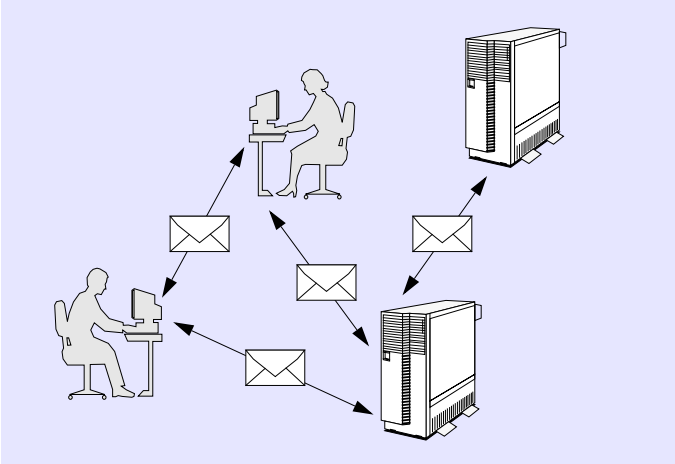
Verteilte Systeme

Friedemann Mattern

Departement Informatik
ETH Zürich



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

Übersicht I

1. Verteilte Systeme: Definition und Motivation
2. Multiprozessoren, Multicomputer
3. Überblick Rechnernetze
   - LAN, WAN
   - Schichtenmodell, Protokolle

   ← teilweise Wiederholung aus dem Grundstudium
4. Kommunikation
   - Mitteilung / Auftrag
   - synchron / asynchron
   - Puffer
   - RPC (Protokolle, Fehlersemantik)
   - Mailbox, Ports, Kanäle
   - Gruppenkommunikation (Broadcast-Semantik)
   - verteilte Ereignisse
   - Tupelräume
5. Verteilte Programmiersprachen
   - Occam als Beispiel
6. Namensverwaltung
7. Schutz und Sicherheit
   - Autorisierung, Capabilities
   - Authentifizierung
   - DES, Public-key-Systeme
   - Kerberos

   ← teilweise Wiederholung aus dem Grundstudium

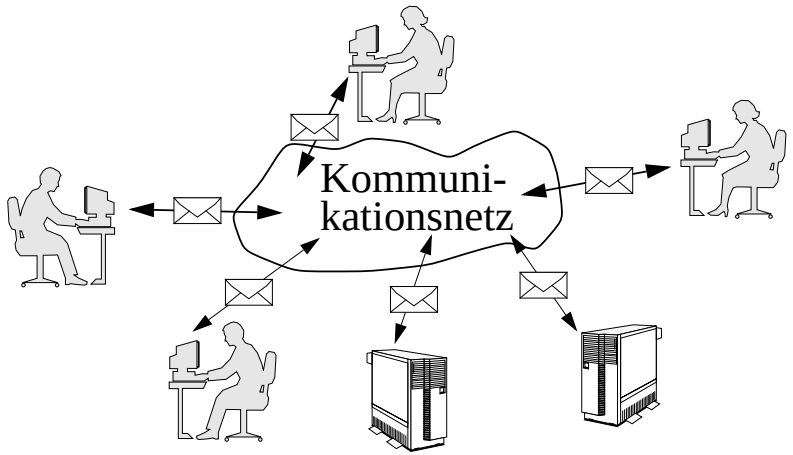
Übersicht II

8. Client / Server-Modell
   - verteilte Betriebssysteme
   - konkurrente Server und Servergruppen
   - zustandsbehaftete / zustandslose Server
   - verteilte Dienste
9. Middleware: Software für offene, verteilte Systeme
   - Sun-RPC
   - OSF-DCE
   - CORBA
   - Jini
10. Logische Zeit, verteilte Algorithmen
    - Lamport-Uhren
    - wechselseitiger Ausschluss
11. Physische Zeit, Uhrensynchronisation
12. Mobiler Code und mobile Agenten

