Verteilte Systeme

1. Einführung

Sommersemester 2011

Institut für Betriebssysteme und
Rechnerverbund

TU Braunschweig

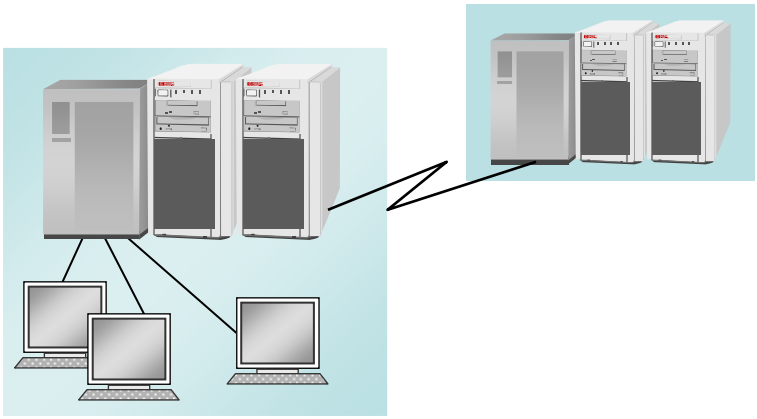
Dr. Christian Werner
– Bundesamt für Strahlenschutz –

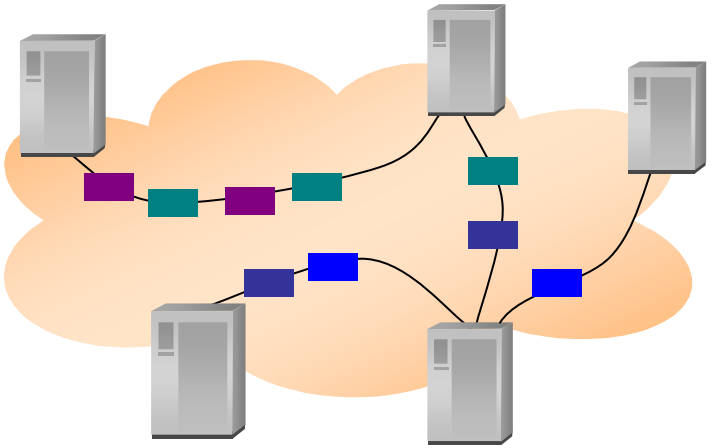
Überblick

- Entwicklung der Rechnerkommunikation
- Begriff des Verteilten Systems bzw. der Verteilten Anwendung
- Warum Verteilte Systeme?
- Verteilte Anwendungen heute
- Wünschenswerte Eigenschaften Verteilter Systeme
- Hardwarekonzepte verteilter Systeme
  - Multiprozessor
  - Multicomputer
- Softwarekonzepte
  - Verteilte Betriebssysteme
  - Netzwerkbetriebssysteme
  - Middleware
- Interaktionsmodelle
  - Client-Server
  - Multi-Server
  - Service-Oriented Architecture (SOA)
  - Multi-Tiered
  - Peer-to-Peer
  - Grid Computing
- Zusammenfassung

Computernetzwerke: 1960s-1970s

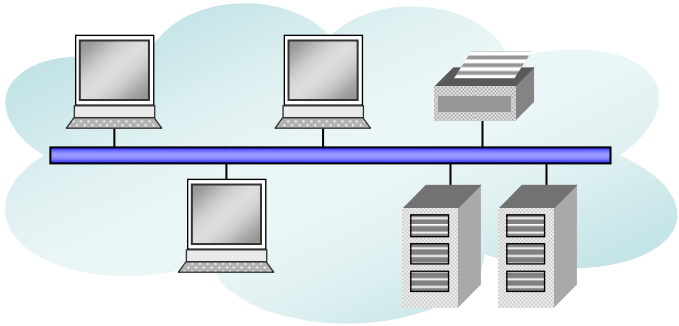
- Terminal-Host-Kommunikation über serielle Leitungen
- Host-zu-Host-Kommunikation
  - Basierend auf proprietären Netzwerken wie IBM SNA, DECnet
  - schon bald basierend auf Paketvermittlung und damit TCP/IP-Netzen

