributed Systems (2nd ed).* ACM Press and Addison-Wesley, 1994

R.G. Herrtwich, G. Hommel: *Nebenläufige Programme.* Springer-Verlag, 1994

K. Geihs: *Client/Server-Systeme*. Internat. Thomson Publ., 1995

A. Schill: *DCE - Das OSF Distributed Computing Environment*. Springer-Verlag, 1993

W.K. Edwards: *Core Jini.* Prentice Hall, 1999

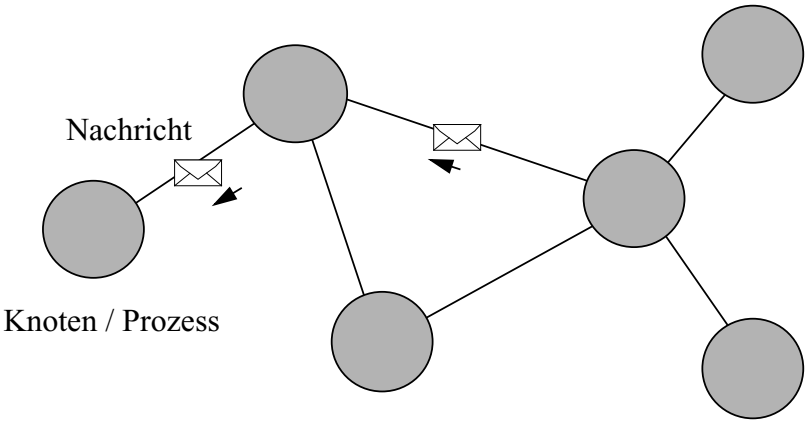
A. Tanenbaum: *Computer Networks (3rd ed.)*. Prentice-Hall, 1996

D. Comer: *Internetworking with TCP/IP, Volume III, Client-Server Programming.* Prentice-Hall, 1993

B. Schneier: *Applied Cryptography (2nd ed.).* Wiley, 1996

In den aufgeführten Büchern findet man weitere Literaturangaben.

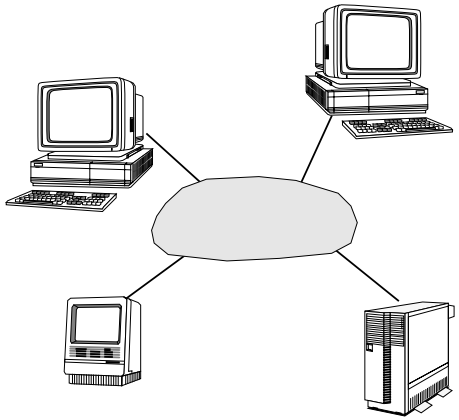
"Verteiltes System"



Physisch verteiltes System:
Mehrrechnersystem ... Rechnernetze

Logisch verteiltes System: Prozesse (Objekte, Agenten)
- Verteilung des Zustandes (keine globale Sicht)
- Keine gemeinsame Zeit (globale, genaue "Uhr")

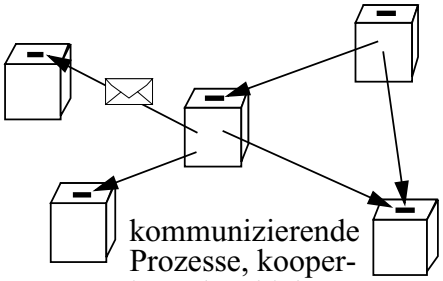
Sichten verteilter Systeme



Rechnernetz mit Rechenknoten:

- Multicomputer (Parallelrechner)
- LAN = Local Area Network
- WAN = Wide Area Network
- Routing, Adressierung....

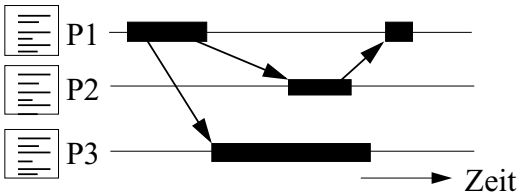
zunehmende Abstraktion



kommunizierende Prozesse, kooperierende Objekte

Objekte eines *Betriebssystems* bzw. einer *Programmiersprache*

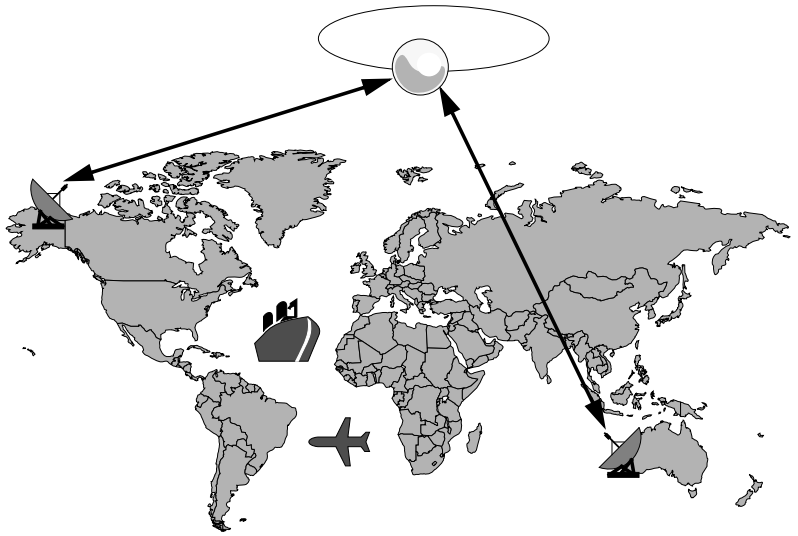
==> “Programmierersicht” (Client, Server...)



Algorithmen- und Protokoll-ebene

- Aktionen, Ereignisfolgen
- Konsistenz, Korrektheit

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

- “Inkonsistente Zustände”: Kriegsende zwischen England und Frankreich war in den Kolonialgebieten erst später bekannt!
- Heute: “zeitkompakter Globus” - weitgehend synchronisierte Uhren.

Motivation (1)

- Was sind sinnvolle verteilte Anwendungen?
- Worin besteht der letztendliche Nutzen?

Ein Szenario:

Die weltweit verteilte virtuelle Bibliothek

Notwendig:

- schnellere Kommunikationsnetze
- bessere Geräte (sehr hohe Auflösung, flach, portabel...)
- elektronische Speicherung (fast) aller Bücher, Dokumente...
- Software-Infrastruktur, Standards...
- soziale, politische, ökonomische “Akzeptanz”

schwierig

am schwierigsten

Vorteile:

- schnelle Verfügbarkeit, neueste Version
- Kostensenkung
- Suchfunktionen, elektronische Auswertbarkeit
- Querbezüge durch explizite oder implizite Referenzen (--> Hyperlinks)
- Integration verschiedener Medien: Text, Sprache, Bilder, Video, Animationen... --> Multimedia

Konsequenzen?

(sozial, rechtlich, kulturell, psychisch, ökonomisch...)