"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*

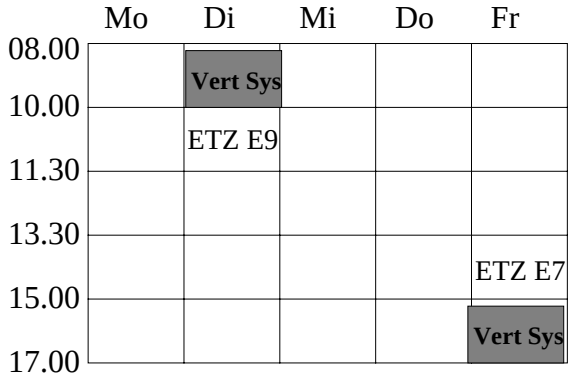
- welche Problemaspekte stecken hinter Lamports Charakterisierung?

Organisatorisches zur Vorlesung

4-stündige Vorlesung (inkl. Übungen)

Sinnvolle Vorkenntnisse:

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



Di 08:15 - 10:00, ETZ E9
Fr 15:15 - 17:00, ETZ E7
} *Vorlesung*

Absicht!

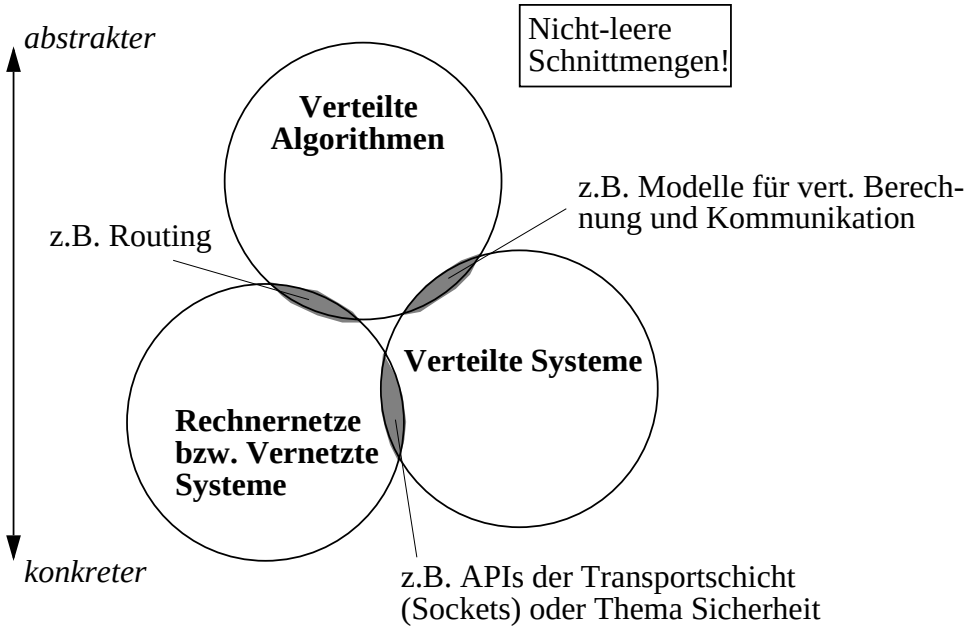
- Gelegentliche *Denkaufgaben* in der Vorlesung
- Praktische Übungen in Form eines begleitenden *Praktikums* (korreliert nur schwach mit dem Inhalt der Vorlesung) *komplementieren* Vorlesung
- Gelegentliche *Übungsstunden* (zu den Vorlesungsterminen) zur Besprechung des Praktikumteils