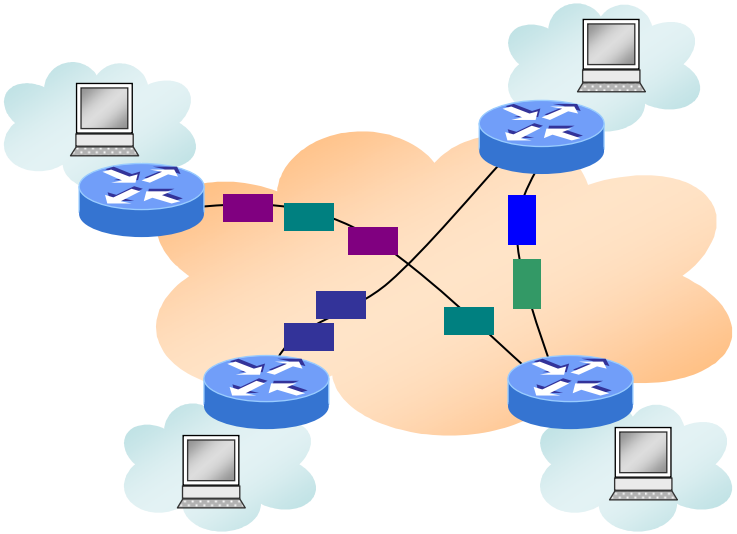




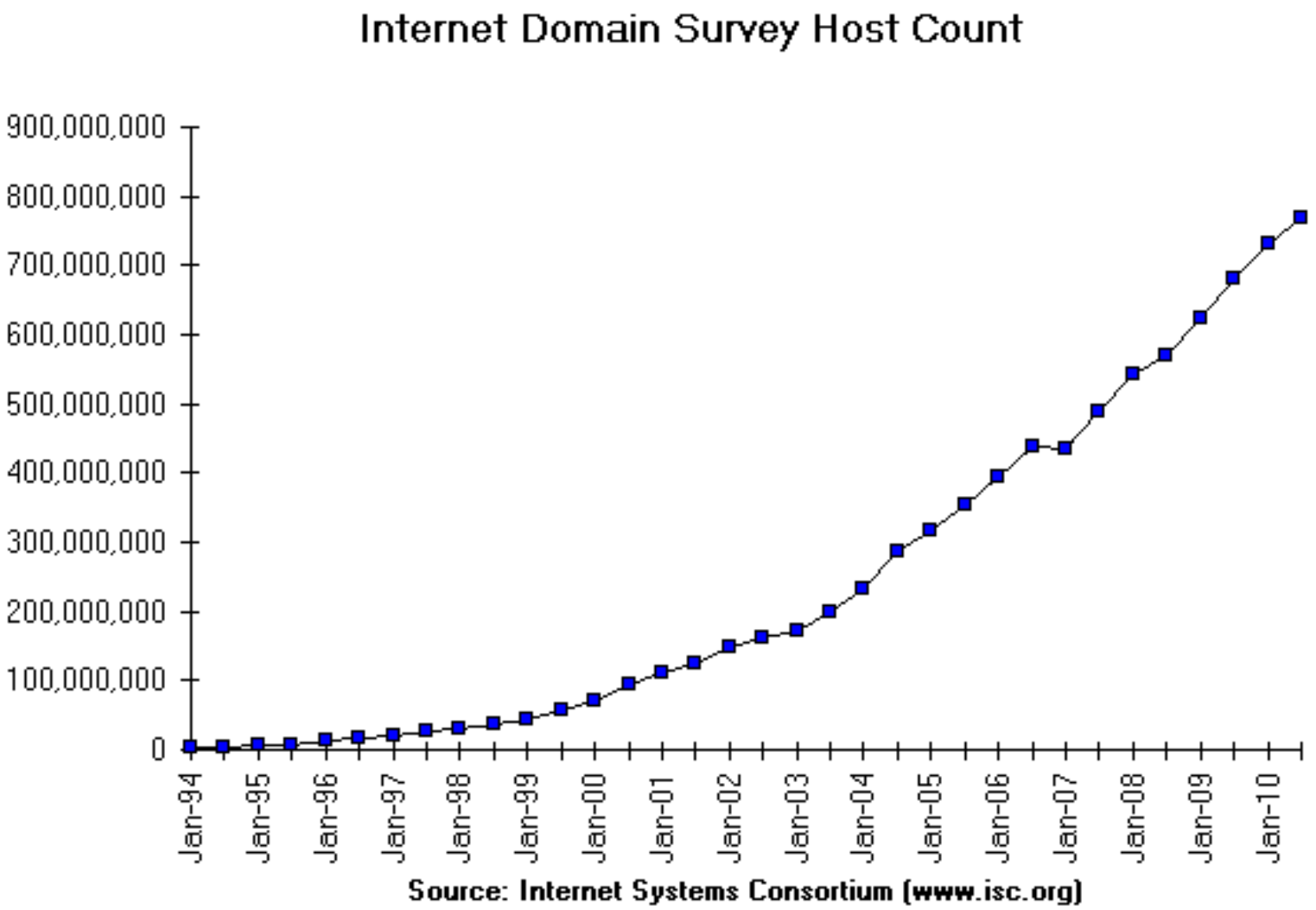
Computernetzwerke: 1980s

- Hochleistungs-LANs zu niedrigen Kosten
  - PC-Netze
  - Verteilte Client-Server-Systeme als neues Paradigma
- Öffentliche und private WANs
  - Ausgereifte Technik der Paketvermittlung
  - Wichtige neue Anwendungen (Email, File Transfer, Telnet, etc.)
- Offene Standards setzen sich eindeutig durch (OSI, TCP/IP)





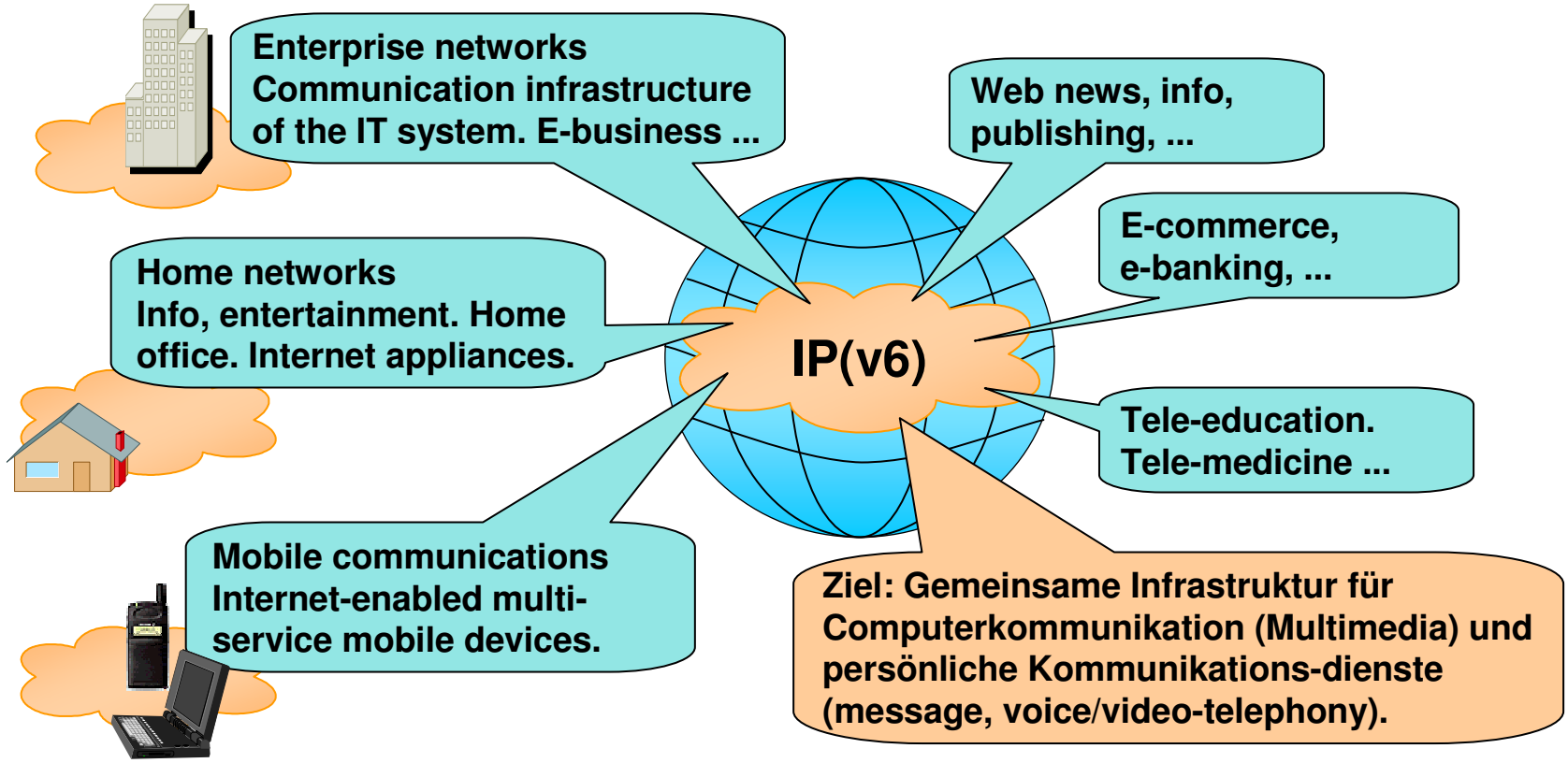
Computernetzwerke: Internet seit 1990



Computernetzwerke: 2000s

"IP on everything" (Vint Cerf, Internet-Patriarch)

In Richtung eines globalen Netzwerkes mit vielen Diensten und IP(v6) als dem gemeinsamen Kern.



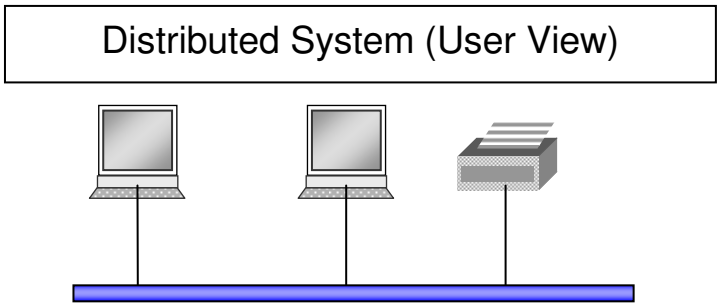
Netzwerke und verteilte Systeme

- Wir haben die Entwicklung der Computernetzwerke betrachtet.
- Netzwerke sind nicht im Fokus dieser Vorlesung, wir werden uns auf eine kurze Einführung beschränken
- ABER:
  - Man benötigt ein Computernetzwerk, um ein Verteiltes System zu realisieren.

- Was also ist ein Verteiltes System?

Was ist ein verteiltes System?

- Eine praxisorientierte Definition:
  - Ein Verteiltes System
    - besteht aus einer Menge autonomer Computer
    - die durch ein Computernetzwerk miteinander verbunden sind und
    - mit einer Software zur Koordination ausgestattet sind.



Was ist ein Verteiltes System?

- Eine allgemeinere Definition:
  - Ein Verteiltes System ist ein System, in dem
    - Hard- und Softwarekomponenten,
    - die sich auf miteinander Vernetzten Computern befinden,
    - miteinander kommunizieren und ihre Aktionen koordinieren,
    - indem sie Nachrichten austauschen.
  - Eine Verteilte Anwendung ist eine Anwendung,
    - die ein verteiltes System zur Lösung eines Anwendungsproblems nutzt.
    - Sie besteht aus verschiedenen Komponenten, die mit den Komponenten des VS sowie den Anwendern kommuniziert.
    - Wichtiger Aspekt dabei: Transparenz, d.h. die Verteiltheit wird vor dem Anwender verborgen.

Warum Verteilte Systeme?

- Durch die Verteilung von Systemkomponenten wird das Gesamtsystem normalerweise deutlich komplexer.
- Das heißt, der Aufwand zur Erstellung eines VS ist deutlich größer als der eines zentralisierten Systems.