Einige Bemerkungen dazu

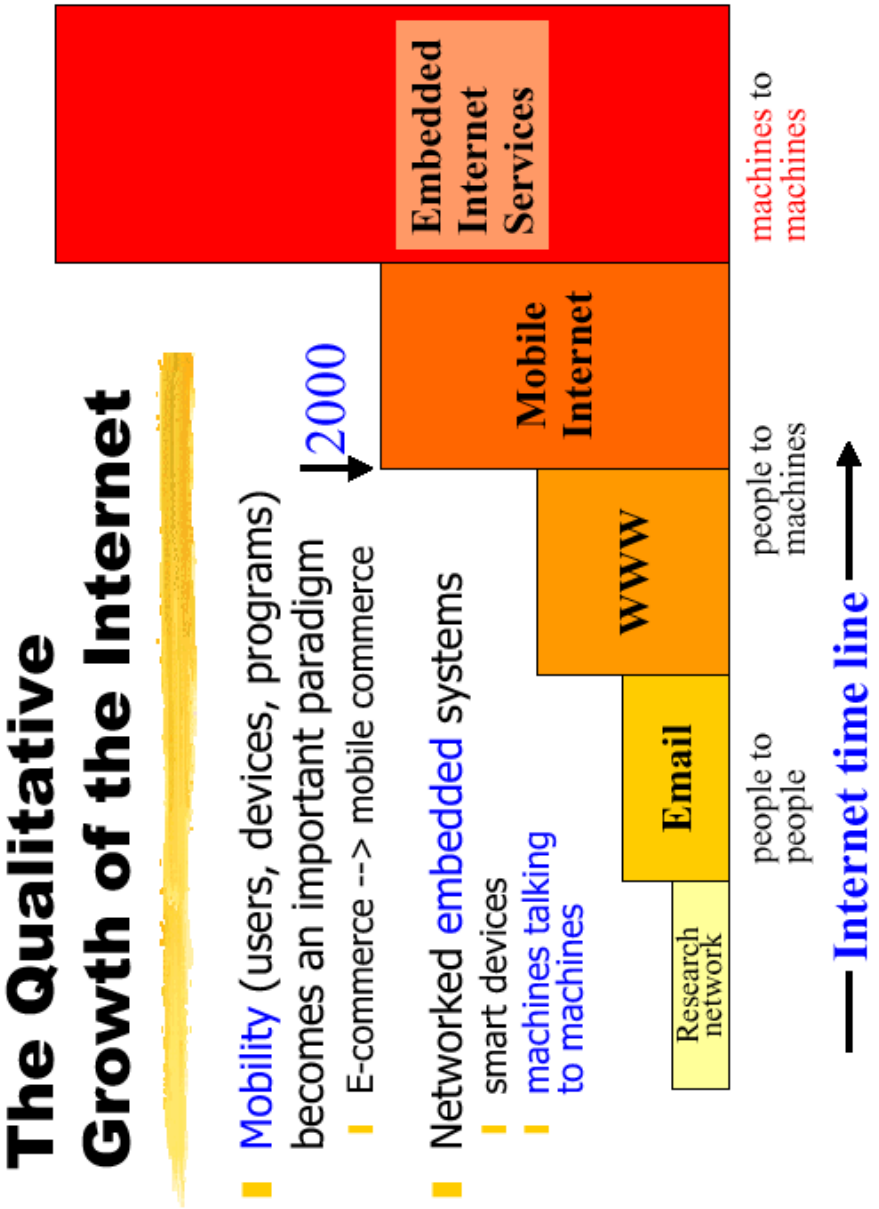
- Abrechnung fälschungssicher
- Copyright garantieren
- Authentizität garantieren
- Vertraulichkeit gewährleisten (was liest Mr. X?)
- Ausfallsicherheit
- Suchsystem (Indizes, Metadaten, Suchmaschinen...)
- Ortstransparenz
- Heterogenität (Geräte, Standards,...)
- Effizienz (z.B. ggf. parallele Suche im Netz)
- Dezentrale Organisation
- ...

- Architekturaspekte
  - Systemarchitektur soll u.a. obige Aspekte berücksichtigen
  - Schnittstellen, Teilsysteme
  - “offen” zu existierenden Diensten, anderen Systemen
  - viele Entwurfsentscheidungen (z.B.: data shipping vs. function shipping bei Suchfunktionen)

Motivation (2): Bessere Nutzung global verfügbarer Ressourcen

- WWW-Zugriff auf weltweite Wissensbestände
- Nutzung von Hochleistungsrechnern oder Spezialsystemen
  - Supercomputer
  - Software-Bibliotheken
- „Aufsammeln“ ungenutzter Rechenleistung im Internet
  - Faktorisierung grosser Zahlen in verteilter Weise
  - Brechen einer DES-Verschlüsselung
- Nutzung der Zeitverschiebung zwischen Kontinenten
  - billige „Nachtrechenzeit“ für Anwender mit normaler Tagesarbeitszeit
  - Hotline rund um die Uhr
- Bearbeitung eines Problems an mehreren Standorten
  - internationaler Konzern mit nationalen Niederlassungen
  - internationale Spezialistenteams (Medizin, Forschung)

===> ökonomischer Effekt
+ ermöglicht “Globalisierung” } auf der Basis der Vernetzung (“Internet”)



Motivation (3): Middleware für das Internet

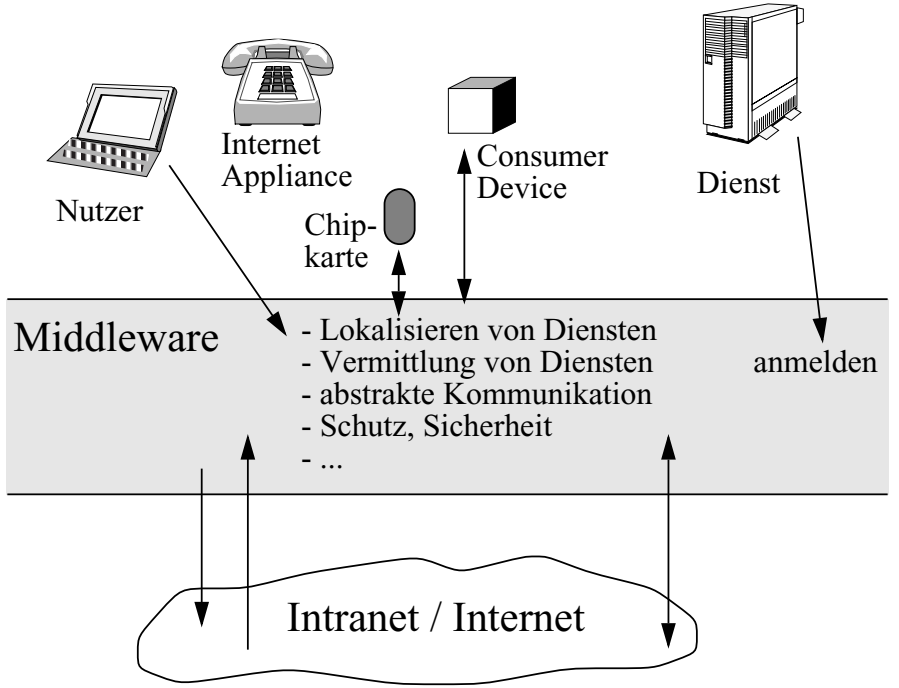
- Phänomen: das Internet verbreitet sich immer weiter
  - mehr Nutzer, Popularisierung
  - bis in die Haushalte
  - immer exotischere Endgeräte (PDA, Handy, Kühlschrank, Chipkarte)
  - bald enthalten vielleicht auch Briefmarken, Kleidungsstücke etc. kommunikationsfähige Chips
- Mobile Geräte, dynamische 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?
  - wichtig auch für Electronic Commerce-Szenarien

Middleware für das Internet

- Plattform / Infrastruktur für verteilte Anwendungen



- Beispiel: *Jini*
  - Zweck: Interaktion mit dem Netz und mit Diensten vereinfachen
  - Lookup-Service (“Bulletin-Board”)
  - “Leasing” von Objekten (Freigabe bei Ablauf des Vertrages)
  - hot plugging von Objekten, Teildiensten etc.
  - Garbage-Collection von Objekten im Netz
  - Vermittlung von Ereignissen (events); API für events
  - Unterstützung mobiler Geräte und Dienste
  - Föderation kooperierender Java-VMs (Gruppenkonzept)
  - Mobiler Code (Java-Bytecode, Applet); z.B. Druckertreiber als “Proxy”
  - Kommunikation über entfernter Methodenaufruf oder (persistente) Tupel-Räume

