TU Braunschweig
Institut für Betriebssysteme
und Rechnerverbund

Verteilte Systeme

Prof. Dr. Stefan Fischer

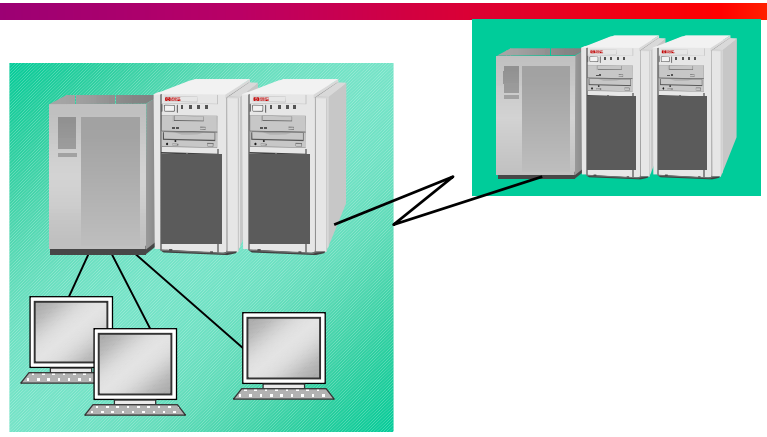
Kapitel 1: Einführung

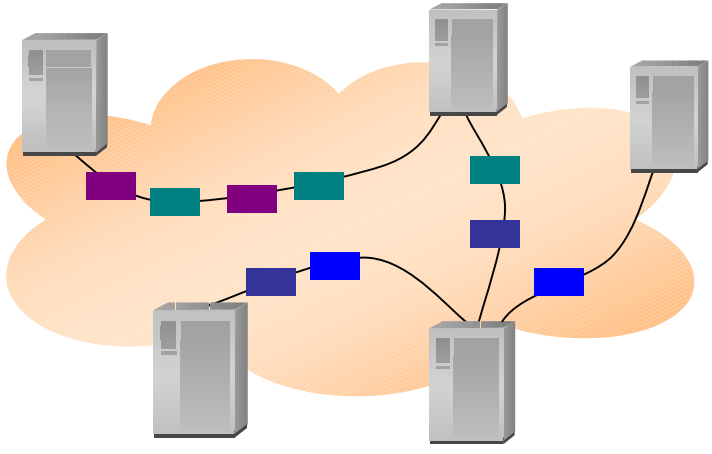
Überblick

- Eine kurze Geschichte der Netze und verteilten Systeme
- Definition des Begriffs „Verteilte Systeme“
- Beispiele für verteilte Systeme
- Gründe für die Nutzung verteilter Systeme
- Wünschenswerte Eigenschaften verteilter Systeme

Computernetzwerke: 1960s-1970s

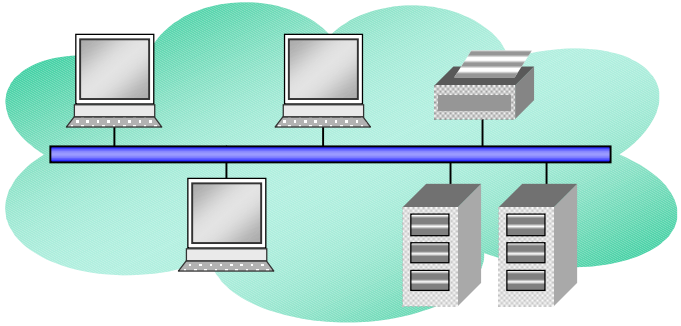
- Terminal-Host-Kommunikation über serielle Leitungen
- Host-zu-Host-Kommunikation
  - Basierend auf proprietären Netzwerken wie IBM SNA, DECnet
  - schon bald basierend auf Paketvermittlung und damit TCP/IP-Netzen