- Warum also VS?
- Lösung: Vorteile eines Rechnerverbunds

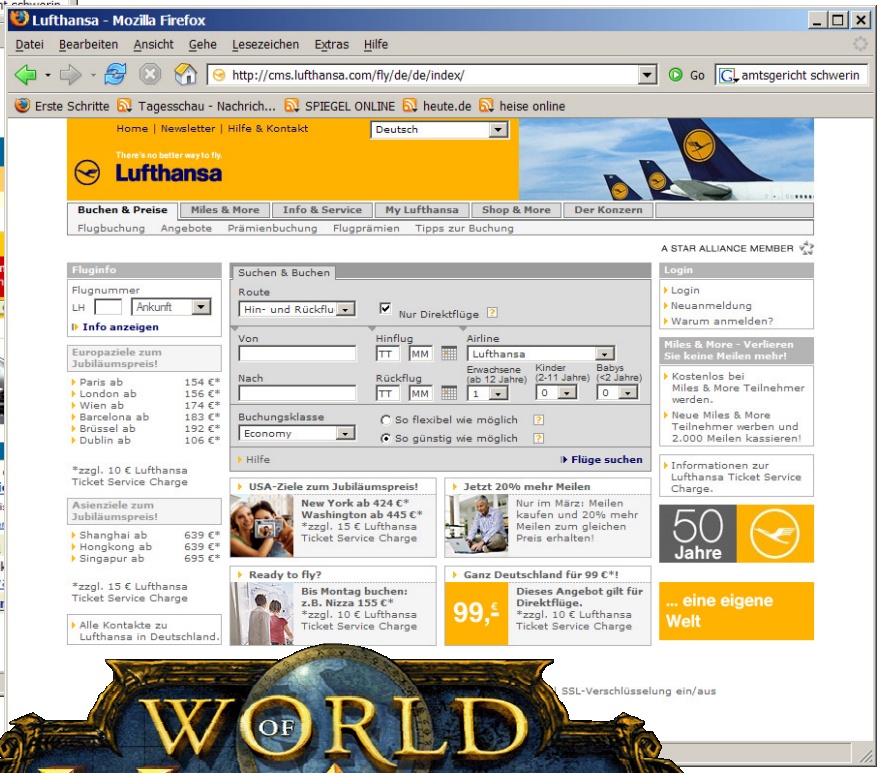
Rechnerverbünde (I)

- Kommunikationsverbund
  - Übertragung von Daten, insbesondere Nachrichten, an verschiedene, räumlich getrennte Stellen
  - Beispiel: Email
- Informationsverbund
  - Verbreiten von Information an interessierte Personen
  - Beispiel: WWW
- Datenverbund
  - Speicherung von Daten an verschiedenen Stellen
  - Bessere Speicherauslastung, erhöhte Verfügbarkeit, erhöhte Sicherheit
- Lastverbund
  - Aufteilung stoßweise anfallender Lasten auf verschiedene Rechner
  - Gleichmäßige Auslastung verschiedener Ressourcen

Rechnerverbünde (II)

- Leistungsverbund
  - Aufteilung einer Aufgabe in Teilaufgaben
  - Verringerte Antwortzeiten
- Wartungsverbund
  - Zentrale Störungserkennung und –behebung
  - Schnellere und billigere Wartung verschiedener Rechner
- Funktionsverbund
  - Verteilung spezieller Aufgaben auf spezielle Rechner (Superrechner, auch: Embedded Systems)
  - Bereitstellung versch. Funktionen an versch. Orten

Verteilte Anwendungen heute



INSTITUT FÜR BETRIEBSSYSTEME UND RECHNERVERBUND
Prof. Dr.-Ing. L. Wolf | Prof. Dr. S. Fekete
IBR

Wünschenswerte Eigenschaften

- Gut funktionierende und akzeptierte VS haben eine Reihe wichtiger Eigenschaften:
  - Offenheit
  - Nebenläufigkeit
  - Skalierbarkeit
  - Sicherheit
  - Fehlertoleranz
  - Transparenz
- Betrachten wir diese Eigenschaften etwas genauer.

Offenheit

- Bestimmt, wie gut sich das System auf verschiedenen Wegen erweitern lässt
- Existierende Dienste sollten dabei nicht unterbrochen werden
- Dies wird erreicht durch eine Veröffentlichung der Schnittstellen
- Beispiel: UNIX ist ein offenes Betriebssystem. Es umfasst die Programmiersprache C und kann durch den Zugriff auf Systemaufrufe (system calls) erweitert werden.
- Schnittstellen: Sockets, Middleware

Nebenläufigkeit (Concurrency)

- Mehrere gleichzeitig existierende Prozesse innerhalb eines Systems
- Gibt es nur einen Prozessor, erfolgt die Ausführung durch *Interleaving*.
- Bei n Prozessoren können die Prozesse parallel ausgeführt werden.
- Nebenläufigkeit kann es bei Clients (Anwendungsprogramme) und Servern (Zugriff auf Ressourcen) geben.
- Wichtiges Thema: Synchronisation

Skalierbarkeit

- Algorithmen, Protokolle und Prozeduren, die mit einigen wenigen Systemkomponenten gut funktionieren, sollen auch mit vielen Komponenten gut funktionieren.

- Das ist zum Teil schwieriger, zum Teil leichter als in zentralisierten Systemen
  - Höhere Komplexität
  - Ressourcen können beliebig hinzugefügt werden
- Beispiele für nicht skalierende zentrale Ansätze
  - Zentraler Server für einen bestimmten Dienst
  - Online-Telefonbuch in einer einzigen Datenbank

Sicherheit

- Sicherheit in VS hat viele Aspekte:
  - Vertraulichkeit (confidentiality): Daten können nur von dem gewünschten Empfänger gelesen werden
  - Integrität: Die Daten wurden während der Übertragung nicht verändert.
  - Authentizität: Die Daten wurden tatsächlich von der Person gesendet, die behauptet, der Sender zu sein.
- Sicherheit gehört heute zu den wichtigsten Themen bei der Entwicklung von VS, da
  - Oft Geld im Spiel ist
  - Oft persönliche Daten übertragen werden
- Sicherheit werden wir hier nur relativ kurz behandeln. Für Interessierte gibt es im Sommer eine eigene Vorlesung.

Fehlertoleranz

- In einem verteilten System können sehr viel mehr Fehler auftreten als in einem zentralisierten.
- Ein System ist gut, wenn diese Fehler abgefangen werden.
- Typische Fehlannahmen:
  - Das Netzwerk ist zuverlässig.
  - Das Netzwerk ist sicher.
  - Das Netzwerk ist homogen.
  - Die Topologie ändert sich nicht.
  - Die Latenzzeit beträgt null.
  - Die Bandbreite ist unbegrenzt.
  - Die Übertragungskosten betragen null.
  - Es gibt genau einen Administrator.
- Was passiert, wenn diese Annahmen nicht zutreffen?

Transparenz

- Transparenz hat viele Aspekte.
- Generell verstehen wir darunter, dass die Benutzer eines VS sich nicht zu sehr der Tatsache bewusst sind, dass es sich um ein VS handelt.
- Vielmehr wird das System als ein einheitliches System wahrgenommen.