Transparenz

Transparenz = unsichtbar (“durchsichtig”) sein

Verteiltheit wird vor dem Benutzer / Anwendungsprogrammierer verborgen, so dass das System als Ganzes gesehen wird (statt als Menge von Einzelkomponenten)

--> Umgang mit der Verteiltheit wird einfacher
--> Abstraktion von “internen” Aspekten

Verschiedene Arten der Transparenz, z.B.:

Ortstransparenz

Ort, an dem sich Daten befinden oder an dem ein Programm ausgeführt wird, ist unsichtbar

Replikationstransparenz

Unsichtbar, wieviele Replikate eines Objektes (z.B. Datei) existieren

Concurrency-Transparenz

Mehrere Benutzer / Prozesse können gemeinsame Objekte (z.B. Dateien) benutzen, ohne dass es zu Inkonsistenzen kommt

Leistungstransparenz

Kein (spürbarer) Leistungsunterschied zwischen lokaler und entfernter Bearbeitung

Ausfalltransparenz

Ausfall einzelner Komponenten ist unsichtbar --> Fehlertoleranz

Transparenz (2)

Aufwand zur Realisierung von Transparenz ist hoch!

Implementierung von Transparenz auf verschiedenen Ebenen möglich, z.B.::

- Betriebssystem (--> alle Anwendungen profitieren davon)
- Anwendungsprogramm (Nutzung der Semantik)

Transparenz ist *graduelle Eigenschaft*

- oft nicht nur einfach “vorhanden” / “nicht vorhanden”

Transparenz lässt sich nicht immer (einfach) erreichen

- Beispiel: Fehlertransparenz, Leistungstransparenz
- Sollte daher nicht in jedem Fall perfekt angestrebt werden (vgl. Jini)

Verteilte Systeme als “Verbunde”

- *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*
  - i.a. nur Ausfall von Teilfunktionalität
  - Redundanz durch Replikation

Weitere Gründe für verteilte Systeme

- *Wirtschaftlichkeit*: Vernetzte PCs haben i.a. besseres Preis-Leistungsverhältnis als Supercomputer
  --> Anwendung falls möglich “verteilen” auf mehrere kleine Rechner
- *Geschwindigkeit*: Falls Anwendung “gut” parallelisierbar, ist eine sonst unerreichbare Leistung möglich
  - ggf. (dynamische) Lastverteilung beachten
- Vgl. auch “Grid”

- Es gibt *inhärent geographisch verteilte* Systeme
  --> z.B. Zweigstellennetz einer Bank; Steuerung einer Fabrik
  --> verteilte Informationsdienste (vgl. WWW)
- *Electronic commerce*
  --> kooperative Datenverarbeitung räumlich getrennter Institutionen
  --> z.B. Reisebüros, Kreditkarten,...
- Mensch-Mensch-*Kommunikation*
  - Plattformen für Gruppenarbeit (z.B. Web-basiert)
  - E-mail, Diskussionsforen, IRC, Sprachdienste...

Historische Entwicklung (“Systeme”)

Rechner-zu-Rechner-Kommunikation

- Zugriff auf entfernte Daten (DFÜ)
- Dezentrale Informationsverarbeitung zunächst ökonomisch nicht sinnvoll (zu teuer, Fachpersonal nötig)
--> Master-Slave-Beziehung (RJE, 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...)
- Heute Standard bei PC-Anwendungen im Betriebssystem:
  - Kommunikation über LAN (Resource-Sharing)
  - Software für “Gruppenarbeit” (email, gem. Dateisystem...)

Forschungsprojekte

- z.B. X-Server, Kerberos,...

Kommerzielle Projekte

- z.B. Reservierungssysteme, Banken, Kreditkarten

WWW (und Internet) als Plattform

- für electronic commerce etc.
- XML, web services, peer to peer,...

“Historie” der Konzepte

- Concurrency, Synchronisation...
  - bereits klassisches Thema bei Datenbanken und Betriebssystemen
- Programmiersprachen
  - kommunizierende Objekte
- Physische Parallelität
  - z.B. Multiprozessoren
- Parallele und verteilte Algorithmen
- Semantik
  - math. Modelle, 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!

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 / globale Zeit
  - Inkonsistenzen, z.B. zwischen Datei und Verzeichnis
  - Konkurrenter Zugriff, Replikate, Cache,...

Eingesetzt zur Realisierung von Leistungs- und Ausfalltoleranz

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

- Dynamik, Offenheit
  - “Interoperabilität” zu gewährleisten ist nicht einfach

- Komplexität
  - Verteilte Systeme schwierig zu entwickeln, betreiben, beherrschen
  - Abstraktion als Mittel zur Beherrschung von Komplexität wichtig:
    a) Schichten (Kapselung, virtuelle Maschinen...)
    b) Modularisierung (Services, Mikrokerne...)
    c) “Transparenz”-Prinzip

- Sicherheit
  - Vertraulichkeit, Authenzitität, Integrität, Verfügbarkeit...
  - notwendiger als in klassischen Einzelsystemen
  - aber schwieriger zu gewährleisten (mehr Schwachstellen)

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 *aufwendiger*
- 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

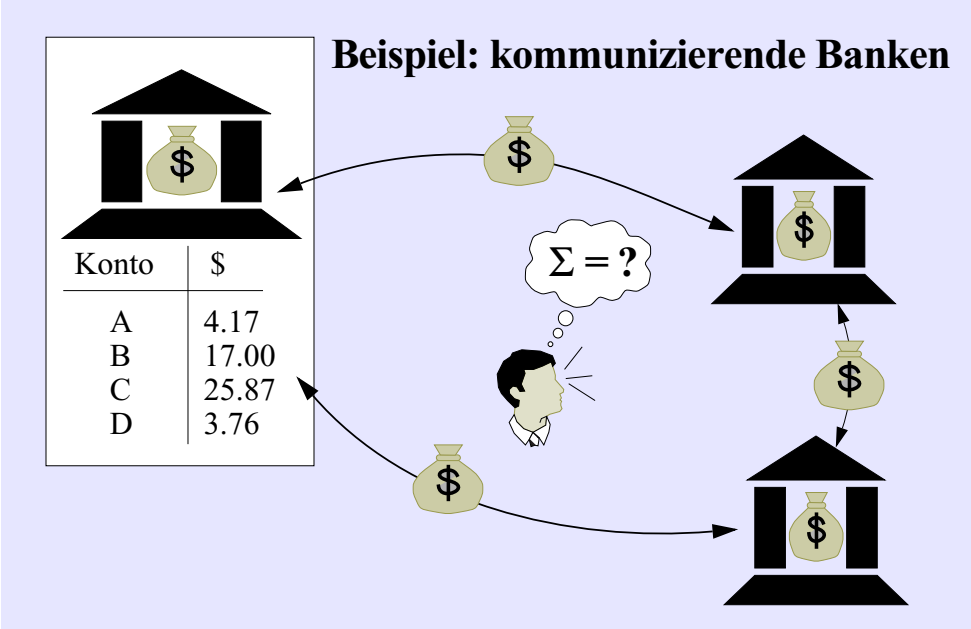
Einige *konzeptionelle* Probleme und Phänomene verteilter Systeme

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" behandelt!