---

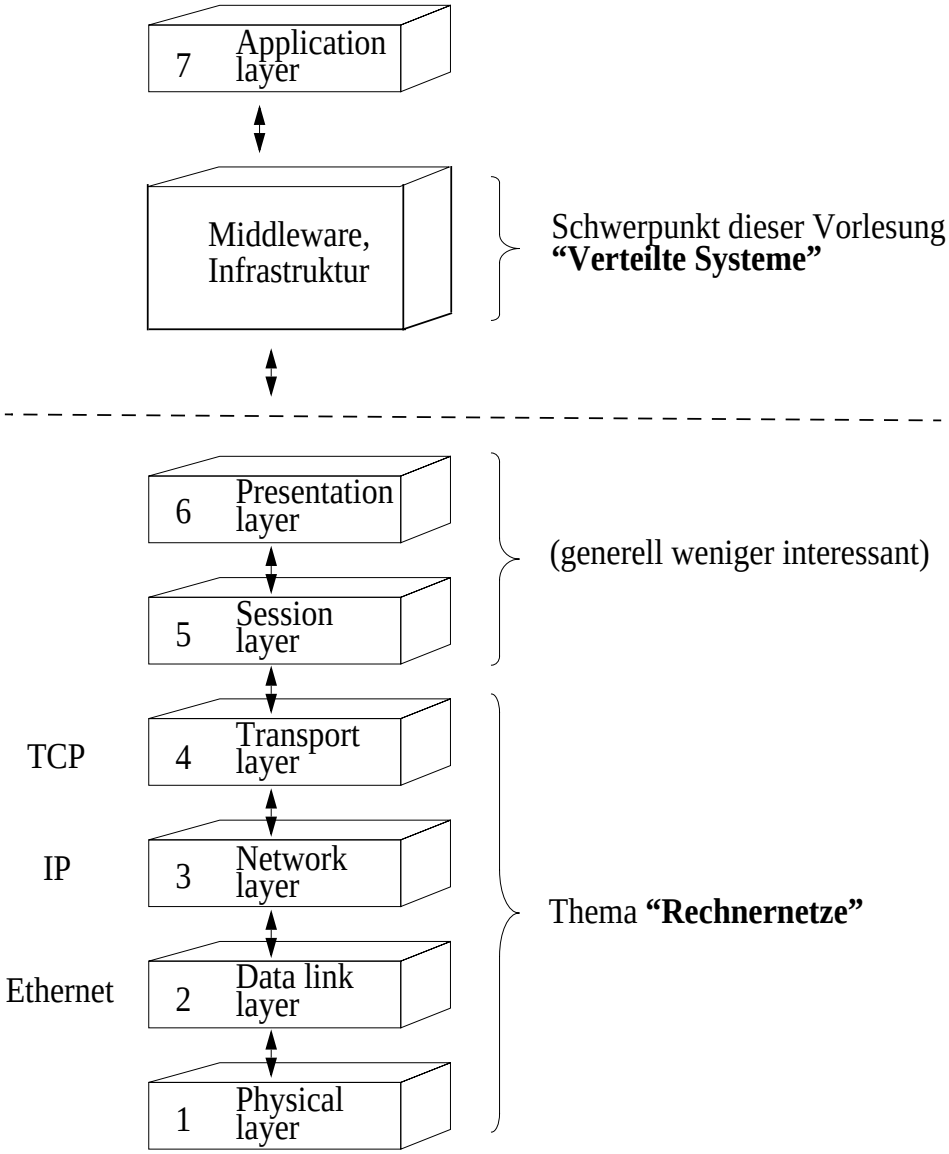
- *Folienkopien* jeweils in der darauffolgenden Vorlesung (auch im WWW im .ps- und .pdf-Format: www.inf.ethz.ch/vs/education/)

Thematisch verwandte Veranstaltungen im Fachstudium

- Mattern (bzw. Plattner): *Verteilte Systeme*, 4st
- Mattern (bzw. Widmayer): *Verteilte Algorithmen*, 3st
- *Mobile Computing* (geplant)
- *Ubiquitous Computing* (WS 2000/01)
- Einschlägige *Seminare*
- *Semester- und Diplomarbeiten*



Netze, Anwendungen, Verteilte Systeme



Literatur (vorläufig)

G. Coulouris, J. Dollimore, T. Kindberg: *Distributed Systems: Concepts and Design (2nd ed.).* Addison-Wesley, 1994

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

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

A. Tanenbaum: *Distributed Operating Systems*. Prentice-Hall, 1995

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

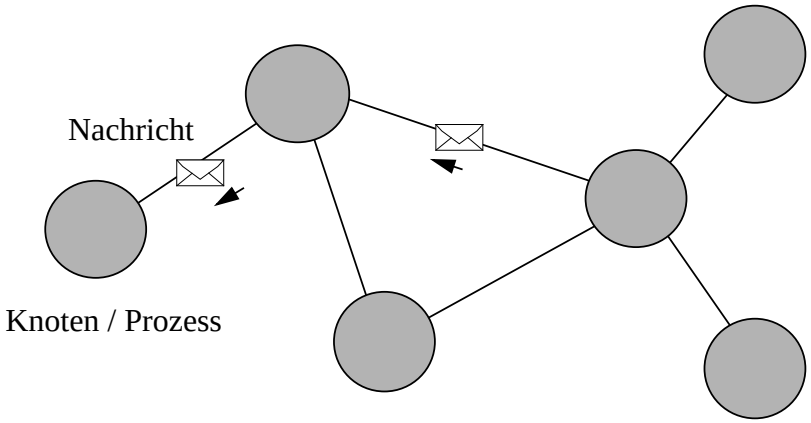
A. Tanenbaum: *Computer Networks (3rd ed.)*. Prentice-Hall, 1996

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.

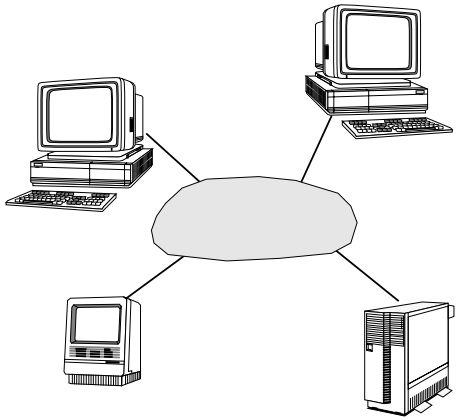
“Verteiltes System”



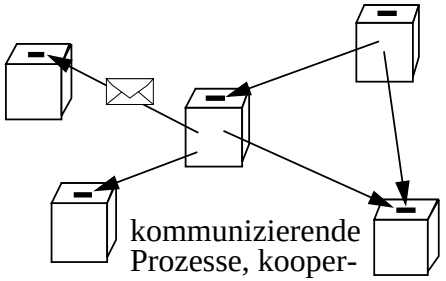
Physisch verteiltes System:
Mehrrechnersystem ... Rechnernetze

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

Sichten verteilter Systeme



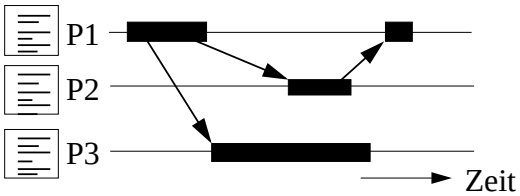
Rechnernetz mit Rechenknoten:
- Multicomputer (Parallelrechner)
- LAN = Local Area Network
- WAN = Wide Area Network
- Routing, Adressierung....



kommunizierende Prozesse, kooperierende Objekte

Objekte eines *Betriebssystems* bzw. einer *Programmiersprache*

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



Algorithmen- und Protokoll-ebene

- Aktionen, Ereignisfolgen
- Konsistenz, Korrektheit

zunehmende Abstraktion

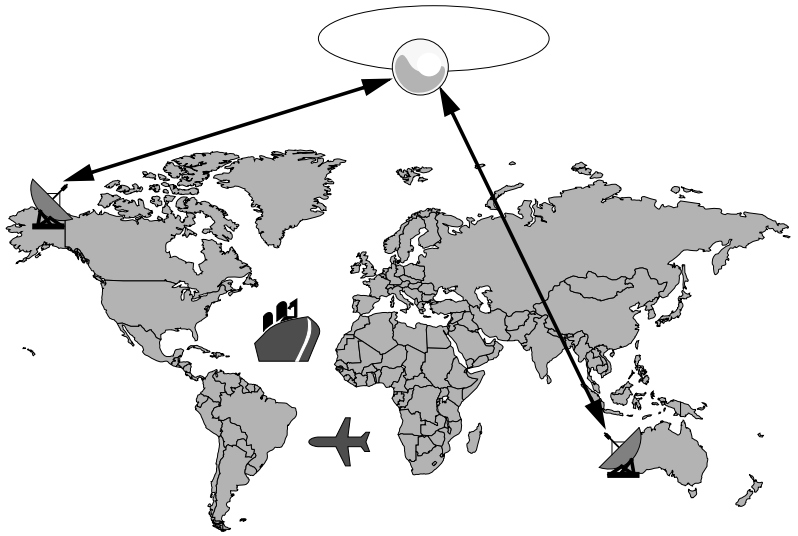
Sichten (2)