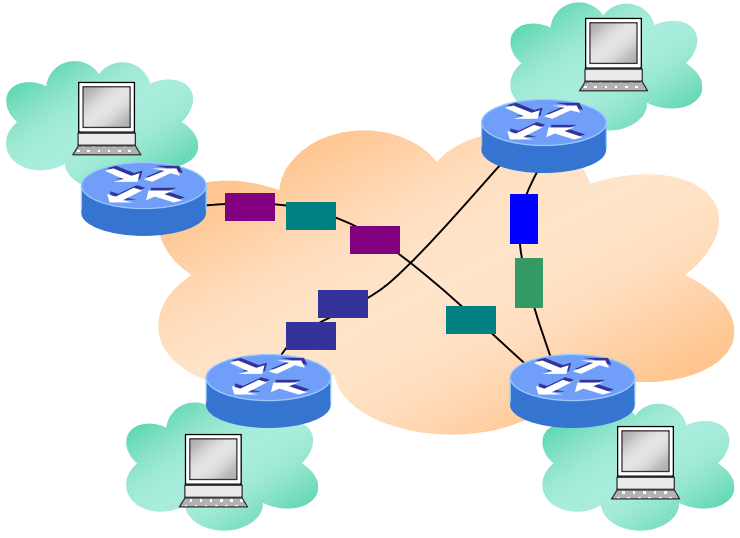




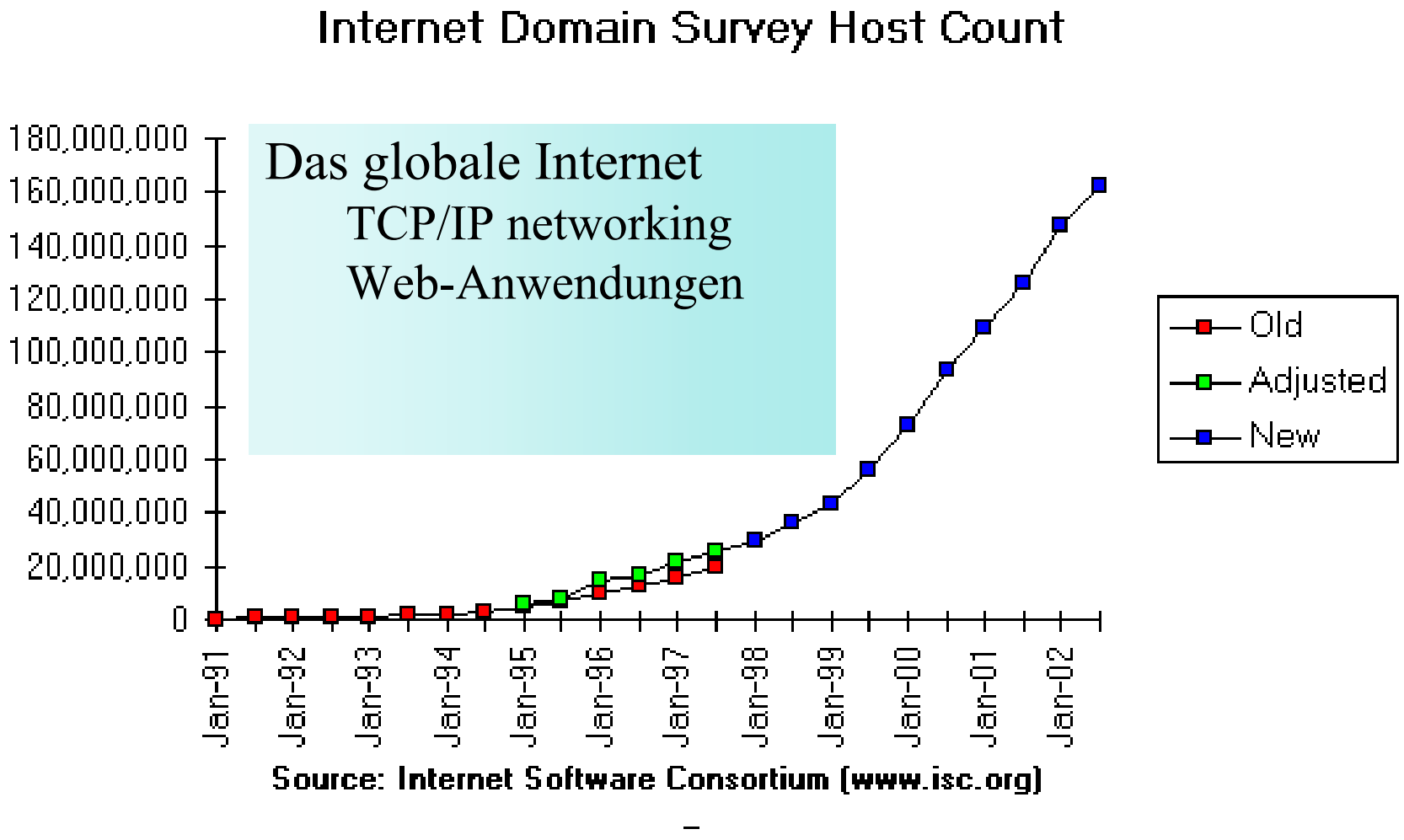
Computernetzwerke: 1980s

- 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)