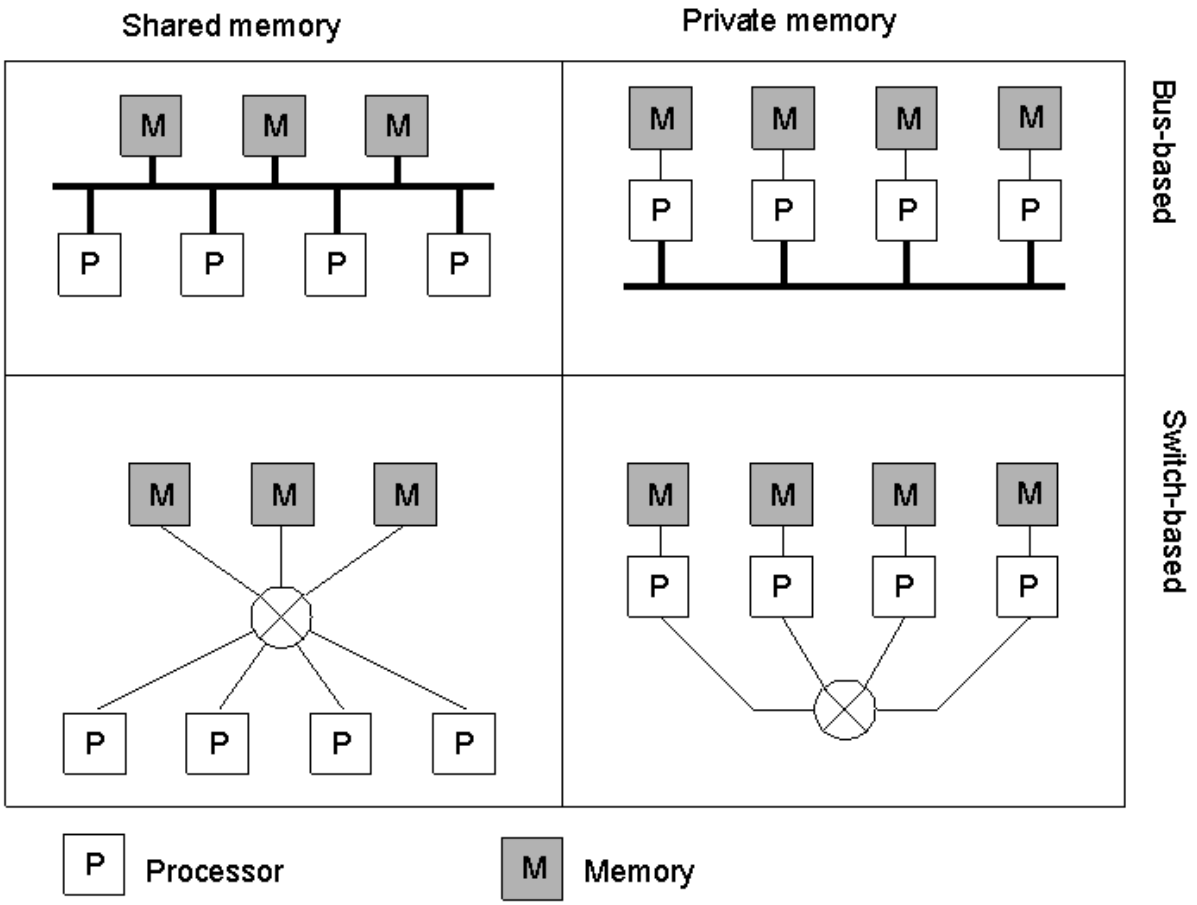
Transparenztypen

Transparenz	Beschreibung
Zugriff	Verbirgt Unterschiede in der Datendarstellung und die Art und Weise, wie auf eine Ressource zugegriffen wird
Ort[1]	Verbirgt, wo sich eine Ressource befindet
Migration	Verbirgt, dass eine Ressource an einen anderen Ort verschoben werden kann
Relokation	Verbirgt, dass eine Ressource an einen anderen Ort verschoben werden kann, während sie genutzt wird
Replikation	Verbirgt, dass eine Ressource repliziert ist
Nebenläufigkeit	Verbirgt, dass eine Ressource von mehreren konkurrierenden Benutzern gleichzeitig genutzt werden kann
Fehler	Verbirgt den Ausfall und die Wiederherstellung einer Ressource

Hardware-Konzepte

- Verteilte Systeme laufen auf Rechnern mit **mehreren CPUs**.
- Üblicherweise werden die Systeme in zwei große Gruppen unterteilt, die den **Grad der Kopplung** widerspiegeln:
  - **Multiprozessorsysteme** sind eng gekoppelt über gemeinsamen Speicher.
  - **Multicomputer** sind lose gekoppelt und kommunizieren über Nachrichten.
- Zusätzliche Einteilungen:
  - Kommunikationsverbindung: Broadcast oder Switch
  - Homogene oder heterogene Rechner

Grundlegende Hardware-Organisationsformen



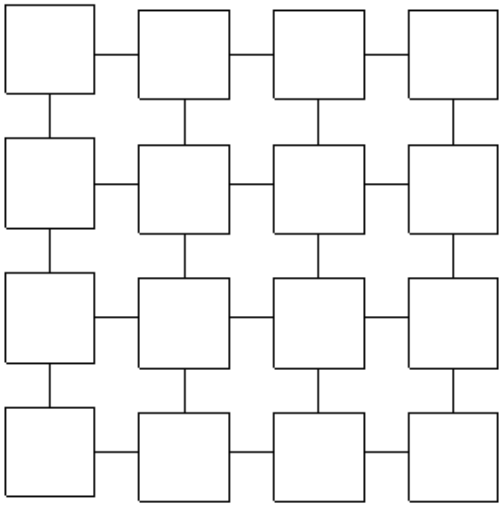
Multiprozessorsysteme

- Typische Vertreter dieser Art: Parallelrechner
- In den 80ern und Anfang der 90er sehr populär
- Problem: **teuer** durch die Spezialarchitektur
- Praktisch alle Firmen, die sich auf diese Art von Rechnern spezialisiert hatten, sind heute pleite
- Größere Firmen wie IBM oder NEC produzieren jedoch noch solche Supercomputer, s. auch http://www.top500.org/

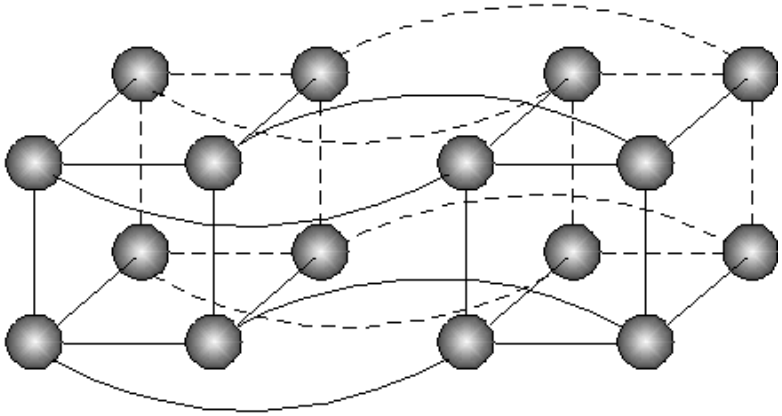
Multicomputer

- Unterscheidung
  - Homogene Multicomputer mit gleichen CPUs bzw. Architekturen
    - Beispiel: Cluster of HP Workstations, Massively Parallel Processors
  - Heterogene Multicomputer mit sehr unterschiedlichen Architekturen
    - Verteilte Anwendungen im Internet

Architekturen homogener Multicomputer



(a)



(b)

Softwarekonzepte

- Verteilte Betriebssysteme (Distributed Operating System, DOS)
- Netzwerkbetriebssysteme (Network Operating System, NOS)
- Middleware

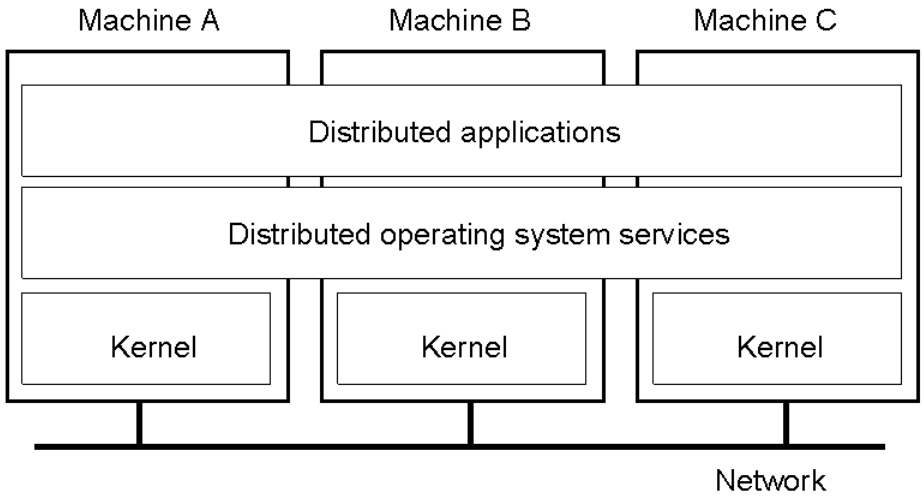
System	Description	Main Goal
DOS	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

Verteilte Betriebssysteme

- Wichtige Unterscheidung zwischen OS für Multiprozessorsysteme und für Multicomputer
- Multiprozessorsysteme benötigen nur einfache Erweiterungen des Uniprozessorbetriebssystems
  - Vor allem zur Handhabung des Shared Memory
  - Semaphore, Monitore spielen eine wichtige Rolle
- Multicomputerbetriebssysteme benötigen eine andere Architektur