- Von Technologie bis zu Prinzipien
  - physische Sicht: z.B. Techniker
  - Programmiersicht: z.B. Programmierer, Systementwickler
  - abstrakte Sicht: z.B. Algorithmentheoretiker

- Entspricht dem Kanon der verschiedenen Lehrveranstaltungen:
  - „Rechnernetze“: technische Eigenschaften, Topologien, Realisierungsweisen
  - **„Verteilte Systeme“**: Infrastruktur, Komponenten, Protokolle
  - „Verteilte Algorithmen“: Korrektheit, grundsätzliche Phänomene, Konzeptualisierung

- „*Logisch* verteilte Systeme“: unabhängig von *physischer* Verteilung
  - fehlender globaler Zustand, fehlende globale Zeit
  - Parallelität
  - Autonomie (z.B. unabhängige Prozesse, Objekte, Agenten …)

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

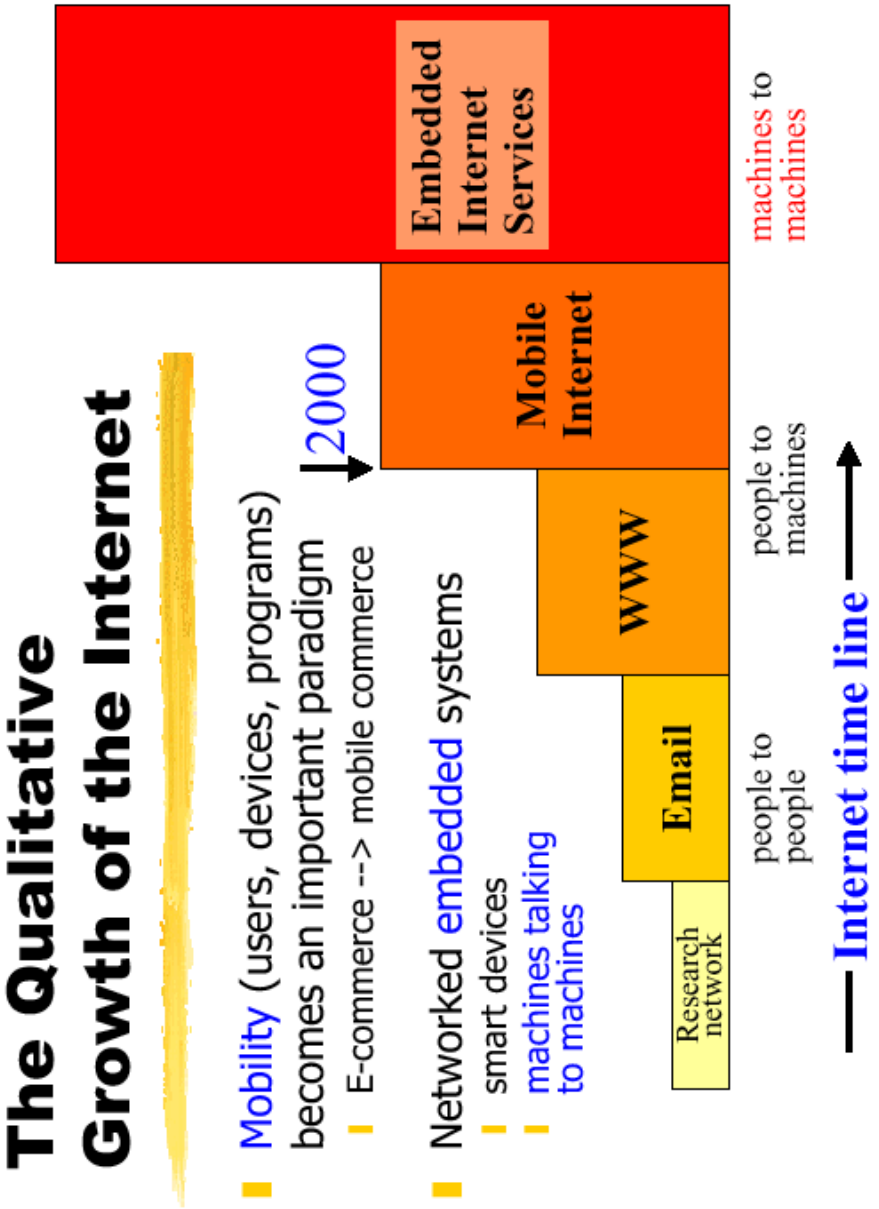
- „Aufsammeln“ ungenutzter Rechenleistung im Internet
  - Faktorisierung grosser Zahlen per Email
  - Brechen einer DES-Verschlüsselung

- 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

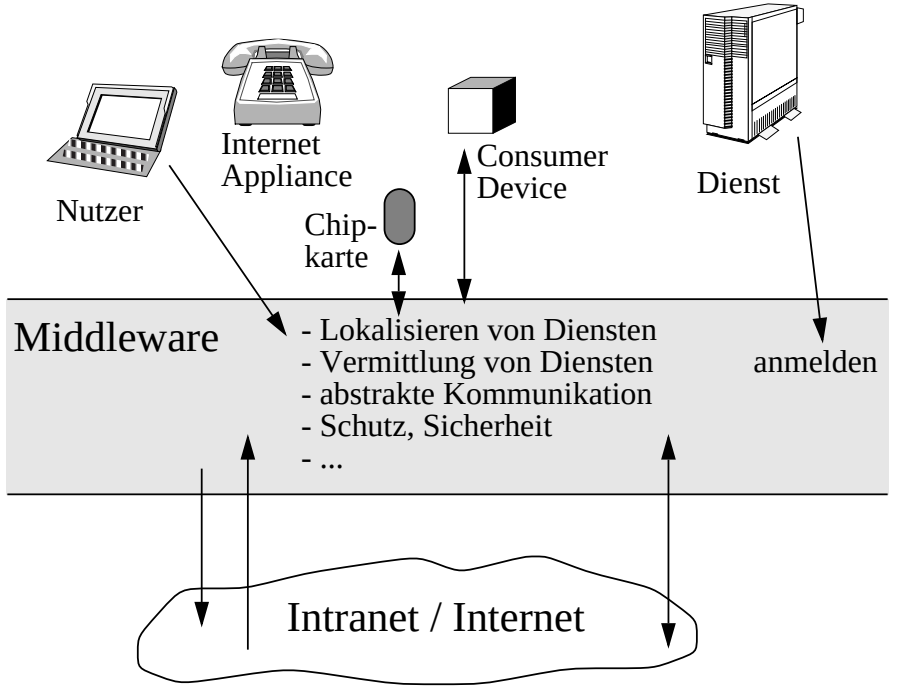
- Phänomen: das Internet verbreitet sich immer weiter
  - mehr Nutzer, Popularisierung
  - bis in die Haushalte
  - immer exotischere Endgeräte (PDA, Handy, Kühlschrank, Chipkarte)
  - bald enthalten vielleicht auch Briefmarken, Kleidungsstücke etc. kommunikationsfähige Chips
- Mobile Geräte, dynamische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

- Betriebssystem (--> alle Anwendungen profitieren davon)
- Subkomponenten (z.B. Dateisystem)
- Anwendungsprogramm (Nutzung der Semantik)

Transparenz ist *graduelle Eigenschaft.*

Bsp. Ortstransparenz:

schwach: Benutzer sieht verschiedene Rechner

- Zielrechner muss bei Prozessgründung, login... angegeben werden
- Bsp. UNIX: remote login, ftp, rexec

mittel: Einige einfache “verteilte Anwendungen”

- Bsp. UNIX: rwho, email (ortstransparente Adressen)

hoch: Z.B. wichtige “netzwerkweite” Dienste

- Bsp: NFS: Benutzung von Dateien unabhängig vom Ort
- Bsp. XWindows

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 Standardrechner haben i.a. besseres Preis-Leistungsverhältnis als Grossrechner
  --> Anwendung falls möglich “verteilen” (“downsizing”, outsourcing)
- *Geschwindigkeit*: Falls Anwendung “gut” parallelisierbar, ist eine sonst unerreichbare Leistung möglich
  - ggf. (dynamische) Lastverteilung beachten
- *Globalisierung* von Produktentwicklung

- 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*
  - Gruppenarbeit (“CSCW”)
  - E-mail, 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...)
- Motivation für internationale Normung von Protokollen (OSI)
- 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 in Form von Produkten:
  - Kommunikation über LAN (Resource-Sharing)
  - Software für “Gruppenarbeit” (email, gem. Dateisystem...)

Forschungsprojekte

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

Kommerzielle Projekte

- z.B. Reservierungssysteme, Banken, Kreditkarten
- kooperatives Arbeiten: Workflow, “CSCW”, joint authoring

“Historie” der Konzepte

- Concurrency, Synchronisation...
  - bereits klassisches Thema bei Datenbanken und Betriebssystemen
- Programmiersprachen
  - kommunizierende Objekte, CSP...
- Physische Parallelität
  - Array-, Pipeline-, Multiprozessoren
- Parallele und verteilte Algorithmen
- Semantik
  - math. Modelle, CCS, 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, gelegentlich 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 Systemen
  - 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

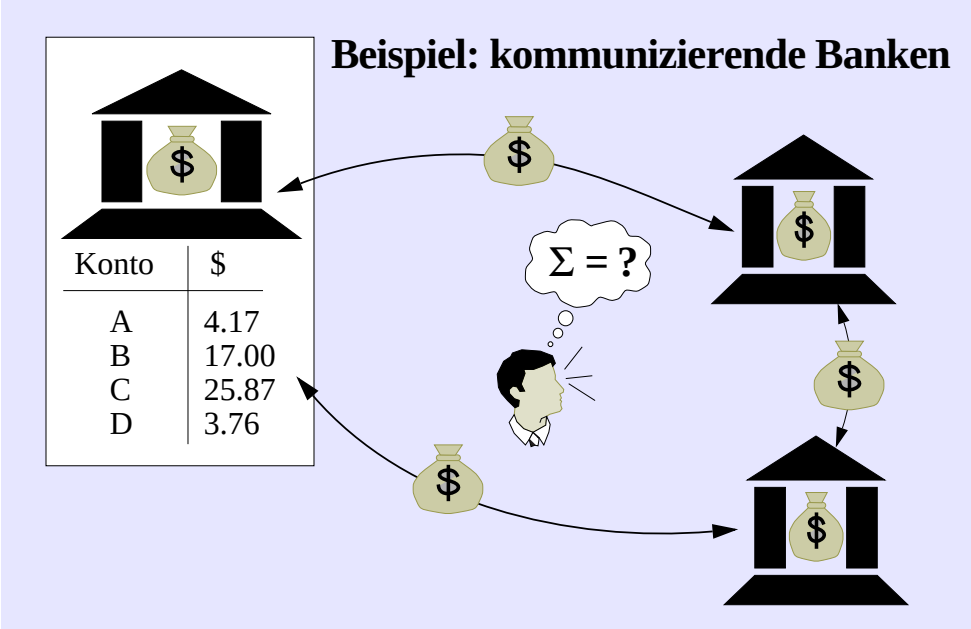
Einige *konzeptionelle* Probleme und Phänomene verteilter Systeme

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