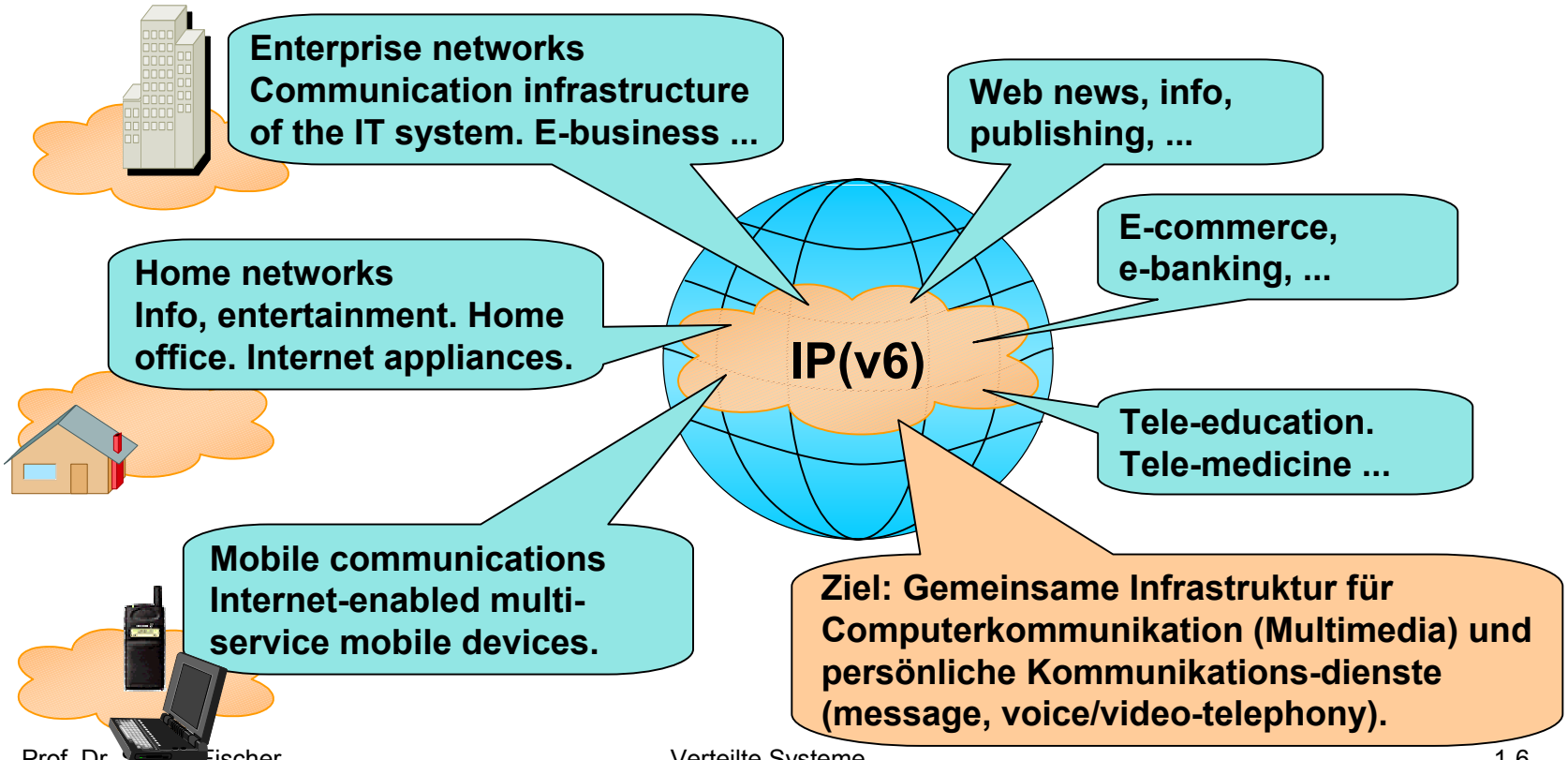
Computernetzwerke: 1990s



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.

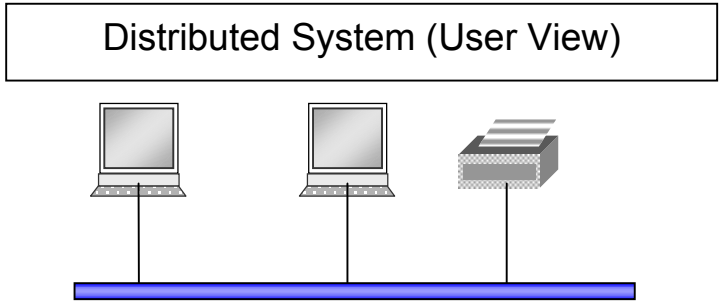
- Was also ist ein verteiltes System?

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.