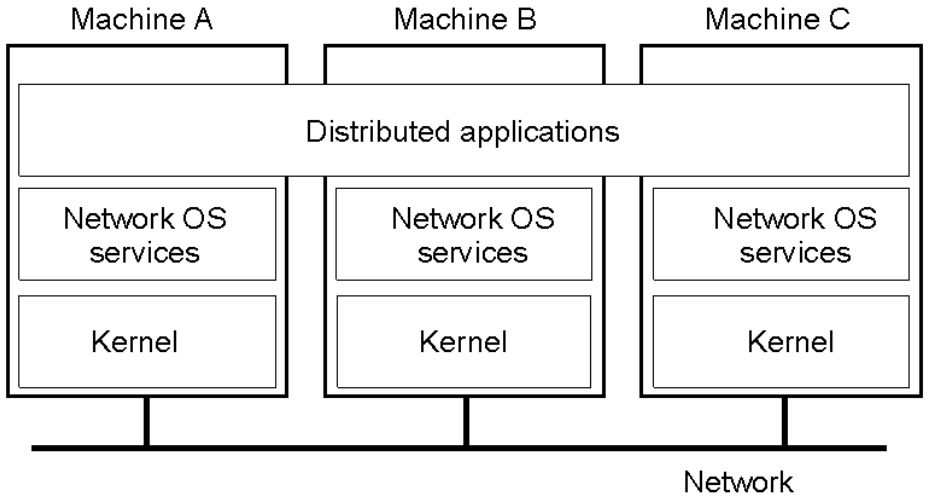
Verteiltes Multicomputerbetriebssystem

- Wichtigster Unterschied: Nachrichtenaustausch wird notwendig
- Lokale Mechanismen implementieren systemweit verfügbare Dienste
- Z.B. Realisierung eines virtuellen gemeinsamen Speichers
- Vor allem sinnvoll (weil leichter) in homogenen Systemen
- Transparenz ist gegeben – Benutzer müssen z.B. keine Rechnernamen kennen
- Beispiel: Plan 9 from Bell Labs



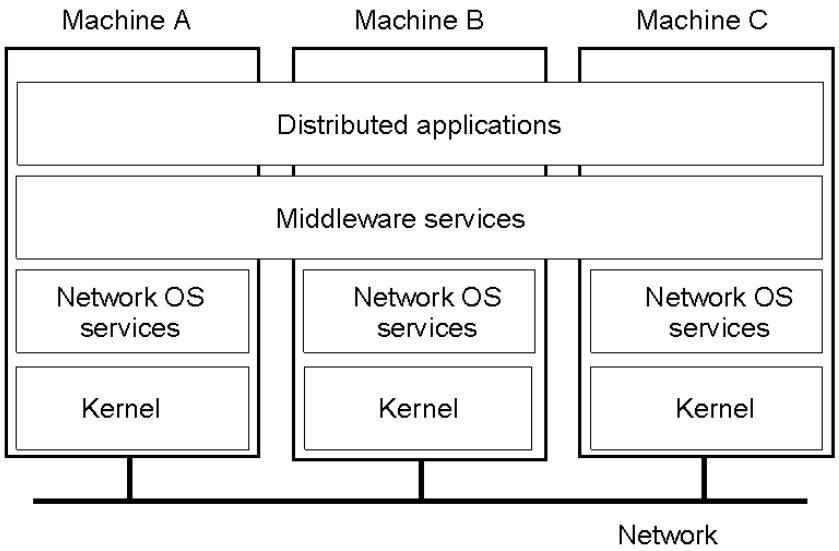
Netzwerkbetriebssysteme

- Sinnvoll vor allem in heterogenen Systemen
- Benutzer verwenden explizit Services auf anderen Rechnern
  - Beispiele: rlogin, ssh etc.
- Wenig Transparenz, aber dafür hohe Flexibilität



Middleware

- Middleware kombiniert Vorteile des DOS (Transparenz, leichte Nutzbarkeit) mit denen des NOS (Skalierbarkeit, Offenheit)
- Einfügen einer weiteren Softwareschicht ins NOS, die die Heterogenität des Systems verbirgt
- Modelle: verteilte Dateisysteme, RPC, verteilte Objekte, verteilte Dokumentsysteme
- Dieser Kurs behandelt die meisten dieser Ansätze.



Vergleich zwischen den Systemen

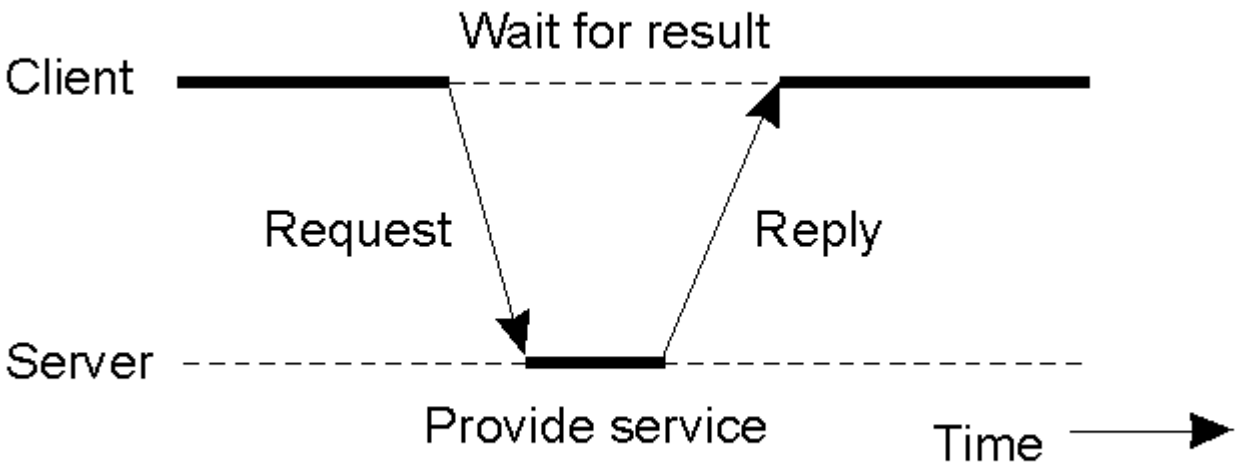
Item	Distributed OS		Network OS	Middleware-based OS
	Multiproc.	Multicomp.		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

Interaktionsmodelle

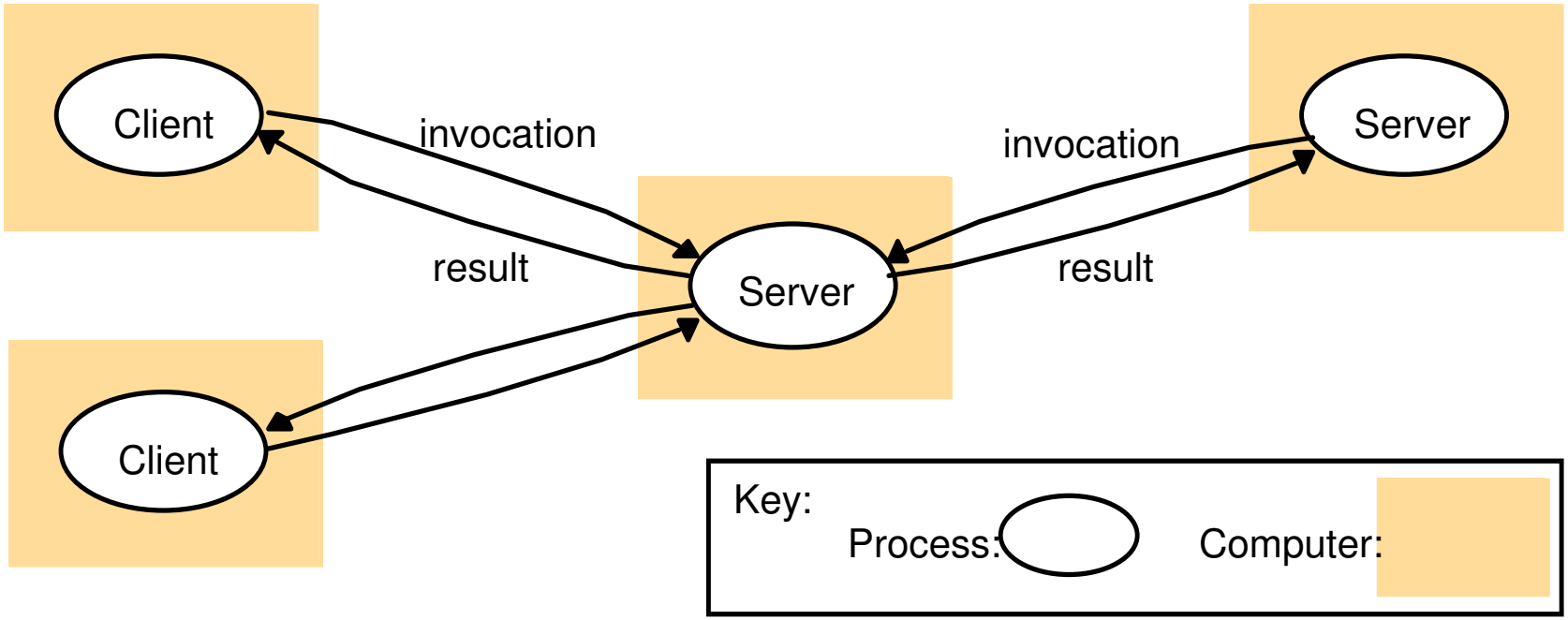
- Es gibt unterschiedlichste Möglichkeiten, wie die Softwarekomponenten auf den Rechnern miteinander interagieren, um ein Anwendungsproblem zu lösen.
- In den letzten Jahren haben sich dazu einige grundlegende Modelle herausgebildet:
  - Client-Server
  - Multi-Server
  - Service-Oriented Architecture (SOA)
  - Multi-Tiered
  - Peer-to-Peer
  - Grid Computing

Client-Server

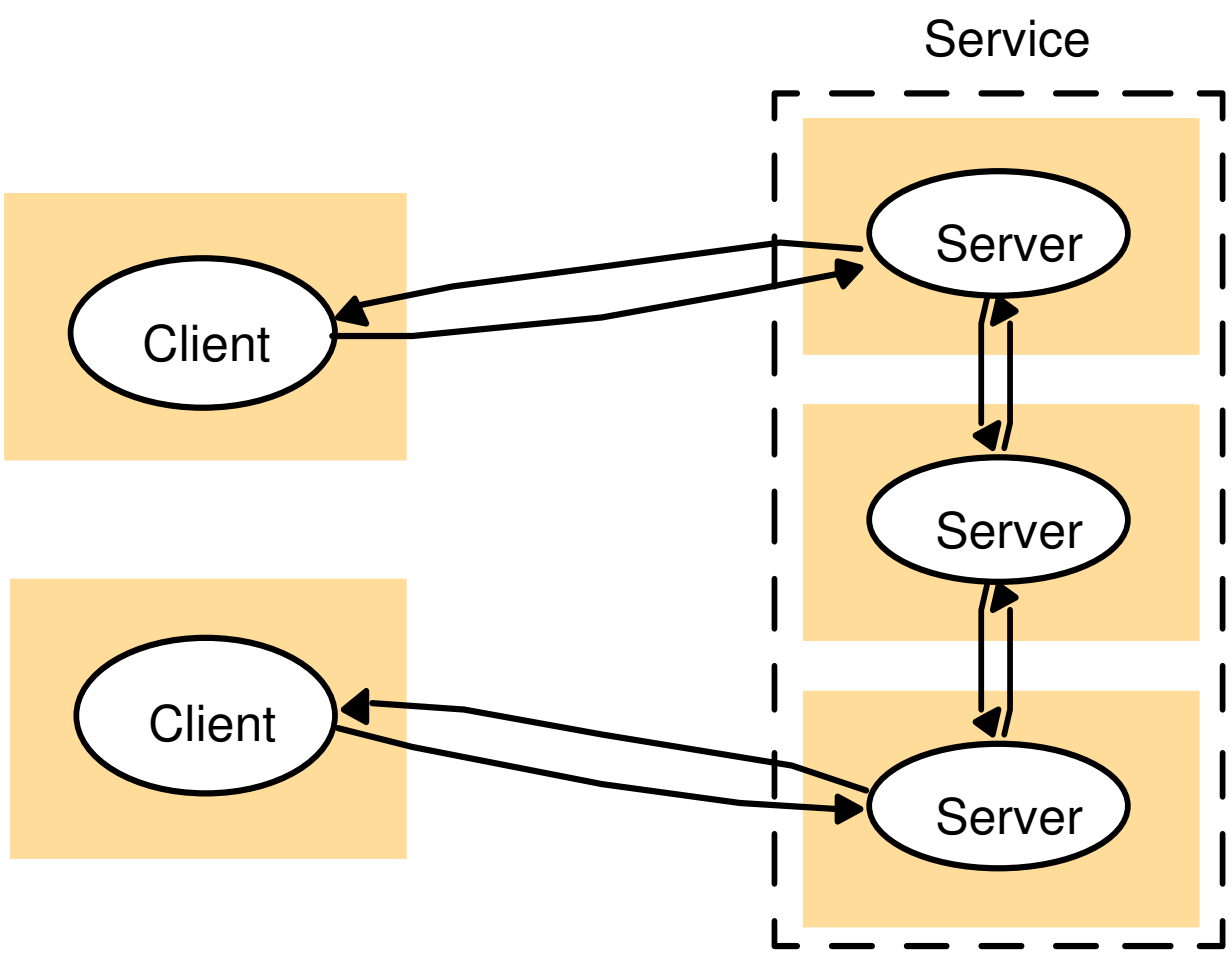
- Grundlegender Ablauf:
  - Client fordert einen bestimmten Service an
  - Server erhält die Anfrage, bearbeitet sie udn schickt ein Ergebnis zurück