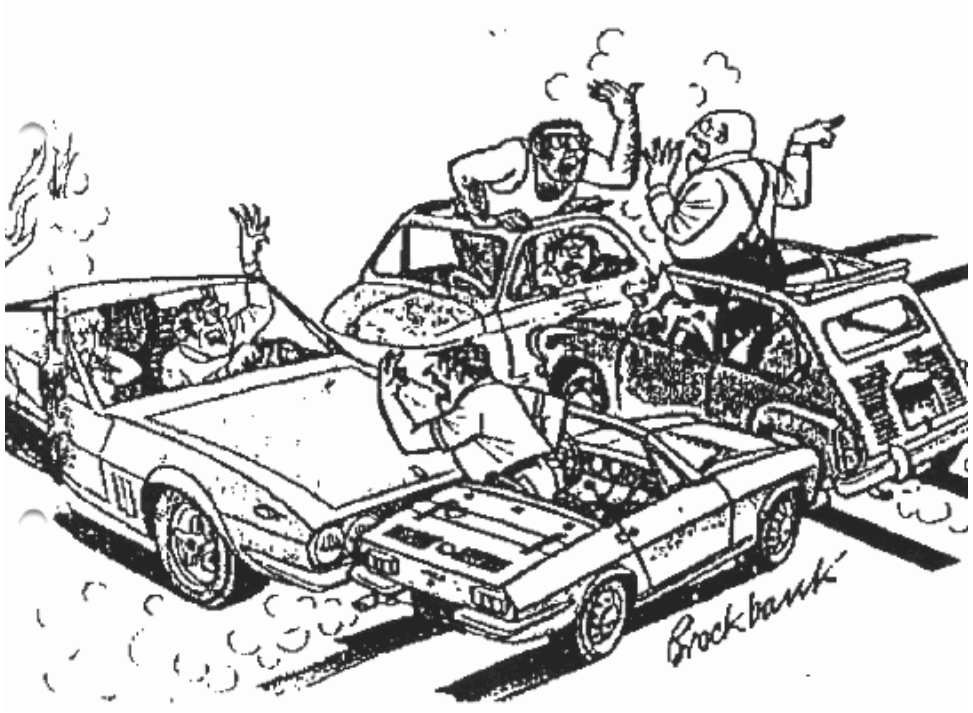
Ein erstes Beispiel: Wieviel Geld ist in Umlauf?

- konstante Geldmenge, oder
- monotone Inflation (--> Untergrenze)

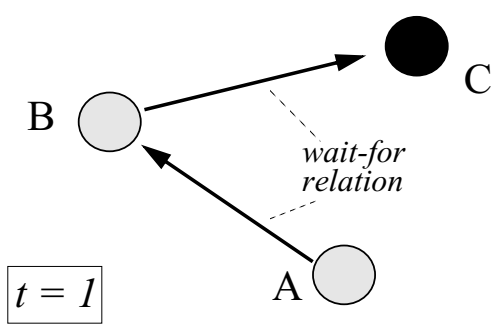


- Modellierung:
  - verteilte Geldkonten
  - ständige Transfers zwischen den Konten
- Erschwerte Bedingungen:
  - niemand hat eine globale Sicht
  - es gibt keine gemeinsame Zeit ("Stichtag")
- Anwendung: z.B. verteilte DB-Sicherungspunkte

Ein zweites Beispiel: Das Deadlock-Problem

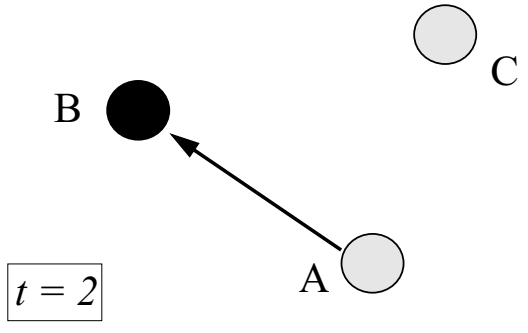


Phantom-Deadlocks

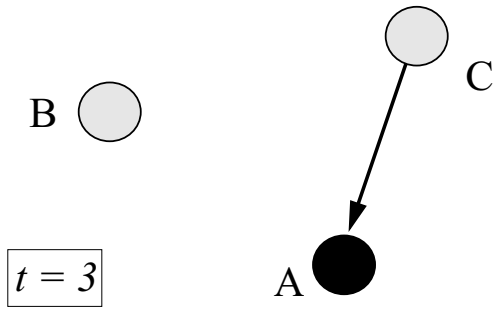


(C benutzt ein exklusives Betriebsmittel)

beobachte B:
==> B wartet auf C



beobachte A:
==> A wartet auf B

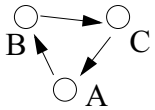


beobachte C:
==> C wartet auf A

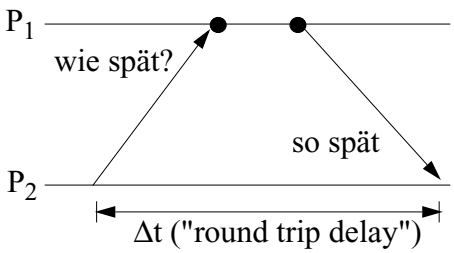
falscher Schluss!

Deadlock!

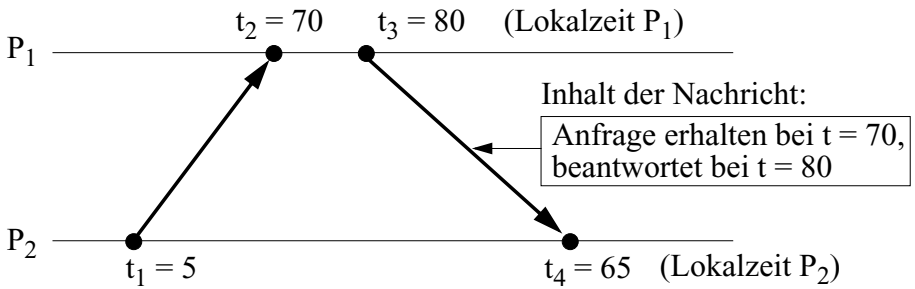
Keine exakte globale Zeit!



Ein drittes Problem: Uhrensynchronisation



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

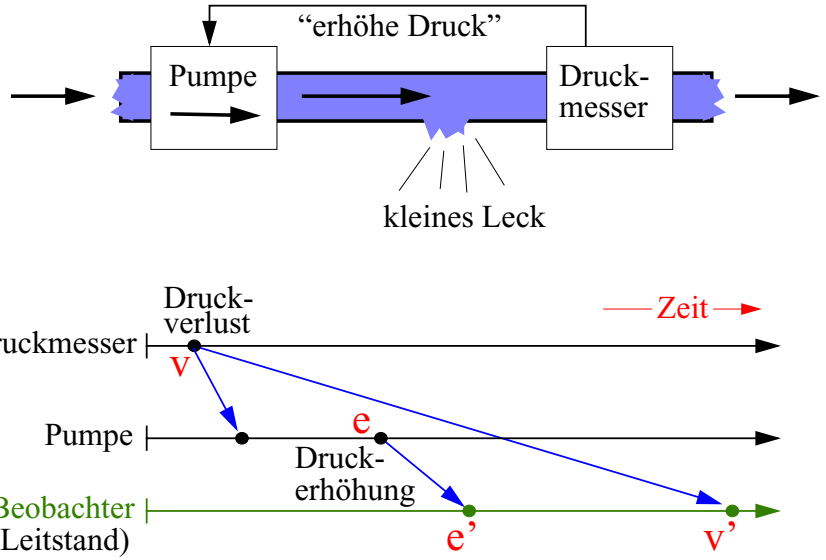


- Uhren gehen nicht unbedingt gleich schnell!
  (wenigstens "Beschleunigung ≈ 0", d.h. konstanter Drift gerechtfertigt?)

- Wie kann man den Offset der Uhren ermitteln oder zumindest approximieren?

Ein viertes Problem: Kausal (in)konsistente Beobachtungen

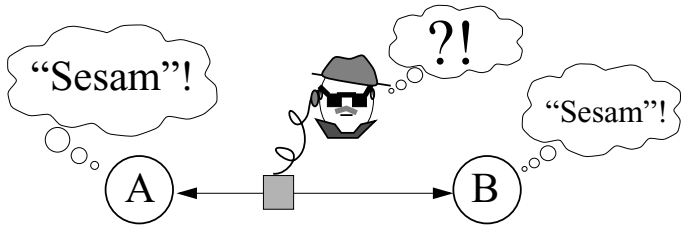
- Gewünscht: Eine Ursache stets vor ihrer (u.U. indirekter) Wirkung beobachten



Falsche Schlussfolgerung des Beobachters:

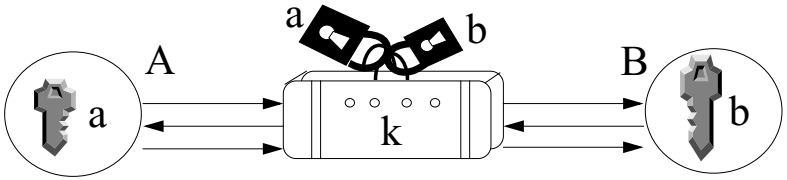
Eine unbegründete Pumpenaktivität erhöhte den Druck bis zum Bersten der Pipeline; daraufhin trat das Öl aus dem Leck aus, was durch den Druckverlust angezeigt wird!

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:



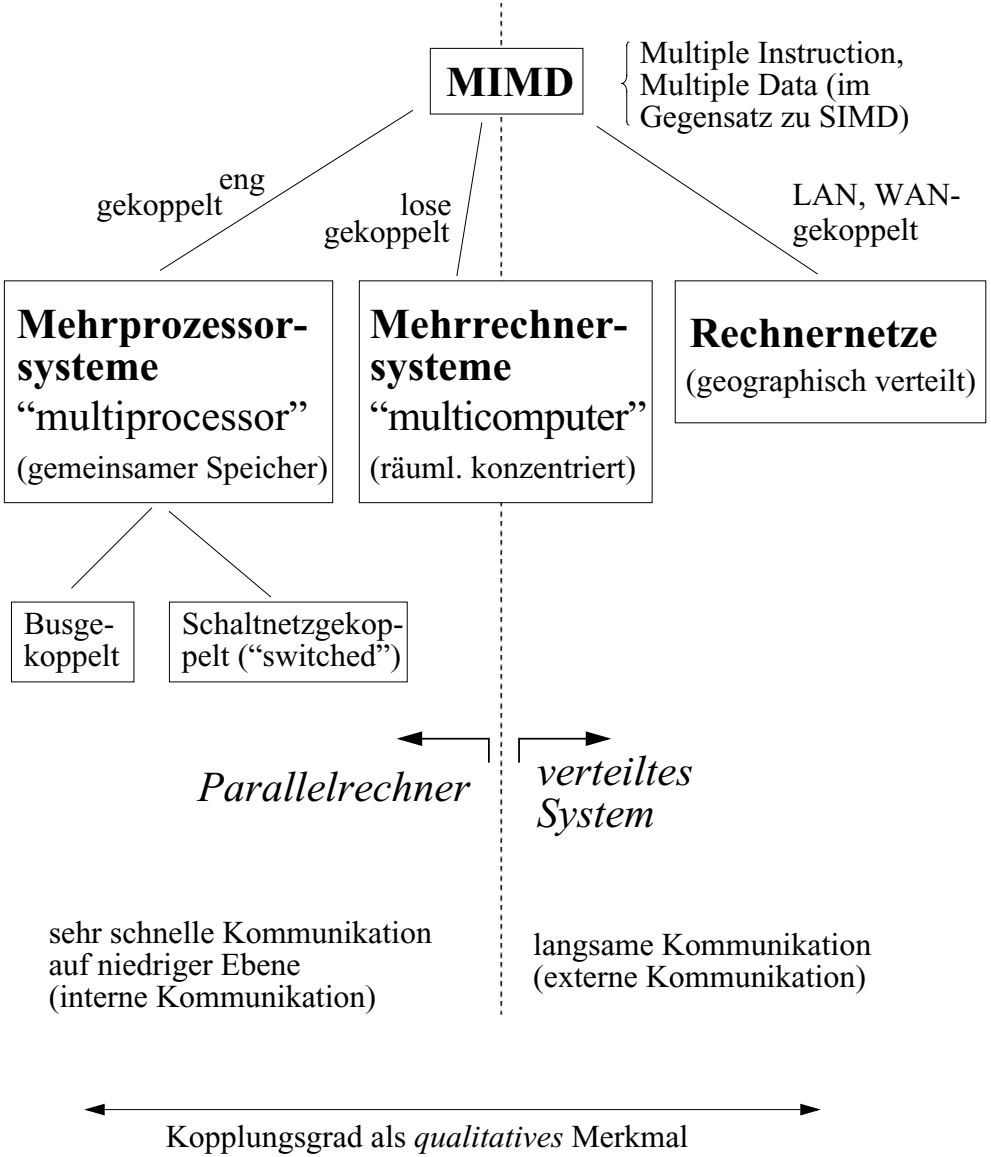
1. A denkt sich Passwort k aus und tut es in die Truhe.
2. A verschliesst die Truhe mit einem Schloss a.
3. A sendet die so verschlossene Truhe an B.
4. B umschliesst das ganze mit seinem Schloss b.
5. B sendet alles doppelt verschlossen an A zurück.
6. A entfernt Schloss a.
7. A sendet die mit b verschlossene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

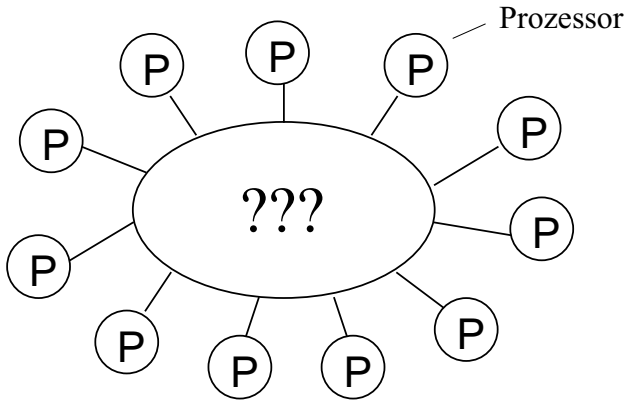
Multiprozessoren und Multicomputer

Abgrenzung Parallelrechner



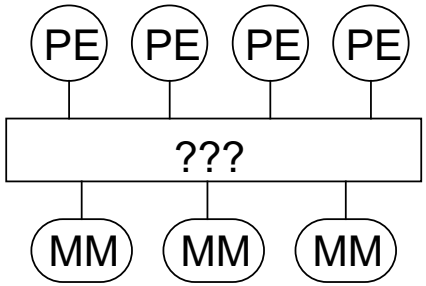
Prozessorverbund

- Autonome Prozessoren + „Kommunikationsnetz“
- Je nach Kopplungsgrad und Grad der Autonomie ergibt sich daraus ein
  - Mehrprozessorsystem
  - Mehrrechnersystem
  - Rechnernetz



Speicherkopplung

- Shared Memory
  - Kommunikation über gemeinsamen Speicher



Processing Elements

Memory Modules

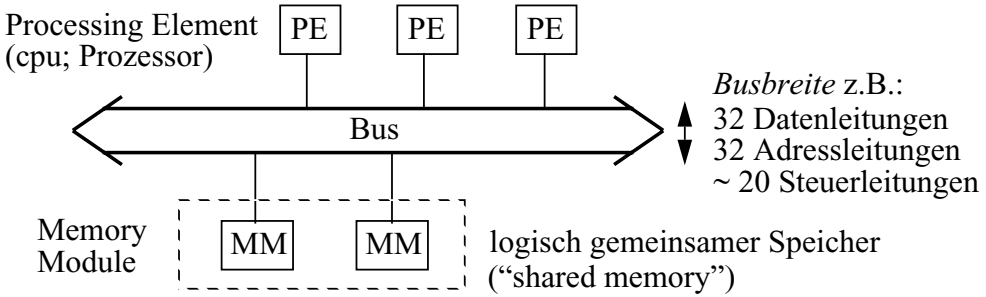
- n Processing Elements teilen sich k Memory Modules

- 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

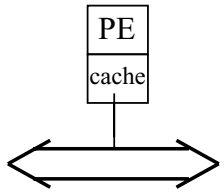


Problem:

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

Lösung:

Lokale Caches
zwischen PE und Bus:



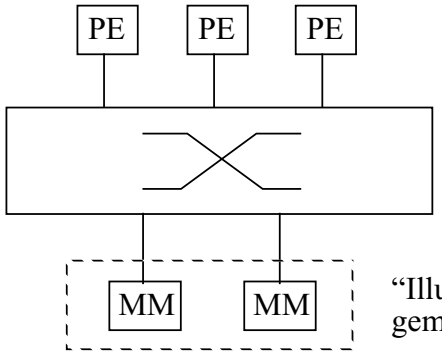
Cache gross genug wählen, um Hitraten > 90% zu erzielen (abhängig von der Hauptspeichergrösse)!

Probleme:

1) Kohärenzproblem der caches
2) Damit Problem nur verschoben
   (ca. 10 Mal mehr Prozessoren möglich)

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

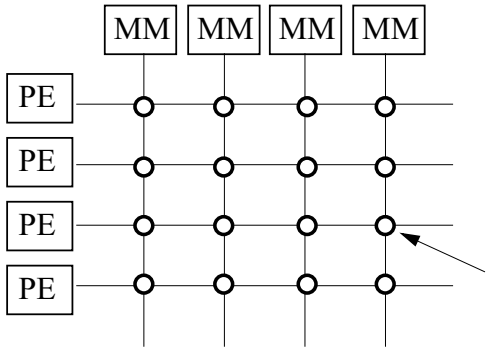
Schaltnetzgekoppelte Multiprozessoren



Schnelles Verbindungsnetz

"Illusion" eines gemeinsamen Speichers

Z.B. Crossbar-switch (Kreuzschienenverteiler):



elektronische Schalter ("crosspoint switch")

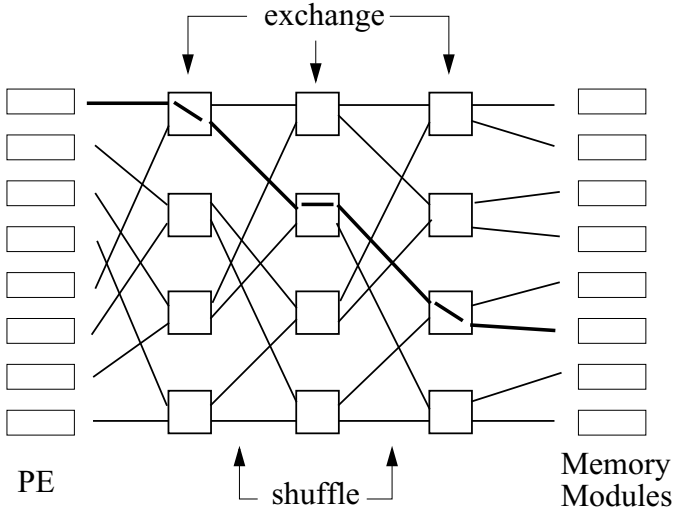
- Mehrere PEs können gleichzeitig auf verschiedene Speichermodule zugreifen
- Schlecht skalierbar (quadratisch viele Schalter)
  (Vermeidung von hot spots durch interleaving, Randomisierung...)

Permutationsnetze

Mehrere Stufen von Schaltelementen ermöglichen die Verbindung jedes Einganges zu jedem Ausgang.

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

straight through exchange



Beispiel:

*Shuffle-Exchange-*Netz

(Omega-Netz)

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

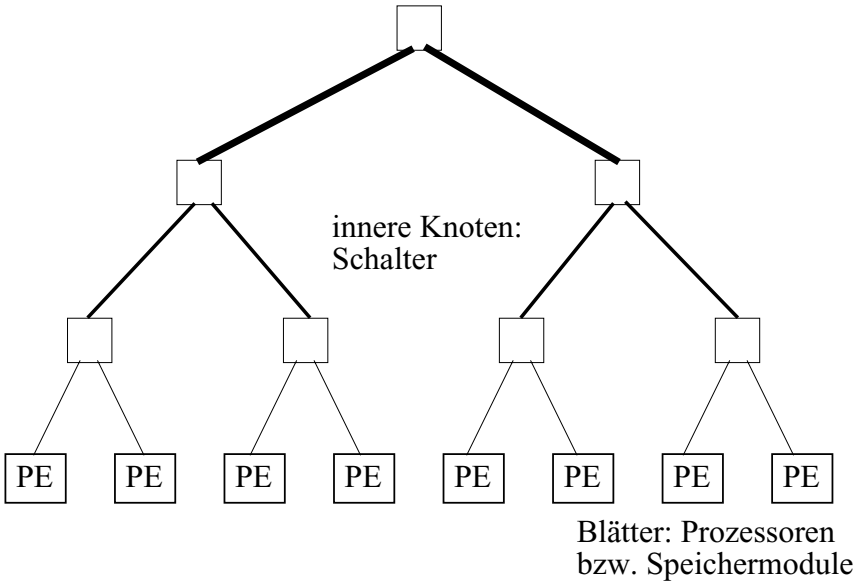
Es gibt weitere ähnliche dynamisch schaltbare Netze.

Designkriterien:

z.B. Butterfly-Netze

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

Fat-Tree-Netze



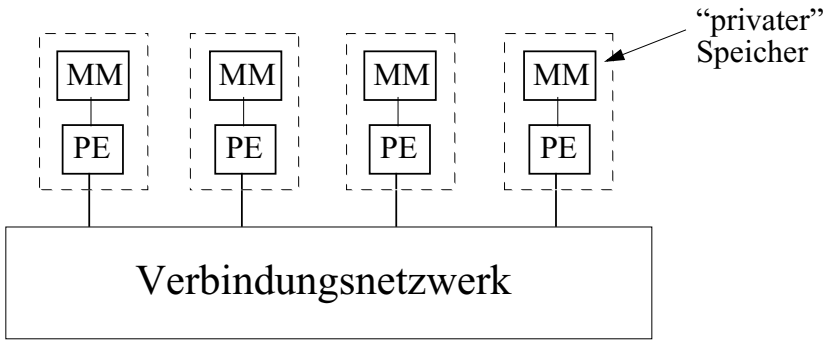
Verbindungsleitungen höherer Bandbreite bzw. mehrere parallele Leitungen auf Niveaus, die näher an der Wurzel liegen.

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
- Lokale PE-Caches sinnvoll
  - Problem der Cache-Kohärenz
- Bewertungskriterien für Verbindungsnetze
  - Realisierungsaufwand (Fläche, Kosten)
  - Skalierbarkeit (mit wachsender Anzahl PEs und MMs)
  - innere Blockadefreiheit (parallele Kommunikationsvorgänge)
  - Anzahl der Stufen (Verzögerung)
  - Eingangsgrad, Ausgangsgrad der Bauelemente

Mehrrechnersysteme ("Multicomputer")

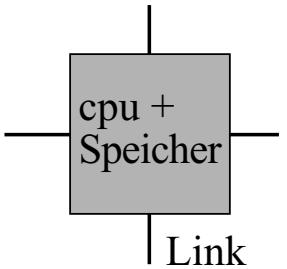
Vernetzung vollständiger Einzelrechner:



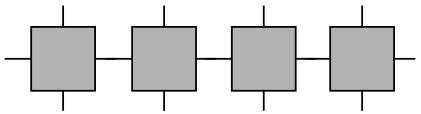
Zugriff auf andere Rechner (bzw. deren private Speicher) nur indirekt über *Nachrichten*.

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

Beispiel: Transputer als Baustein für Multicomputer

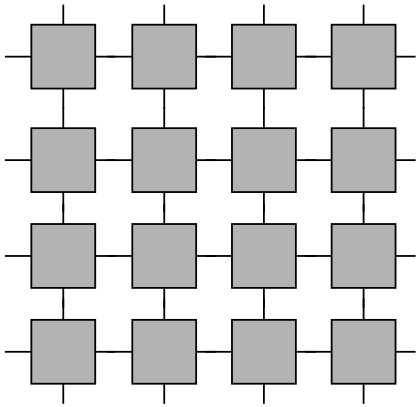


- Typische Topologien:



Pipeline

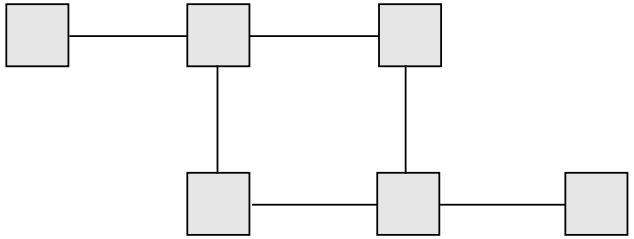
Array (oder Torus)



- bidirektional über mehrere Meter
- kleine set-up time
- synchrone Komm.
- Mehrprozesskonzept mit Scheduler "on chip"
- Eigene Programmiersprache "Occam"

- Hersteller: INMOS (GB)
- Erste Modelle: 1983; T414 (1986), T800 (1987)
  - in den 1990er-Jahren keine Nachfolgetypen mehr entwickelt

Verbindungstopologien für Mehrrechnersysteme



Zusammenhängender Graph mit

Knoten = Rechner

Kante = dedizierte Kommunikationsleitung

Ausdehnung: i.a. nur wenige Meter

Bewertungskriterien:

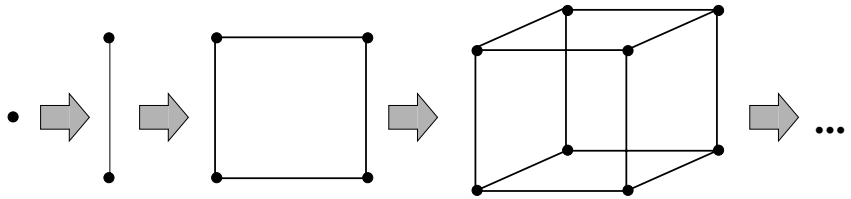
- Gesamtzahl der Verbindungen (bei n Knoten)
- maximale Entfernung zweier Knoten
- durchschnittliche Entfernung
- Anzahl der Nachbarn eines Knotens (“fan out”)
- Symmetrie, Homogenität, Skalierbarkeit...
- Routingkomplexität
- Zahl der alternativ bzw. parallel verfügbaren Wege

Technologische Faktoren:

- Geschwindigkeit, Durchsatz, Verzögerung, eigene Kommunikationsprozessoren...

Hypercube

- Hypercube = "Würfel der Dimension d"



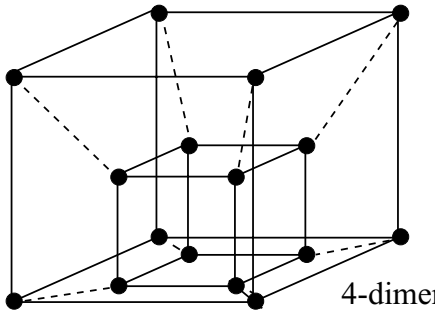
Draufsicht von der Seite liefert jeweils niedrigere Dimension

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

Man vgl. auch das Buch von T. F. Banchoff: Beyond the Third Dimension (Scientific American Library, 1990)

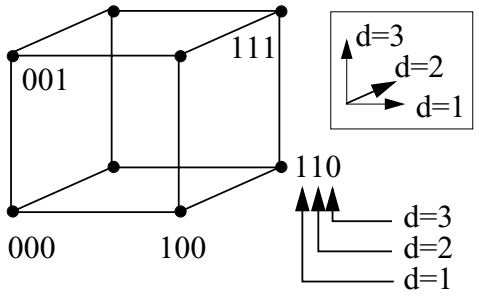
Hypercube der Dimension d

- $n = 2^d$ Knoten

- Anzahl der Nachbarn eines Knotens = d
  (Anzahl der “ports” in der Hardware)

- Gesamtzahl der Kanten (= Verbindungen): $d\ 2^d/2 = d\ 2^{d-1}$
  (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!)

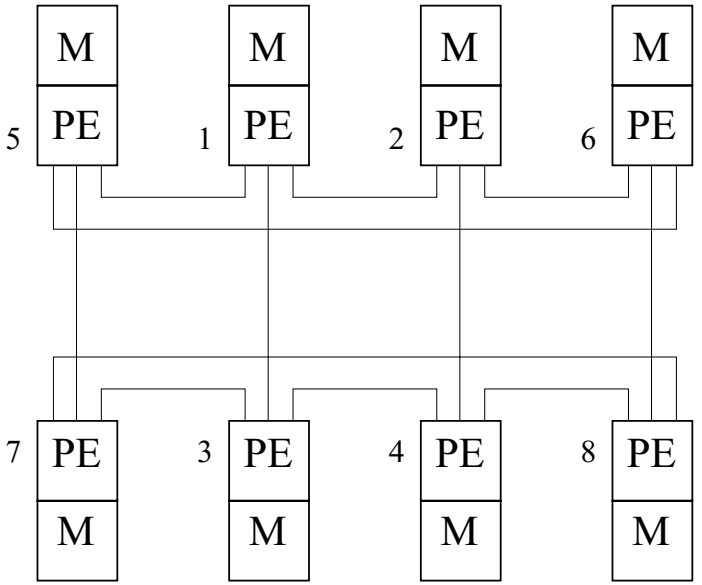
Denkübung: mittlere Weglänge?

-Nachteile:

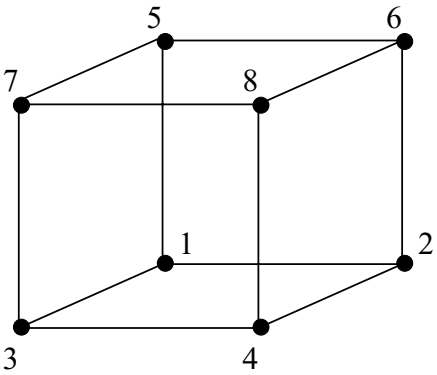
- Anzahl der Nachbarknoten eines Knotens wächst mit der Dimension d
- insgesamt relativ viele Verbindungen: O(n log n)
  (eigentlich genügen n-1)

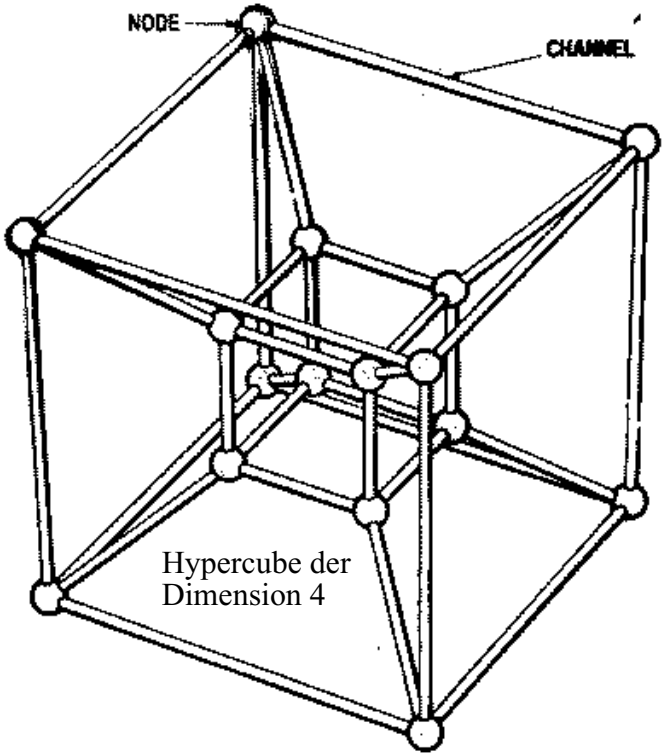
wieviele verschiedene Wege der Länge k gibt es insgesamt?

Layout eines Hypercube



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





Hypercube der
Dimension 4

Flatland (1884)

- Diskussion über Hypercubes und höhere geometrische Dimensionen zwischen “A. Square” und “Sphere”

- Gleichzeitig soziale Satire

§ 16 .---*How the Stranger vainly endeavoured to reveal to me in words the mysteries of Spaceland*

...

Sphere. ... We began with a single Point, which of course -- being itself a Point -- has only ONE terminal Point. One Point produces a Line with TWO terminal Points.

One Line produces a Square with FOUR terminal Points.

Now you can give yourself the answer to your own question: 1, 2, 4, are evidently in Geometrical Progression. What is the next number?

I. Eight.

Sphere. Exactly. The one Square produces a SOMETHING-WHICH-YOU-DO-NOT-AS-YET-KNOW-A-NAME-FOR-BUT-WHICH-WE-CALL-A-CUBE with EIGHT terminal Points. Now are you convinced?

...

Sphere. How can you ask? And you a mathematician! The side of anything is always, if I may so say, one Dimension behind the thing. Consequently, as there is no Dimension behind a Point, a Point has 0 sides; a Line, if I may so say, has 2 sides (for the points of a Line may be called by courtesy, its sides); a Square has 4 sides; 0, 2, 4; what Progression do you call that?

I. Arithmetical.

Sphere. And what is the next number?

I. Six.

Sphere. Exactly. Then you see you have answered your own question. The Cube which you will generate will be bounded by six sides,that is to say, six of your insides. You see it all now, eh?

"Monster," I shrieked, "be thou juggler, enchanter, dream, or devil, no more will I endure thy mockeries. Either thou or I must perish." And saying these words I precipitated myself upon him.

Online-Text erhältlich bei: http://www.geom.umn.edu/~banchoff/Flatland/ oder http://www.alcyone.com/max/lit/flatland/

Das *Buch* ist erhältlich in mehreren Ausgaben; z.B.: Abbott, Edwin A.: Flatland. Penguin, 1987, ISBN: 0140076158, DM 11,90

§ 3 .---*Concerning the Inhabitants of Flatland*

...

Our Women are Straight Lines.

Our Soldiers and Lowest Class of Workmen are Triangles with two equal sides, each about eleven inches long...

Our Middle Class consists of Equilateral or Equal-Sided Triangles.

Our Professional Men and Gentlemen are Squares (to which class I myself belong) and Five-Sided Figures or Pentagons.

Next above these come the Nobility, of whom there are several degrees, beginning at Six-Sided Figures, or Hexagons, and from thence rising in the number of their sides till they receive the honourable title of Polygonal, or many-Sided. Finally when the number of the sides becomes so numerous, and the sides themselves so small, that the figure cannot be distinguished from a circle, he is included in the Circular or Priestly order; and this is the highest class of all.

...

§ 4 .---*Concerning the Women*

If our highly pointed Triangles of the Soldier class are formidable, it may be readily inferred that far more formidable are our Women.

...

But here, perhaps, some of my younger Readers may ask HOW a woman in Flatland can make herself invisible. This ought, I think, to be apparent without any explanation. However, a few words will make it clear to the most unreflecting.

Place a needle on the table. Then, with your eye on the level of the table, look at it side-ways, and you see the whole length of it; but look at it end-ways, and you see nothing but a point, it has become practically invisible. Just so is it with one of our Women.

...

The dangers to which we are exposed from our Women must now be manifest to the meanest capacity of Spaceland. If even the angle of a respectable Triangle in the middle class is not without its dangers... --what can it be to run against a woman, except absolute and immediate destruction? And when a Woman is invisible, or visible only as a dim sub-lustrous point, how difficult must it be, even for the most cautious, always to avoid collision!

...

In the Southern and less temperate climates, where the force of gravitation is greater, and human beings more liable to casual and involuntary motions, the Laws concerning Women are naturally much more stringent. But a general view of the Code may be obtained from the following summary:--

1. Every house shall have one entrance on the Eastern side, for the use of Females only; by which all females shall enter "in a becoming and respectful manner" and not by the Men's or Western door.

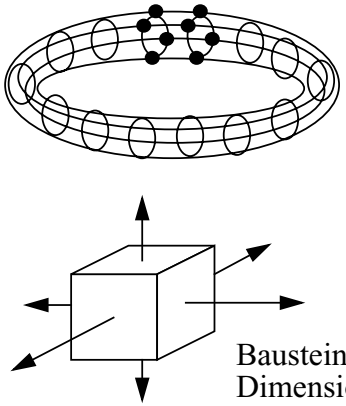
2. No Female shall walk in any public place without continually keeping up her Peace-cry, under penalty of death.

3. ...

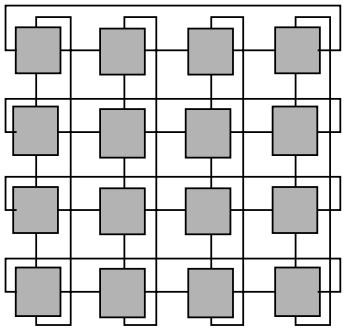
In some of the States there is an additional Law forbidding Females, under penalty of death, from walking or standing in any public place without moving their backs constantly from right to left so as to indicate their presence to those behind them...

Eine andere Verbindungstopologie: der d-dimensionale Torus

= d-dimensionales "wrap-around Gitter"



Baustein für 3 Dimensionen



2 Dimensionen

- Rekursives Konstruktionsprinzip: „Nimm w_{d-1} gleiche Tori der Dimension d-1 und verbinde korrespondierende Elemente zu Ring"

- Bei Ausdehnung w_i in Dimension i:
  $n = w_1 \times w_2 \times .. \times w_d$ Knoten;
  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!

  --> $\Delta = \frac{1}{4} \sum_d 2 = \frac{1}{4}(2\,d) = \frac{d}{2} = \underline{\underline{\frac{1}{2} \log_2 n}}$