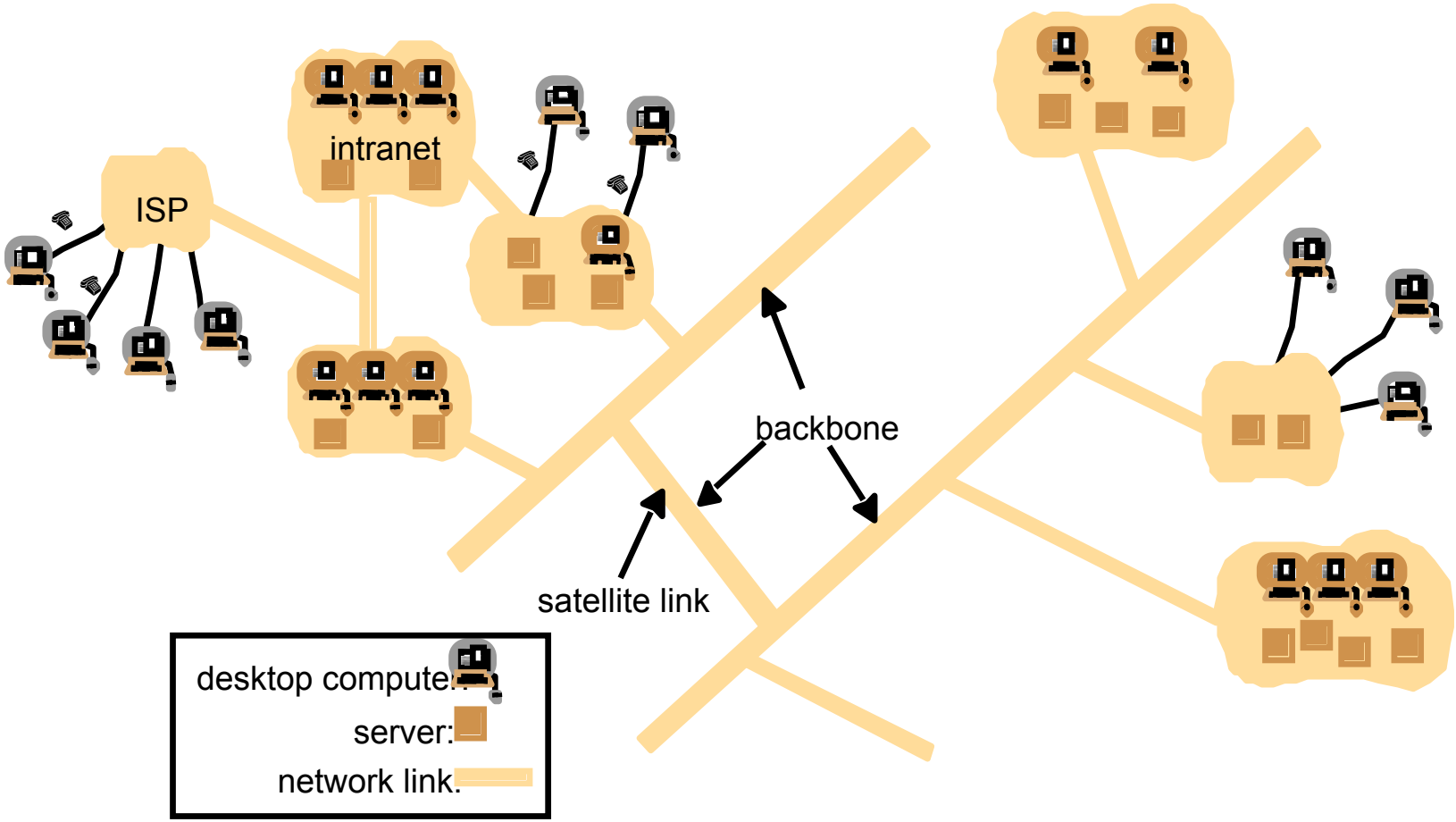


Was ist ein verteiltes System?

- 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.