Client-Server-Interaktionsmodelle



Mehrfache Server



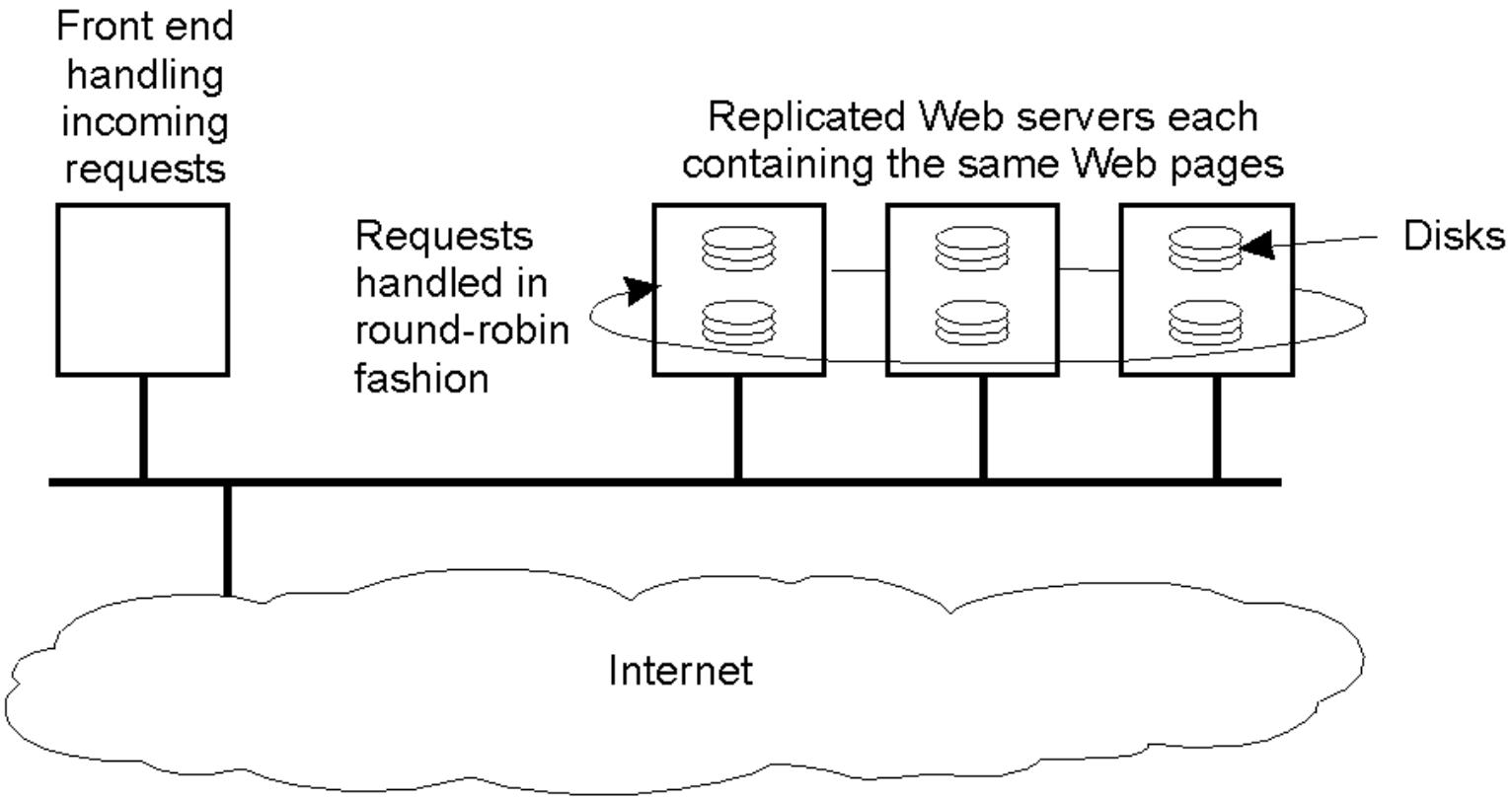
Partition oder Replikation von Diensten

Beispiel Partition: WWW

Beispiel für Replikation: WWW-Server-Farmen

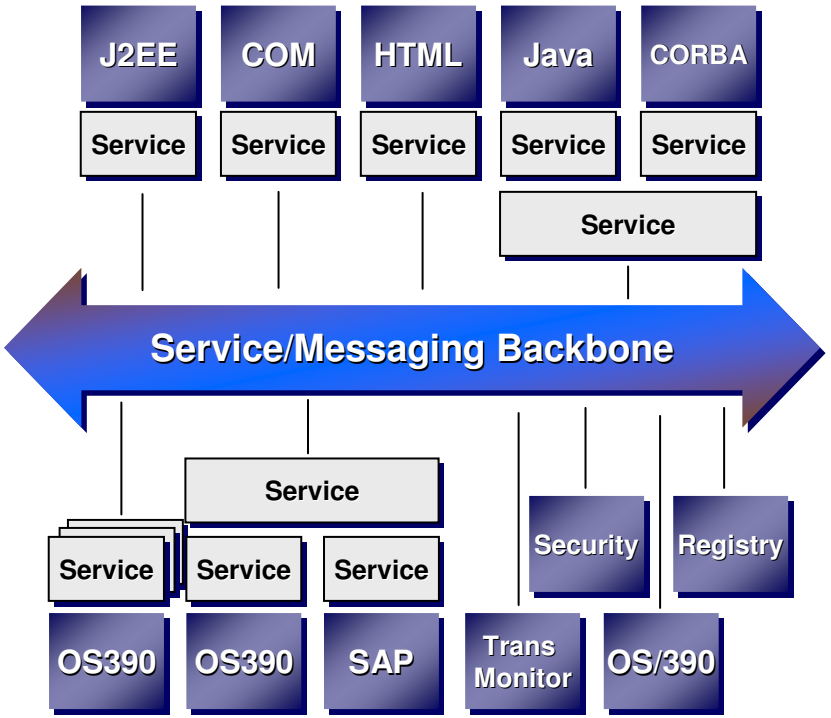
Beispiel für Replikation und Partition: DNS

Beispiel für Replikation



Service-Oriented Architectures

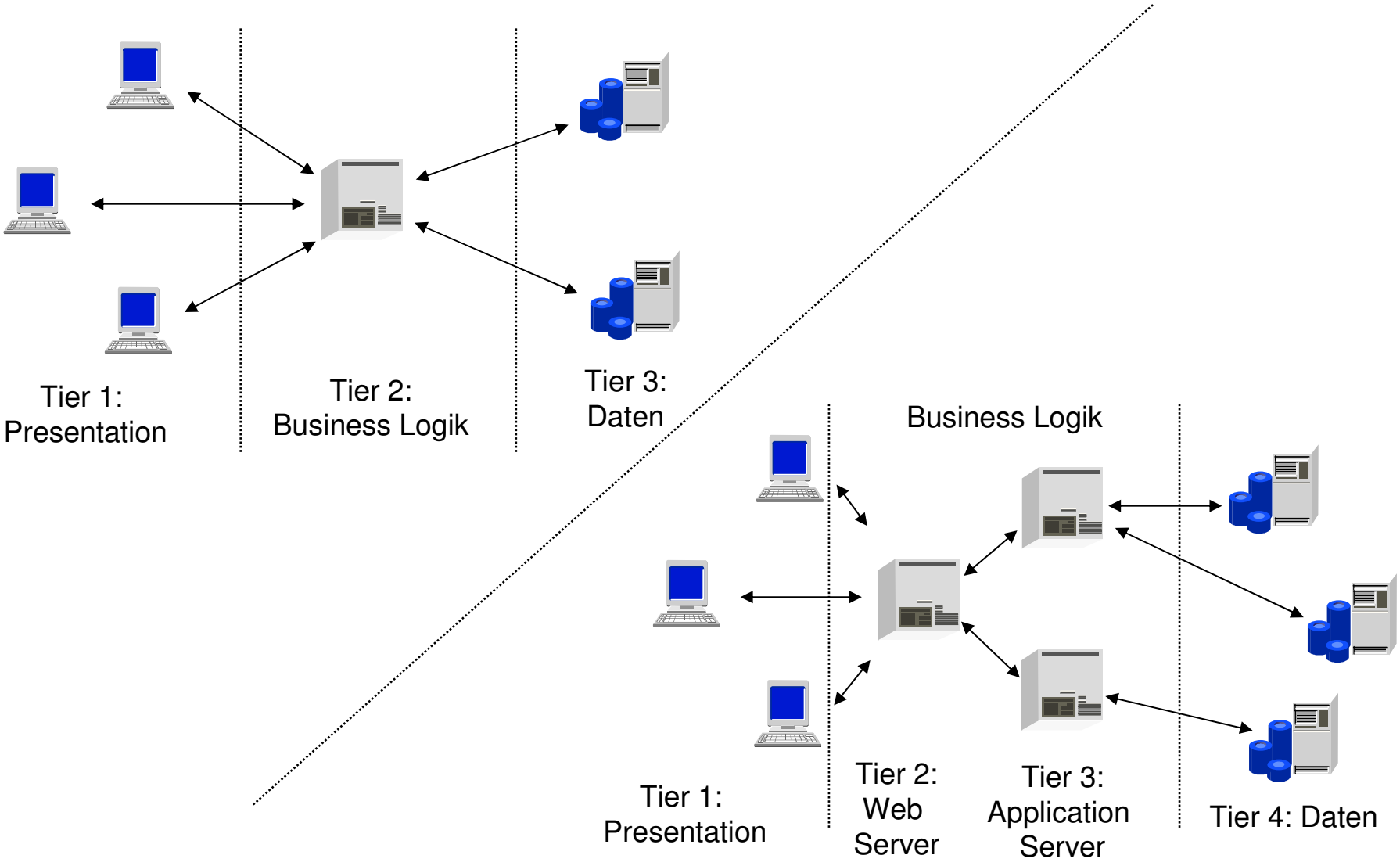
- Konsequente Weiterführung des Client-Server-Prinzips
- Service – eine Softwarekomponente mit einer formal beschriebenen Schnittstelle
- Gibt Zugang zu Anwendungslogik bzw. -komponenten
- Kommuniziert durch Anfragen und Antworten (synchron und asynchron)
- Typischerweise realisiert auf verteilten Plattformen wie COM, CORBA, J2EE oder MQSeries
- Web Services sind eine wichtige Grundlage von SOAs



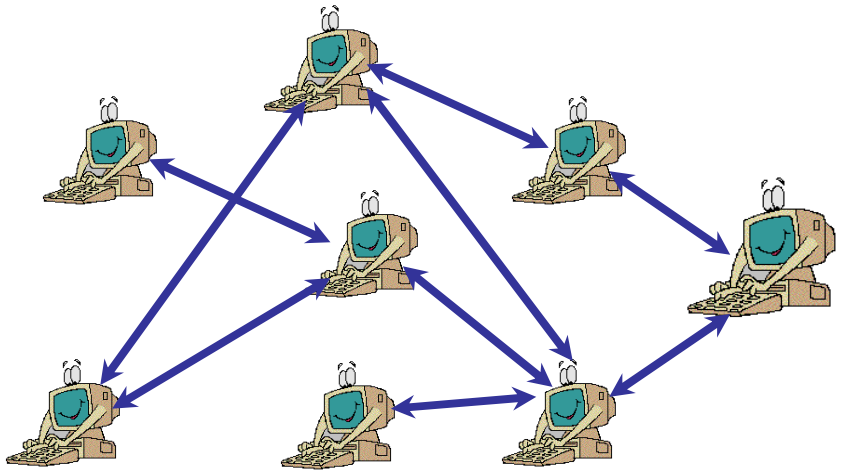
Multi-Tier Architekturen

- Verteilte (Web-)Anwendungen werden heute oftmals als Multi-Tier-Anwendungen entwickelt
- Jedes „Tier“ (Layer, Ebene) hat seine eigene Aufgabe
- Vorteile
  - geringere Komplexität der einzelnen Teile
  - Verteilung der Implementierungsaufgaben
  - Flexibilität bei der Verteilung der einzelnen Aufgaben (thin client)
  - erleichterte Wartbarkeit (keine Client-Software, Austausch von Versionen)
  - Skalierbarkeit, Sicherheit
- Prinzipiell eine Weiterführung des Client-Server-Prinzips

3- und 4-Tier-Architekturen



Definition eines Peer-to-Peer Systems

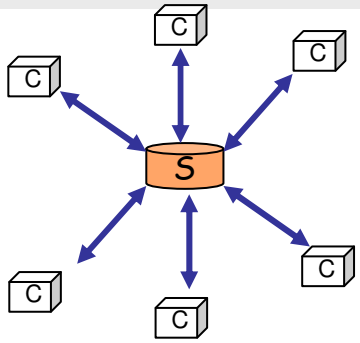


„P2P is a class of applications that takes advantage of resources storage, cycles, content, human presence – available at the edge of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have significant or total autonomy from central servers.“ (Clay Shirky, openp2p.com)

Klassifikation von Peer-to-Peer Systemen

Client-Server systems

- Classical role-based systems
- No interaction between clients



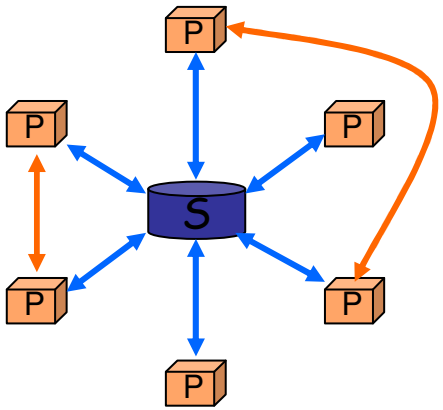
Examples:

WWW

DNS

Hybrid P2P systems

- Joint usage of distributed resources
- Direct interaction between peers
- Usage of servers for coordination



Napster

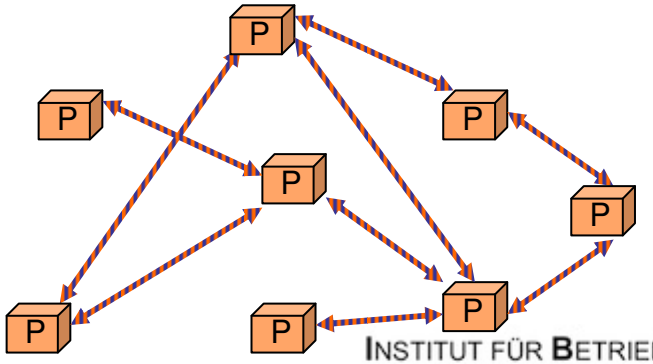
ICQ

AIM (AOL)

Seti@Home

Pure P2P systems

- Completely de-centralized management *and* usage of resources



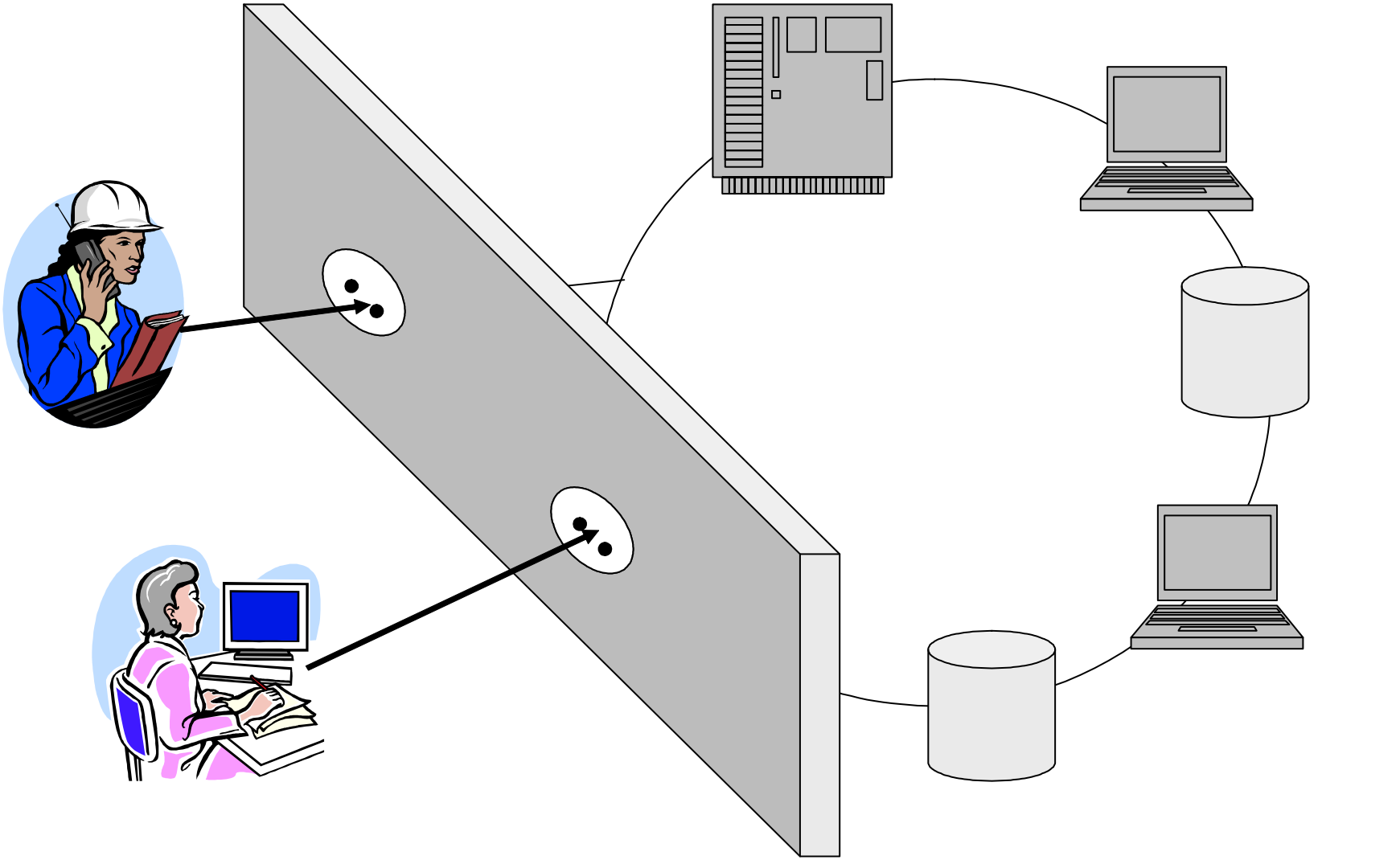
Gnutella

Freenet

Grid Computing

- Grid Computing ist eigentlich eine alte Idee, wird aber zum ersten Mal konsequent umgesetzt.
- Ziel ist es, eine große Menge von Computern als ein einziges System darzustellen:
  - Alle Ressourcen sind nutzbar
  - Ohne dass der Benutzer wissen muss, wo diese Ressourcen liegen

Grid-Architektur



Zusammenfassung

- Verteilte Systeme bieten gegenüber zentralen Systemen einige Vorteile (→ Verbünde).
- Ihre Implementierung ist jedoch auch komplexer.
- Es hat sich eine Vielfalt von Architekturen für verteilte Systeme entwickelt.
- Jede Architektur hat ihre Nische gefunden und wird eingesetzt.
- Es ist eine Vielzahl von Algorithmen und Protokollen notwendig, um die volle Funktionalität Verteilter Systeme nutzen zu können.
- Die wichtigsten werden wir im Laufe dieser Veranstaltung kennenlernen.

Diskussion