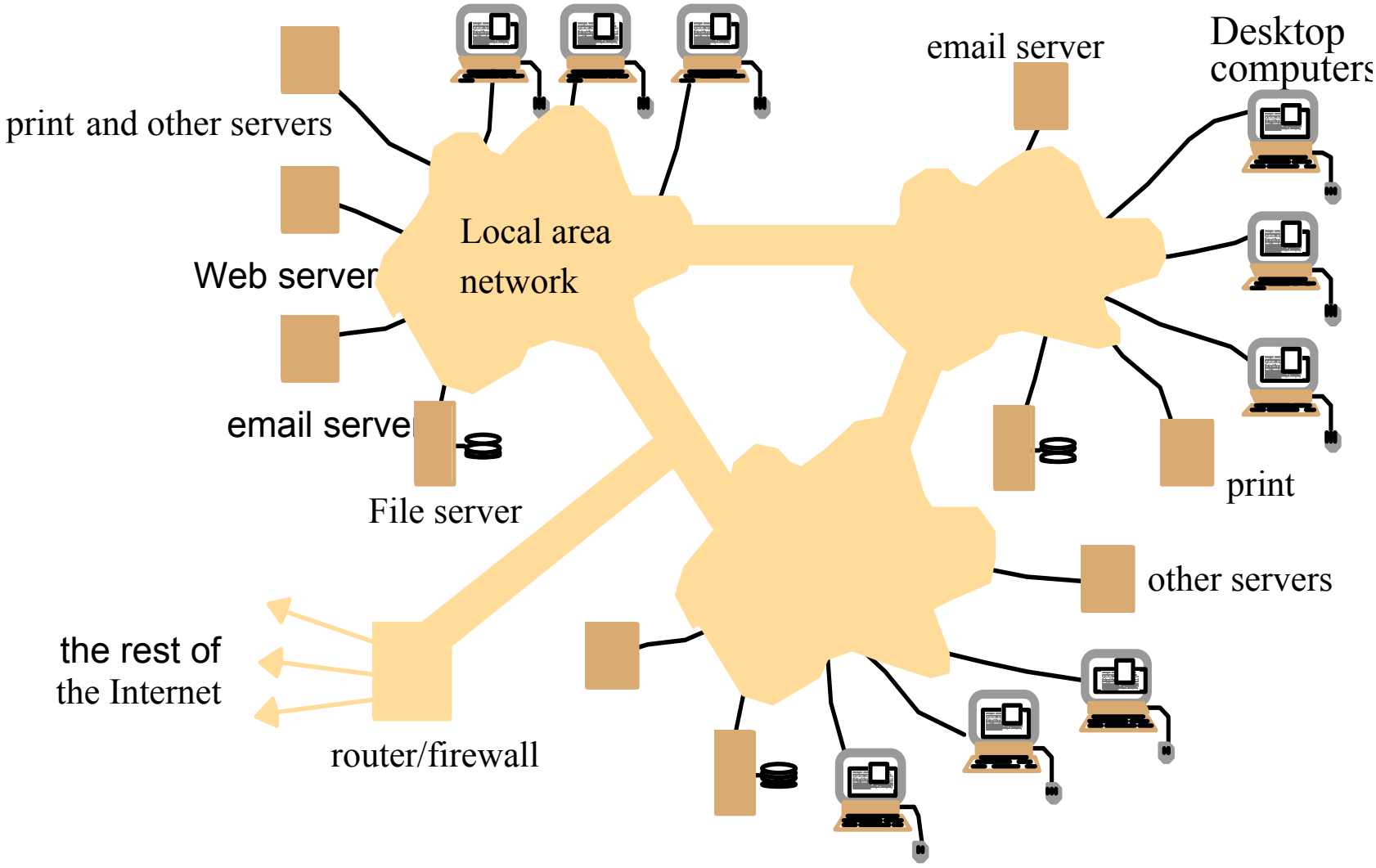
Beispiel #1: Das Internet



Das Internet

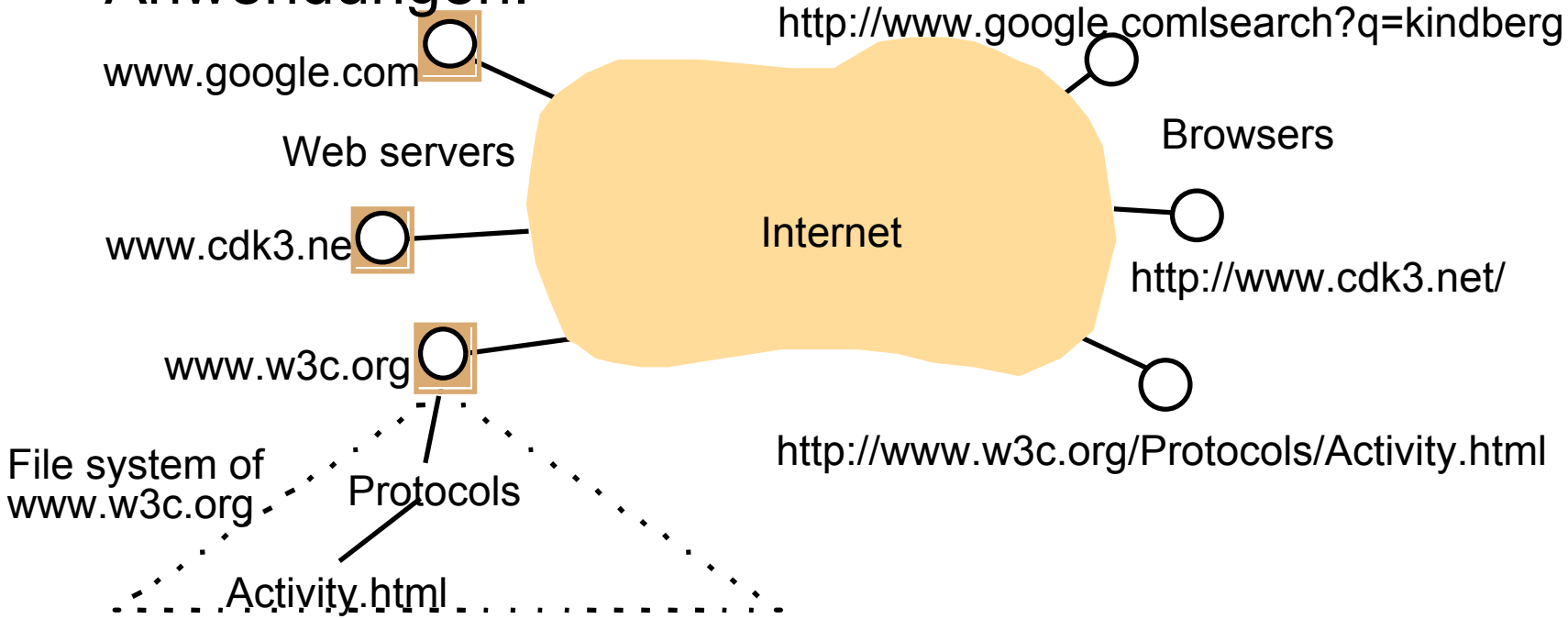
- Erinnerung: In ISO-Terminologie, ist ein „*internet“ (englisch)* oder „*internetwork“ eine Sammlung physikalischer Netzwerke, die miteinander verbunden sind.*
- „Das“ Internet ist ein gigantisches „internetwork“, das sich durch die einheitliche Verwendung der TCP/IP-Protokollfamilie auszeichnet.
- Im Internet gibt es eine große Zahl verteilter Anwendungen
  - Standardisierte Anwendungen wie FTP, Email, WWW etc.
  - Proprietäre Anwendungen, z.B. im Bereich E-Commerce: der Amazon-Buchladen, das Flugbuchungssystem von Lufthansa, etc.
- Wir kommen später noch detailliert darauf zu sprechen.

Beispiel #2: Ein Intranet



Beispiel #3: Das World Wide Web

- Sicherlich die populärste verteilte Anwendung.
- Basiert auf dem Internet als verteiltes System.
- Interessant: WWW als Basis für neue Anwendungen.



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

Rechnerverbunde (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

Rechnerverbunde (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 (Feldrechner, Superrechner, Transputer)
  - Bereitstellung versch. Funktionen an versch. Orten
- Kapazitätsverbund
  - Ausnutzung sämtlicher zur Verfügung stehender Rechenkapazität
  - Versendung von Aufgaben an möglichst viele verschiedene Rechner

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

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 diesen Winter eine eigene Vorlesung.

Fehlertoleranz

- Wie können Fehler in Computersystemen gelöst werden?
  - Hardware-Redundanz: Standby-Maschinen, doppelte Komponenten
  - Software Recovery: Rückkehr in einen sicheren Zustand, wenn ein Fehler entdeckt wird

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

Access transparency: enables local and remote resources to be accessed using identical operations.

Location transparency: enables resources to be accessed without knowledge of their location.

Concurrency transparency: enables several processes to operate concurrently using shared resources without interference between them.

Replication transparency: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.

Failure transparency: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.

Mobility transparency: allows the movement of resources and clients within a system without affecting the operation of users or programs.

Performance transparency: allows the system to be reconfigured to improve performance as loads vary.

Scaling transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms.