Verteilte Systeme

Organisation

Informatik

Theoretische Informatik

- Formale Sprachen
- Abstraktions-theorie

etc.

Grundlagen

Technische Informatik

- Rechner-technologie
- Netze

etc.

Hardware

Konstruktionslehre der Informatik

- Methoden u. Werkzeuge
- System- u. Anwendungs-komponenten
- Qualitäts-sicherung

etc.

Methodenlehre

Praktische Informatik

- Betriebs-systeme
- Datenbank-systeme
- Kommu-nikations-systeme

etc.

Basissoftware

Anwendungen der Informatik

- Kaufmännische Informations-systeme
- Technische Informations-systeme

etc.

Anwendungen

Abb2 in Informatik-Spektrum, Heft 25/1, S. 39-51, 2002: Erich Ortner, *Sprachingenieurwesen, Empfehlung zur inhaltlichen Weiterentwicklung der (Wirtschafts-)Informatik*

Anforderungen

- Betriebssysteme (BS/BSP)
  da in diesen Umgebungen die Verteilten Systeme arbeiten

- Rechnernetze (RN/RNP)
  da dies die notwendigen Basisstrukturen sind auf/in denen Verteilte Systeme ablaufen

- Kenntnisse der Lehrveranstaltungen des 1-ten bis 4 -ten Semesters
  da Verteilte Systeme *die Informatik* vereint

Vorlesungszyklus des Fachgebiets

Eigene Werke

Bachelor Projekt

WPs & POs

RIOT im IoT

Webarchitekturen

Anwendungen

Aktuelle Internettechnol.

Verteilte Systeme

Grundlagen

Rechnernetze

Betriebssysteme

Arbeitsgruppe Internet-Technologien

- Website:

  inet.haw-hamburg.de

- Sie finden uns in:
  - Raum 580 & Raum 480/1

- Messen & Ausstellungen:
  - CeBIT, Embedded World, NdW
- Auslandskooperationen
  - Wir vermitteln gerne

Nicht nur zum Scheine …

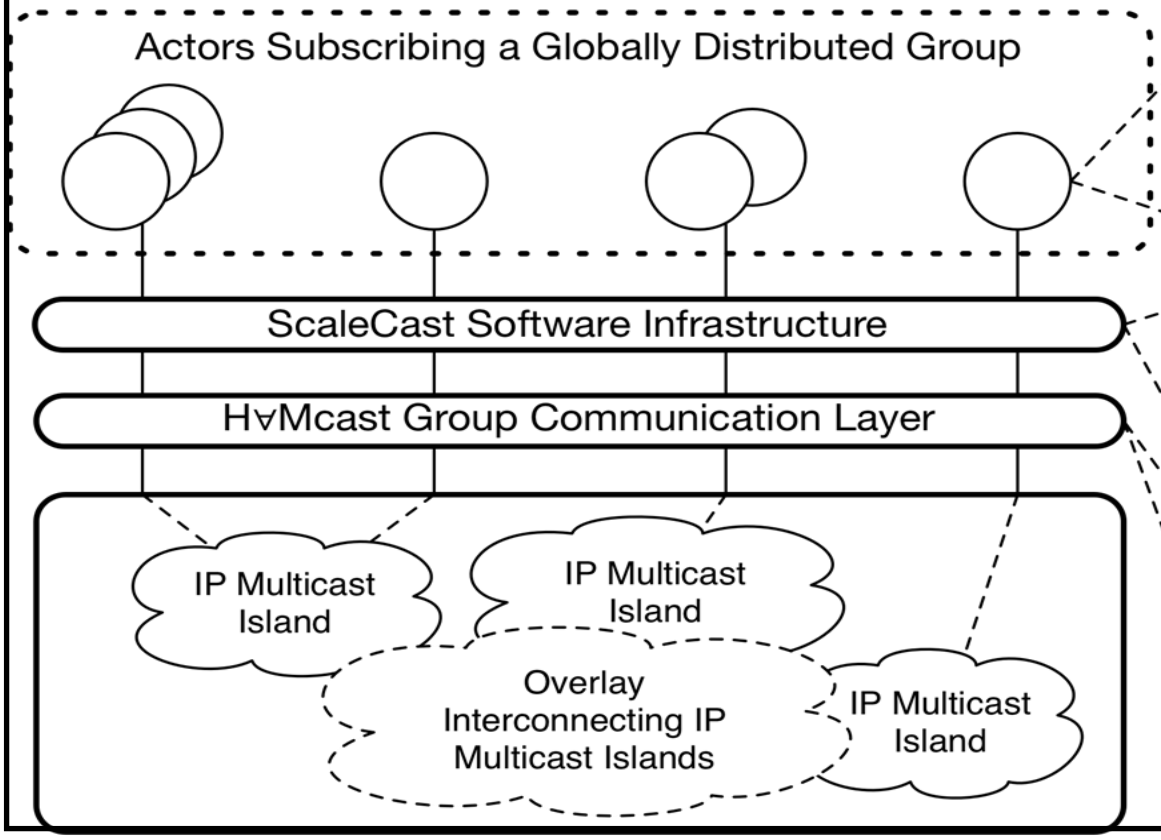
… machen ist die Hochschule da.

Wir wollen Ihnen die Chance auf mehr geben:

- In der Vorlesung: Über Zusammenhänge nachdenken und diskutieren!
- In einem der nachfolgenden Projekte: Mitarbeiten!

C++ Actor Framework - CAF: Scalable Distributed Programming

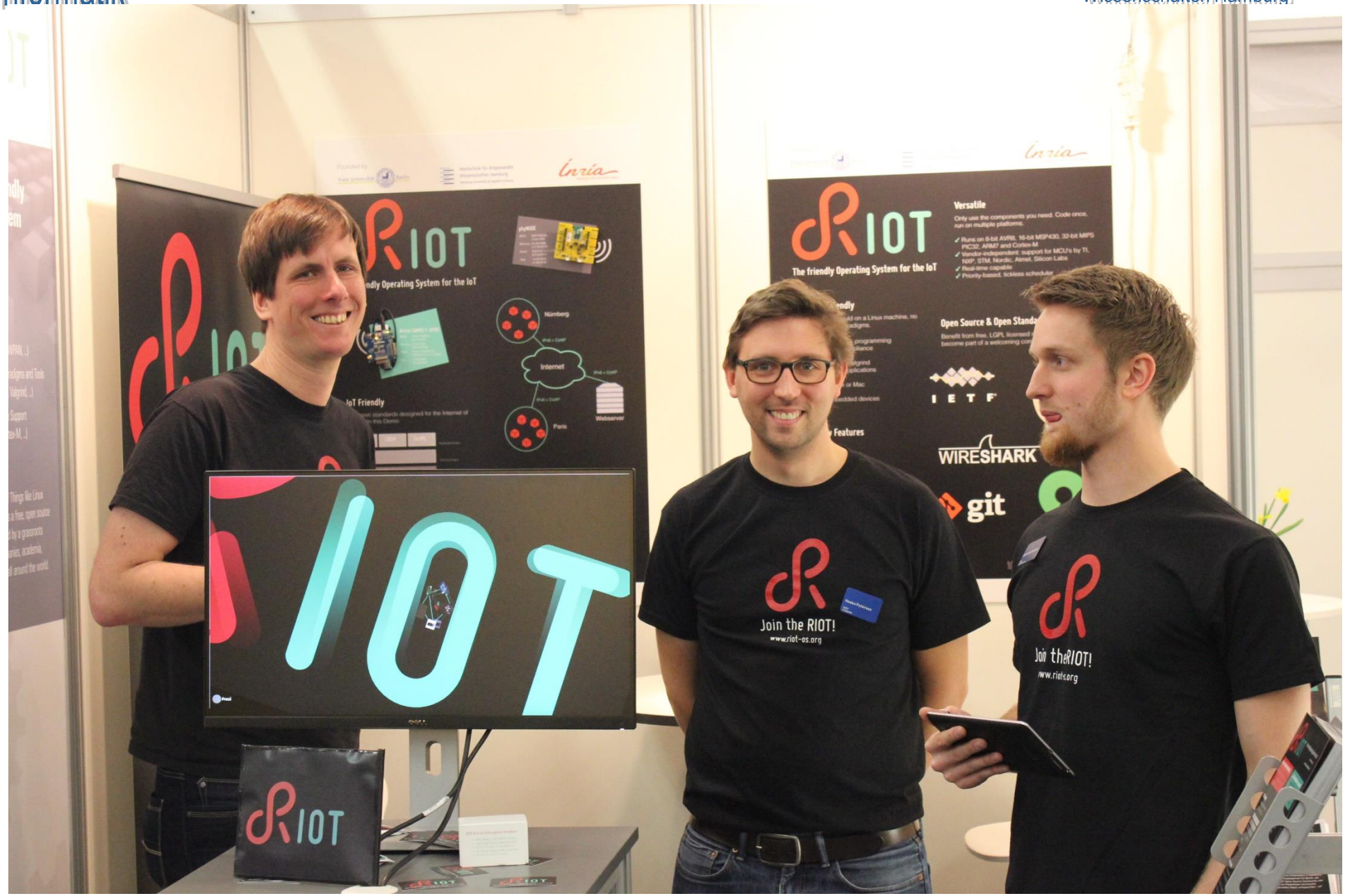
Globally Distributed Actors Using Publish/Subscribe




```
void my_actor() {
    join(group::get("hamcast",
                    "sip://news@cnn.com"));
    receive_loop (
        on<news>() >> [](const news& n) {
            // ...
        }
    );
}
```

- Schedules actors in a distributed system
- Provides network transparent messaging, group communication & failure detection

- Enables globally available, name-based Publish/Subscribe


RIOT
The friendly Operating System for the IoT
Versatile
Open Source & Open Standa
IETF
WIRESHARK
git
Join the RIOT!
www.riot-os.org
Join theRIOT!
www.riot.org
IOT
Internet
Nürnberg
Paris
Webserver

Basis-Literatur

- G. Coulouris, J. Dollimore, T. Kindberg: *Verteilte Systeme*, Pearson Studium
- Andrew S. Tanenbaum, Maarten van Steen: *Verteilte Systeme: Grundlagen und Paradigmen*, Pearson Studium

Zum Kauf empfohlen!

- U. Hammerschall: *Verteilte Systeme und Anwendungen,* Pearson Studium
- Claudia Eckert: *IT-Sicherheit,* Oldenbourg
- N. A. Lynch: *Distributed Algorithms*, Morgan Kaufmann
- U. Lang, R. Schreiner: Developing Secure Distributed Systems with CORBA, Artech House
- A. S. Tanenbaum: *Moderne Betriebssysteme*, Hanser Verlag
- A. S. Tanenbaum: *Computernetzwerke*, Prentice Hall
- D. Baum, M. Gasperi, R. Hempel, L. Villa: *Extreme Mindstorms: An Advanced Guide to Lego Mindstorms*, Apress Verlag

Praktikum / PVL

- 4 Praktikumsaufgaben
  - RMI: Client-Server Anwendung
  - Message Passing: Konzeption und Realisierung
  - Zeitgesteuerte Gruppenkommunikation
  - RIOT distributed sensing
- Gruppenaufteilung: **2 Studierende** in einer Gruppe
- **PVL-Bedingungen**
  - **Baut stark auf vorbereitender Arbeit** auf!
  - **Details** siehe Aufgabenstellungen!
    - Bis **Montag Abend vor** dem Praktikumstermin
    - Am **Anfang des Praktikumstermins:** Konzeptgespräch
    - Bis **Dienstag der Folgewoche**: Abgabe des Codes
    - Alle Abgaben an t.schmidt@haw-hamburg.de und frank.matthiesen@haw-hamburg.de senden.
  - Anwesenheitspflicht (gesamte Praktikumszeit!)
  - Erfolgreiche Bearbeitung **aller** Aufgaben: siehe Aufgabenstellungen

Hinweis zu den Folien

- Die Folien sind **kein vollständiges Skript** und genügen normalerweise nicht zur Prüfungsvorbereitung oder als Nachschlagewerk!

- Sie sollten sich deshalb auf jeden Fall zumindest mit der aufgeführten **Basis-Literatur beschäftigen**, insbesondere mit den zum Kauf empfohlenen Büchern, und sich von Zeit zu Zeit auch **weiterführende Literatur** und aktuelle **Zeitschriftenartikel anschauen**.

- Bemerkung am Rande: Diese Folien sind zum großen Teil aus Folien anderer Kollegen (auch anderer Hochschulen) zusammengestellt!

Verteilte Systeme

Einführung

Inhalt der Vorlesung

1. Einführung und Systemmodelle
2. Interprozesskommunikation (IP, RMI, Web, J2EE,+++)
3. Namensdienste und Internet-Standardanwendungen
4. C++ Message Passing & Actor Programming mit CAF
5. Zeit, Synchronisation und globale Systemzustände
6. Verteiltes Debugging
7. Übereinstimmung und Koordination
8. Peer-to-Peer Systeme
9. Verteilte Transaktionen
10. Replikation
11. Sicherheit in verteilten Systemen

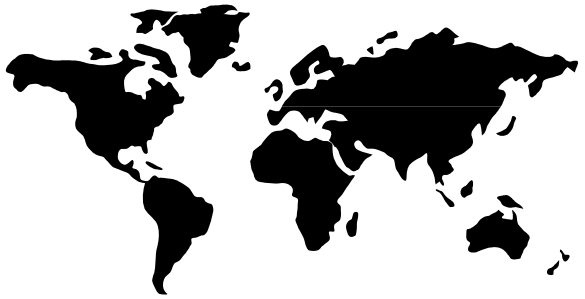
Warum bilden **„Verteilte Systeme“** ein **eigenständiges Thema ?**

- Es gibt **keinen gemeinsamen Speicher**
  (Interaktion durch Nachrichtenaustausch)
- Es gibt **nebenläufige/parallele Aktivitäten**
  (Koordination, Synchronisation)
- Fehler und **Ausfälle sind wahrscheinlich**
  (Transparenz)
- **Komponenten** (Hardware und Software) sind **heterogen**
  (Standardisierung von Schnittstellen)
- **Systeme** können **sehr groß** sein
  (Großsystemeffekte, Umschlag von der Quantität in die Qualität)
- **Sicherheit** der **Systeme** und auch **Dritter** kann durch die Verteilung gefährdet sein

Wozu braucht man ein „**Verteiltes System**“?

- **Kommunikations**verbund (Übertragung von Daten, insbesondere Nachrichten, an verschiedene, räumlich getrennte Stellen; z.B. E-Mail)
- **Informations**verbund (Verbreiten von Information an interessierte Personen/Systeme; z.B. WWW)
- **Daten**verbund (Speicherung von Daten an verschiedenen Stellen: bessere Speicherauslastung, erhöhte Verfügbarkeit, erhöhte Sicherheit)
- **Last**verbund (Aufteilung stoßweise anfallender Lasten auf verschiedene Rechner: gleichmäßige Auslastung verschiedener Ressourcen)
- **Leistungs**verbund (Aufteilung einer Aufgabe in Teilaufgaben: Verringerte Antwortzeiten)
- **Wartungs**verbund (Zentrale Störungserkennung und –behebung: schnellere und billigere Wartung verschiedener Rechner)
- **Funktions**verbund (Verteilung spezieller Aufgaben auf spezielle Rechner; Bereitstellung verschiedener Funktionen an verschiedenen Orten)
- **Kapazitäts**verbund (Ausnutzung sämtlicher zur Verfügung stehender Rechenkapazität)

Verteilte Welt und Probleme



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

>Räumliche Separation, autonome Komponenten

>Heterogenität

>Dynamik, Offenheit

>Komplexität

>Sicherheit

+Probleme sequentieller Systeme

+Nebenläufigkeit

+Nichtdeterminismus

–Synchronisation schwieriger

–Programmierung komplexer

–Testen

(Threads, kritische Abschnitte, etc.) gelernt. Insbesondere habe ich erkannt, dass das Debugging und die Fehlersuche in solchen Systemen ungleich schwerer als in anderen Systemen ist. Das liegt sicher zum einen daran, dass es auch deutlich mehr Fehlermöglichkeiten gibt. Hinzu kam aber noch meine Unerfahrenheit in diesem Bereic

Was ist ein verteiltes System ?

- Definition: **Verteiltes System** (nach *Leslie Lamport*)

  Ein verteiltes System ist ein System, mit dem ich nicht arbeiten kann, weil irgendein Rechner abgestürzt ist, von dem ich nicht einmal weiß, daß es ihn überhaupt gibt.

  - oft die Realität
  - wird aber besser (hoffentlich auch durch diese Vorlesung)
- Definition: **Verteiltes System** (nach *Andrew S. Tanenbaum*)

  Ein verteiltes System ist eine Kollektion unabhängiger Computer, die den Benutzern als ein Einzelcomputer erscheinen.

  - impliziert, daß die Computer miteinander verbunden sind und
  - die Ressourcen wie Hardware, Software und Daten gemeinsam benutzt werden.
  - Es herrscht eine einheitliche Sicht auf das Gesamtsystem vor.

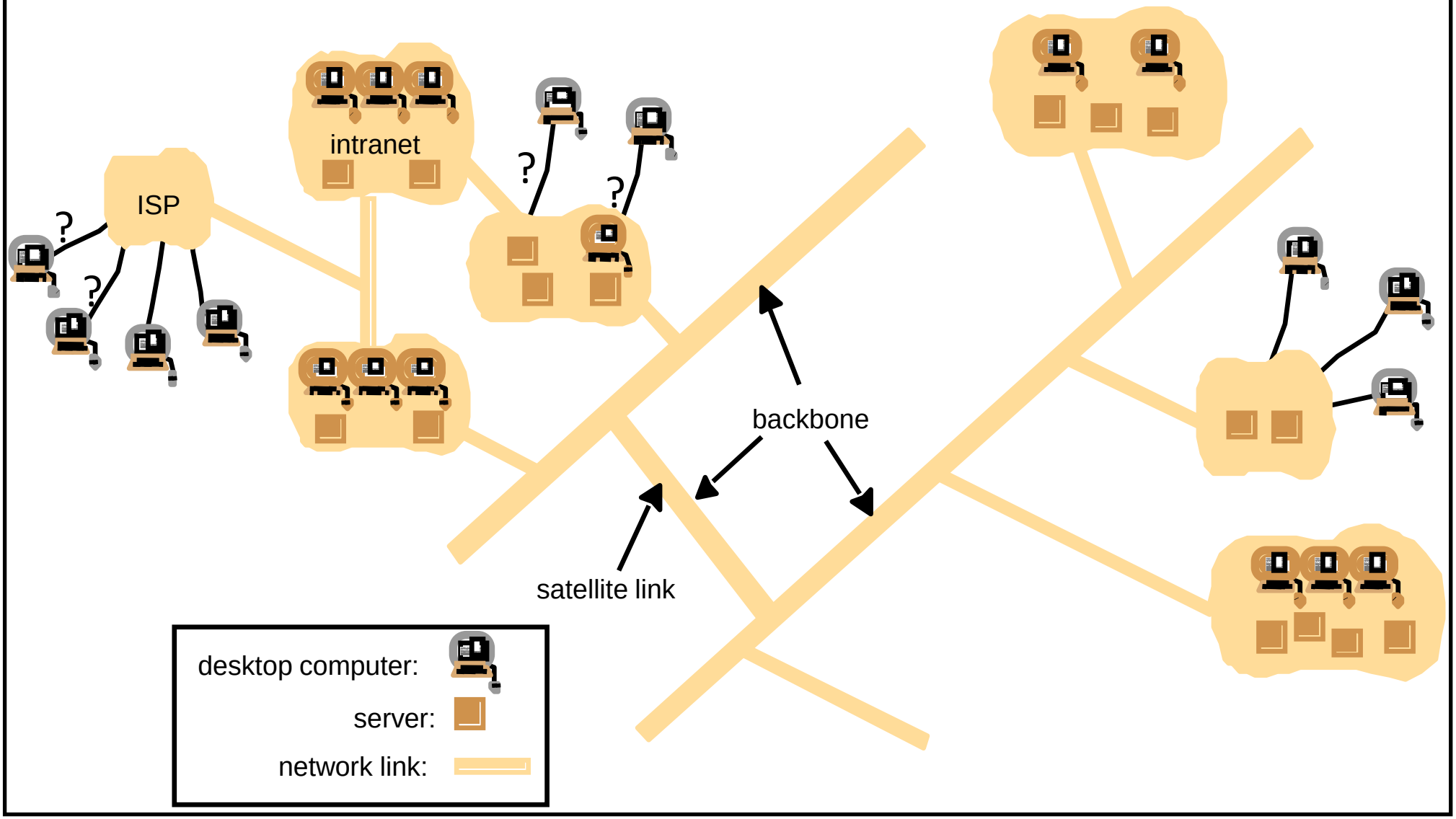
Was ist ein verteiltes System ?

Eine *allgemeinere* Beschreibung:

- 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.

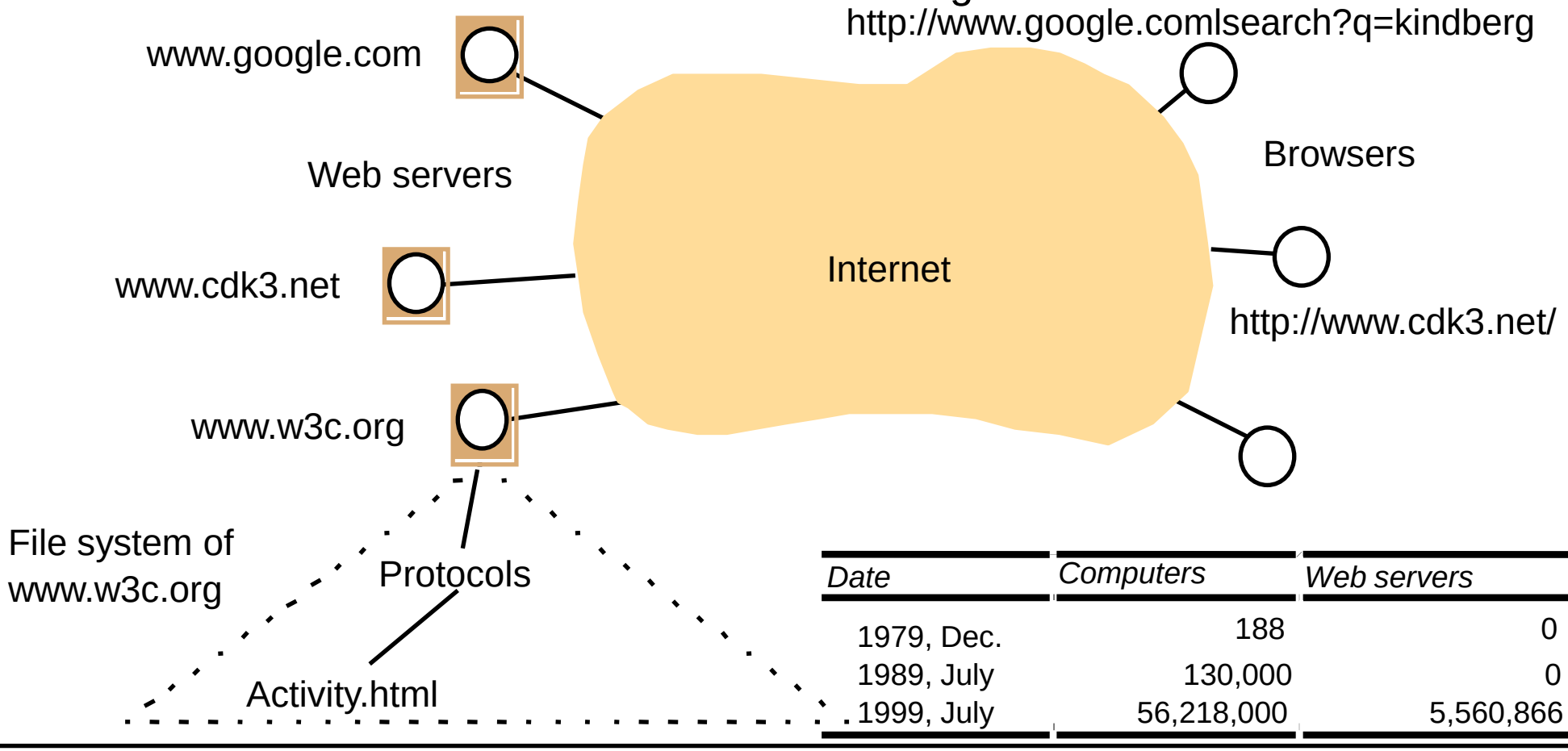
- Eine **verteilte Anwendung** ist eine Anwendung, die ein verteiltes System zur Lösung eines Anwendungsproblems nutzt. Sie besteht aus verschiedenen Komponenten, die mit den Komponenten des VS sowie den Anwendern kommuniziert.

Beispiel Nr1: Das Internet



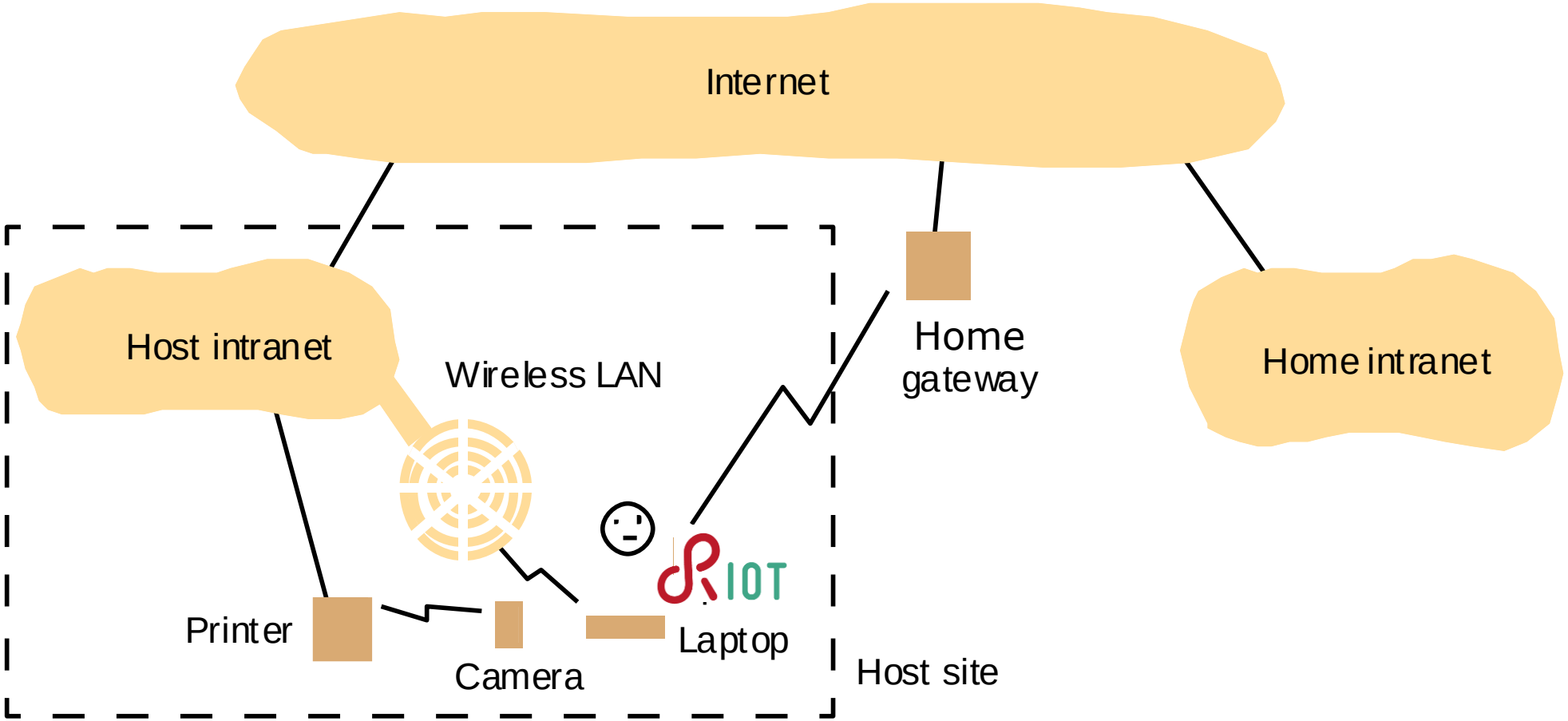
Beispiel Nr2: Das World Wide Web

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

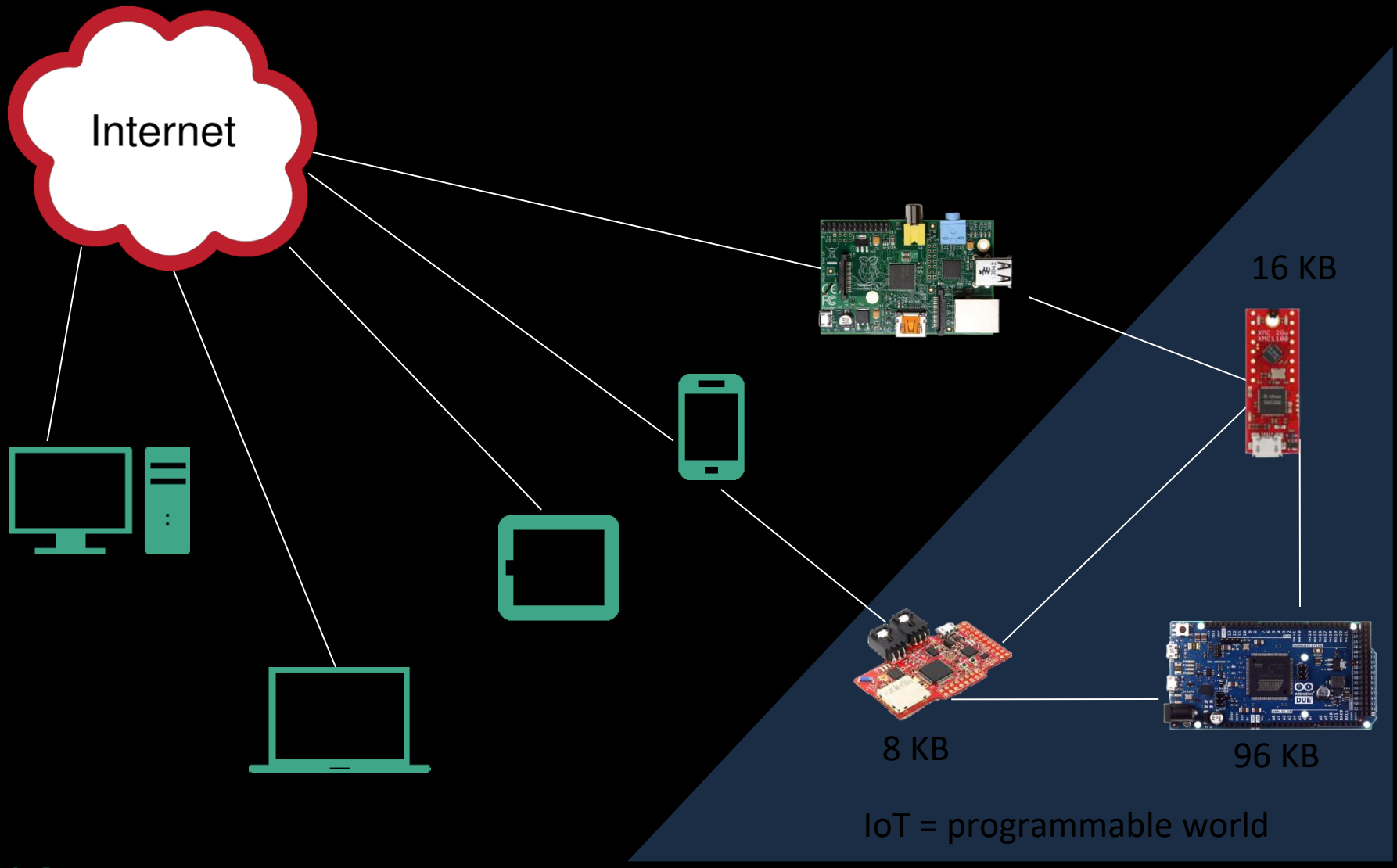


Date	*Computers*	*Web servers*
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866

Beispiel Nr3: „Smart Environment“

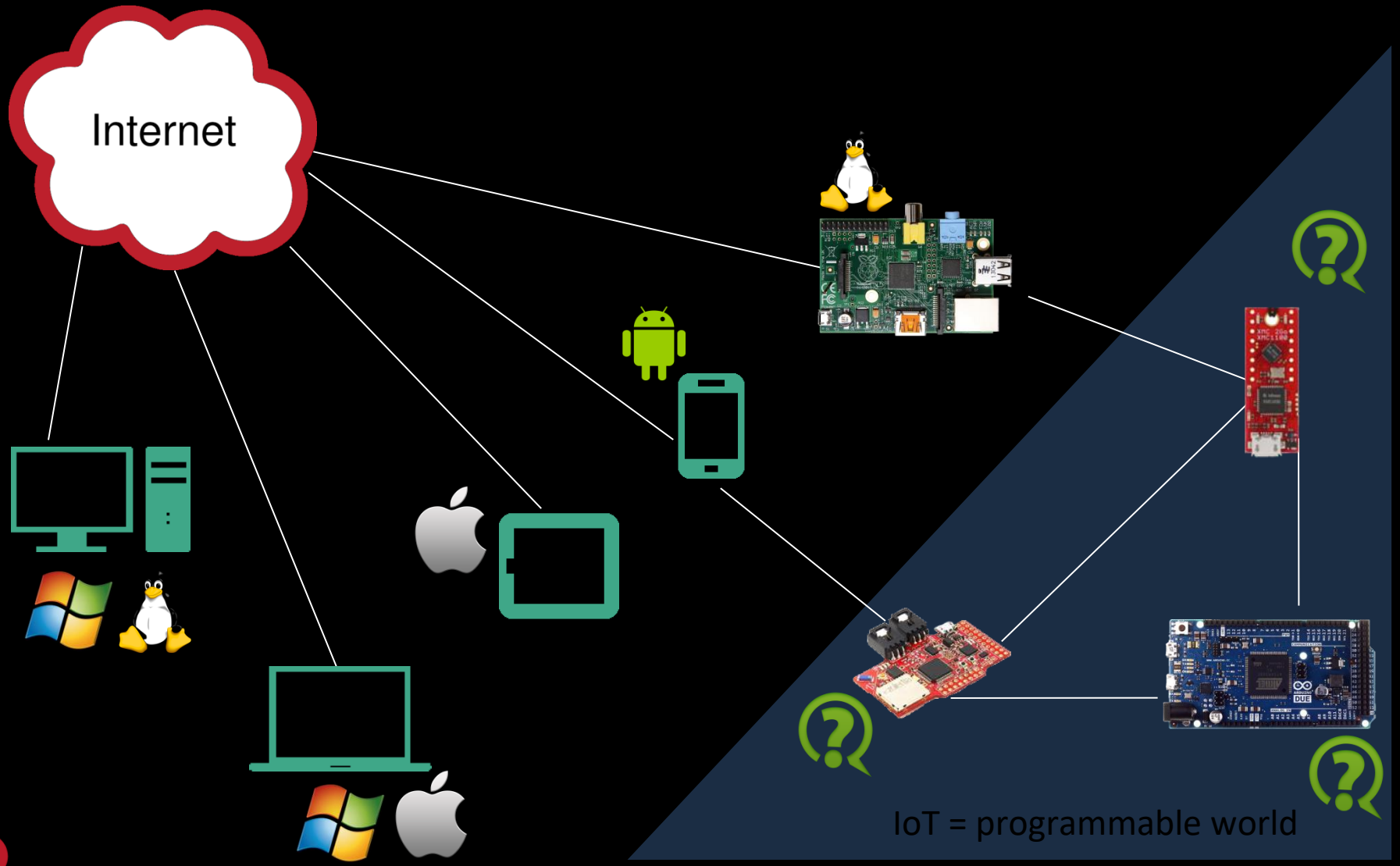


The Internet of Things (IoT)



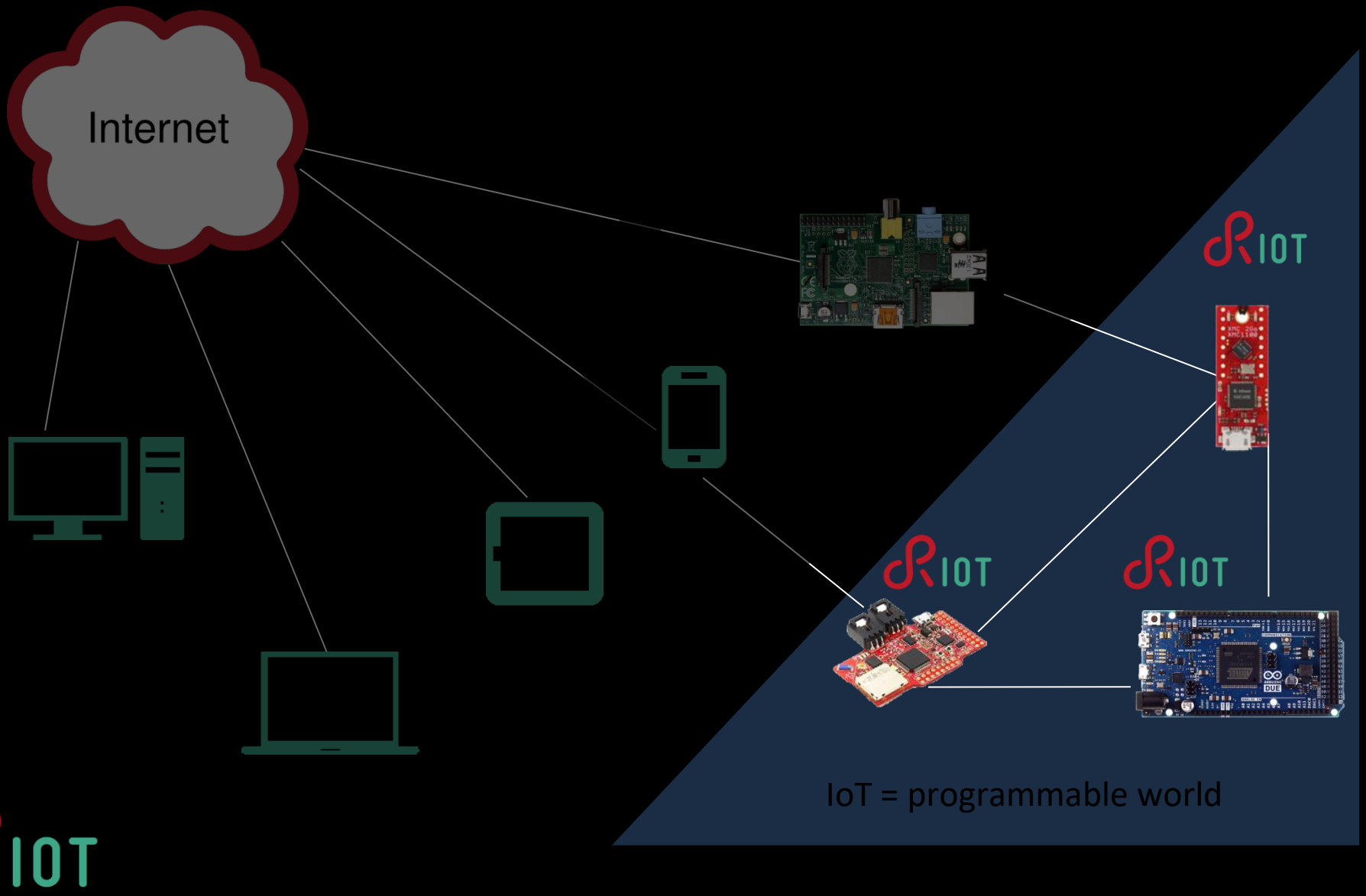
RIOT

IoT: The operating system question



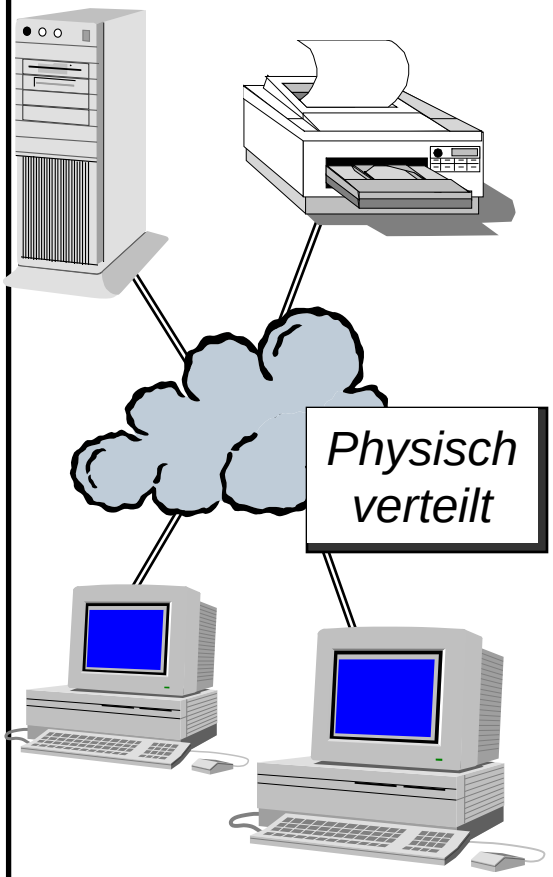
RIOT

RIOT: The friendly IoT operating system

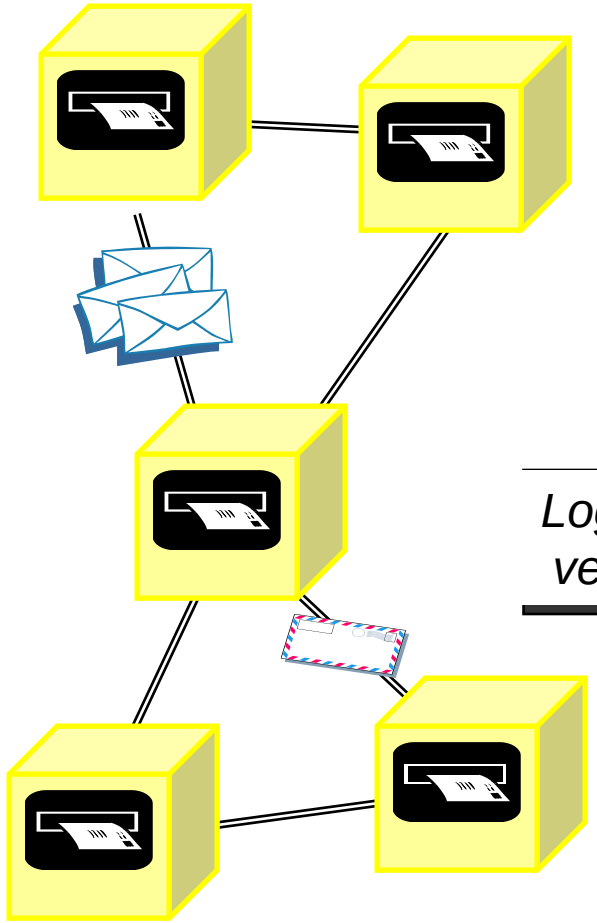


Sichten verteilter Systeme

Rechnernetz mit Rechnerknoten

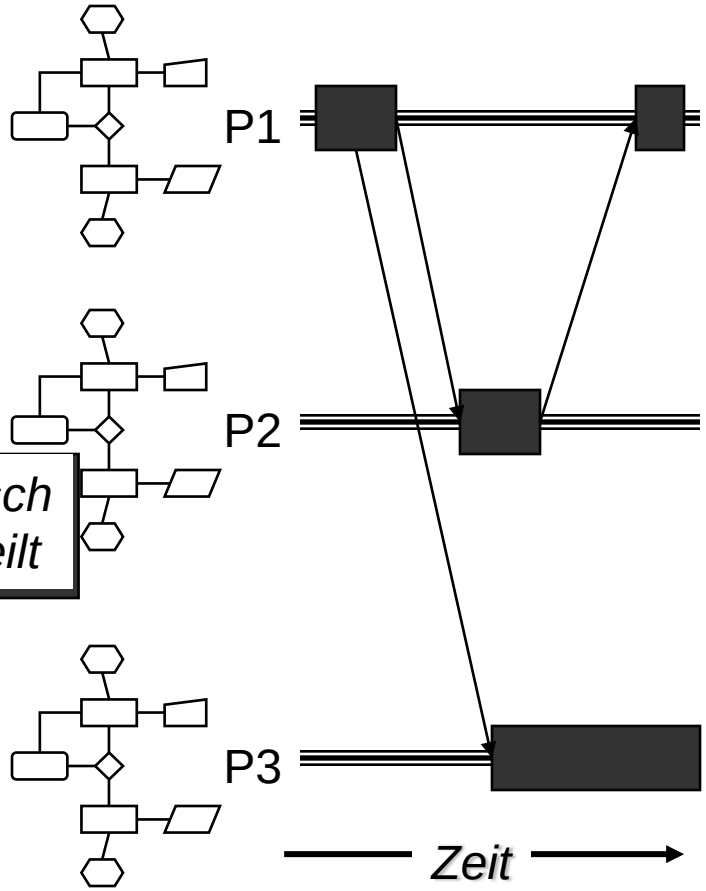


Objekte



Logisch verteilt

Algorithmen u. Protokolle



Wünschenswerte Eigenschaften

- **Gemeinsame Ressourcennutzung**: Hardware, Daten, Dienste etc. gemeinsam nutzen
- **Offenheit**: Schlüsselschnittstellen (einheitlich) offen legen
- **Nebenläufigkeit**: Mehrere gleichzeitig existierende Prozesse
- **Skalierbarkeit**: auch mit vielen Komponenten gut funktionieren können
- **Sicherheit**: Verfügbarkeit, Vertraulichkeit, Integrität, Authentizität, etc
- **Fehlertoleranz**: Fehler erkennen, maskieren, tolerieren
- **Transparenz:** hier im Sinne, **etwas nicht sehen** bzw. durch etwas hindurch sehen können

Transparenz

Transparenz wird definiert als das Verbergen der Separation der einzelnen Komponenten in einem verteilten System vor dem Benutzer und dem Applikationsprogrammierer, so dass das System als Ganzes wahrgenommen wird, und nicht als Sammlung voneinander unabhängiger Komponenten.

ISO (*I*nternational *S*tandards *O*rganization) und ANSA (*A*dvanced *N*etwork *S*ystems *A*rchitecture) identifizieren **acht Formen** der Transparenz*:*

1. **Zugriffstransparenz** ermöglicht den Zugriff auf lokale und entfernte Ressourcen unter Verwendung identischer Operationen.
2. **Positionstransparenz** (Ortstransparenz) erlaubt den Zugriff auf die Ressourcen, ohne dass man ihre Position/ihren Ort kennen muss.
3. **Nebenläufigkeitstransparenz** erlaubt, dass mehrere Prozesse gleichzeitig mit denselben gemeinsam genutzten Ressourcen arbeiten, ohne sich gegenseitig zu stören.

Transparenz

4. **Replikationstransparenz** erlaubt, dass mehrere Instanzen von Ressourcen verwendet werden, um die Zuverlässigkeit und die Leistung zu verbessern, ohne dass die Benutzer oder Applikationsprogrammierer wissen, dass Repliken verwendet werden.
5. **Fehlertransparenz** erlaubt das Verbergen von Fehlern, so dass Benutzer und Applikationsprogrammierer ihre Aufgaben erledigen können, auch wenn Hardware- oder Softwarekomponenten ausgefallen sind.
6. **Mobilitätstransparenz** erlaubt das Verschieben von Ressourcen und Clients innerhalb eines Systems, ohne dass die Arbeit von Benutzern oder Programmen dadurch beeinträchtigt wird.
7. **Leistungstransparenz** erlaubt, dass das System neu konfiguriert wird, um die Leistung zu verbessern, wenn die Last variiert.
8. **Skalierungstransparenz** erlaubt, dass sich System und Applikationen vergrößern, ohne dass die Systemstruktur oder die Applikationsalgorithmen geändert werden müssen.

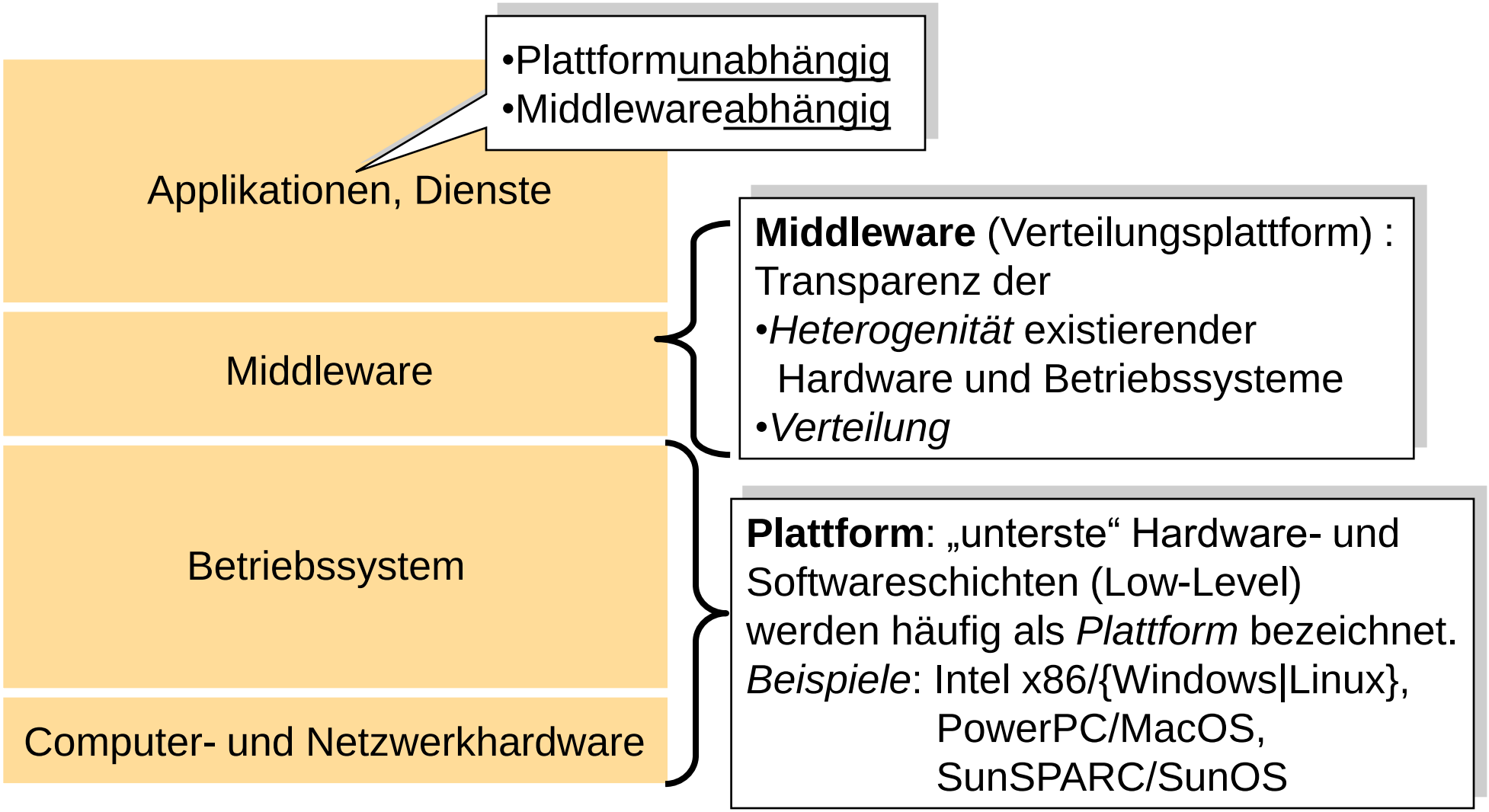
Verteilte Systeme

Systemmodelle

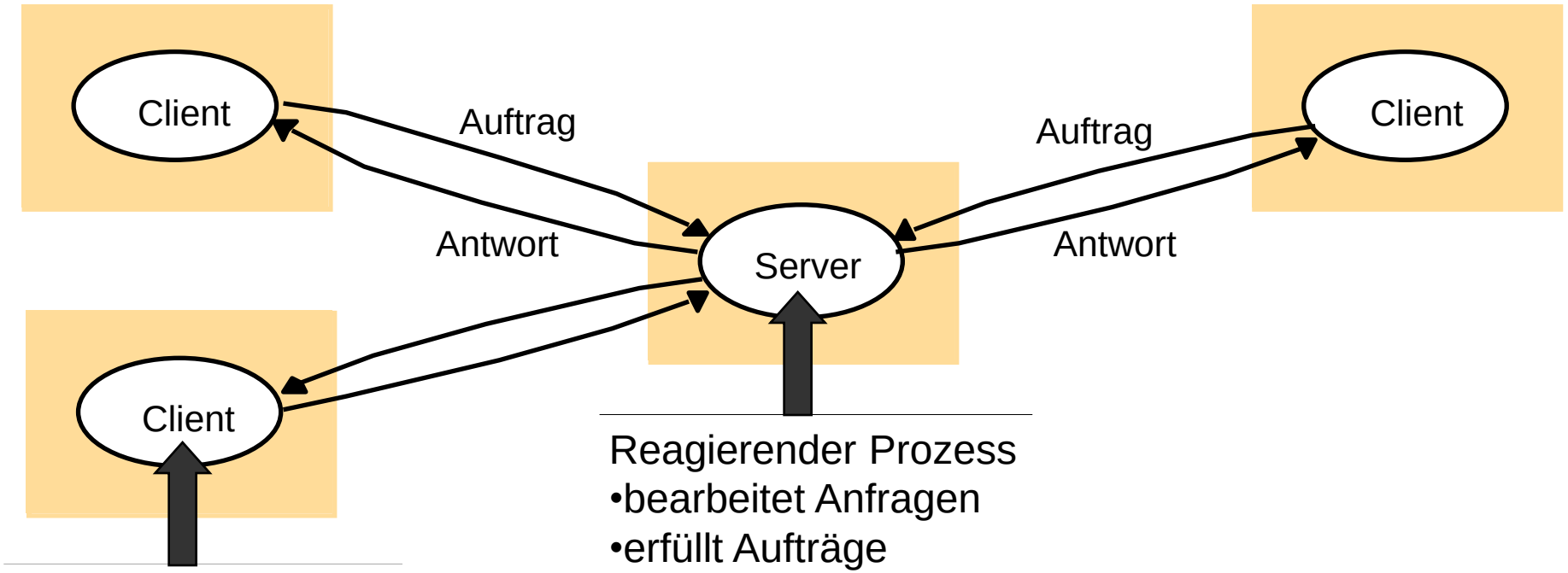
Systemmodelle

- Beschreibung der allgemeinen Eigenschaften und des Designs eines Systems
- Das **Modell** sollte abdecken:
  - Die wichtigsten Komponenten des Systems und ihre Funktion
  - Die Art ihrer Interaktion einschl. Schnittstellen
  - Wie deren individuelles und kollektives Verhalten beeinflusst werden kann
- Ein **Architekturmodell**
  - *vereinfacht und abstrahiert* zunächst die Funktionen der individuellen Komponenten eines verteilten Systems, um dann
  - die *Verteilung* der Komponenten auf ein Netzwerk von Computern und
  - die *Beziehung* der Komponenten (Rolle in der Kommunikation mit anderen, Kommunikationsmuster) untereinander zu beschreiben.
- Weitere Modelle: Interaktionsmodell, Fehlermodell, Sicherheitsmodell

Hardware- und Software-Serviceschichten

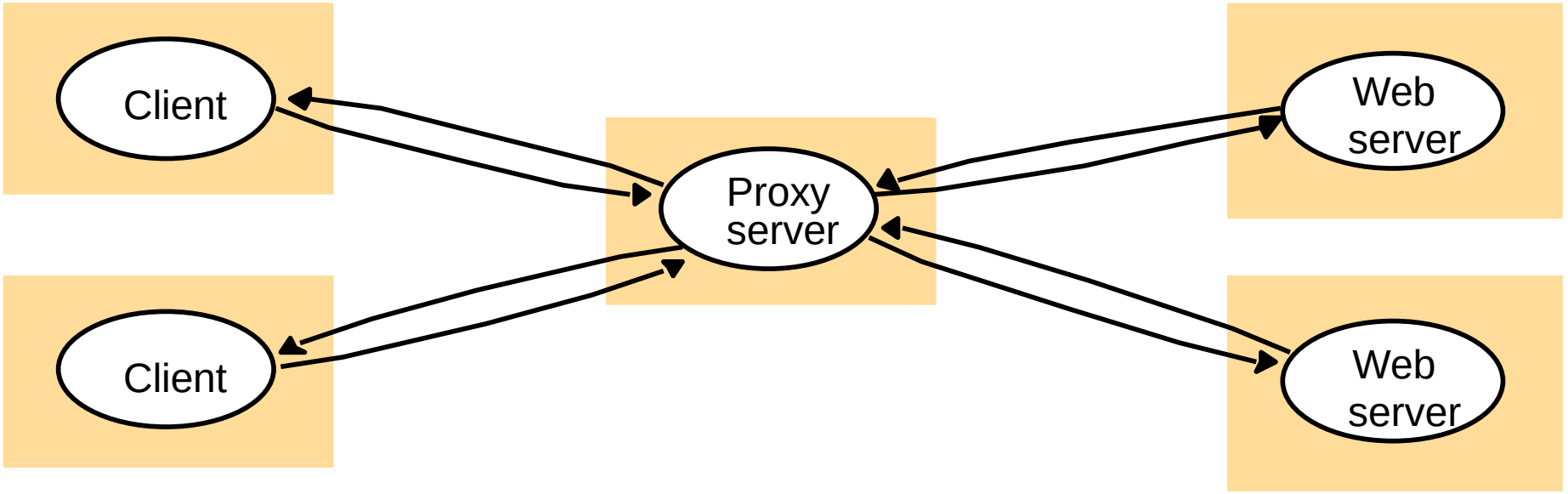


Client/Server Modell



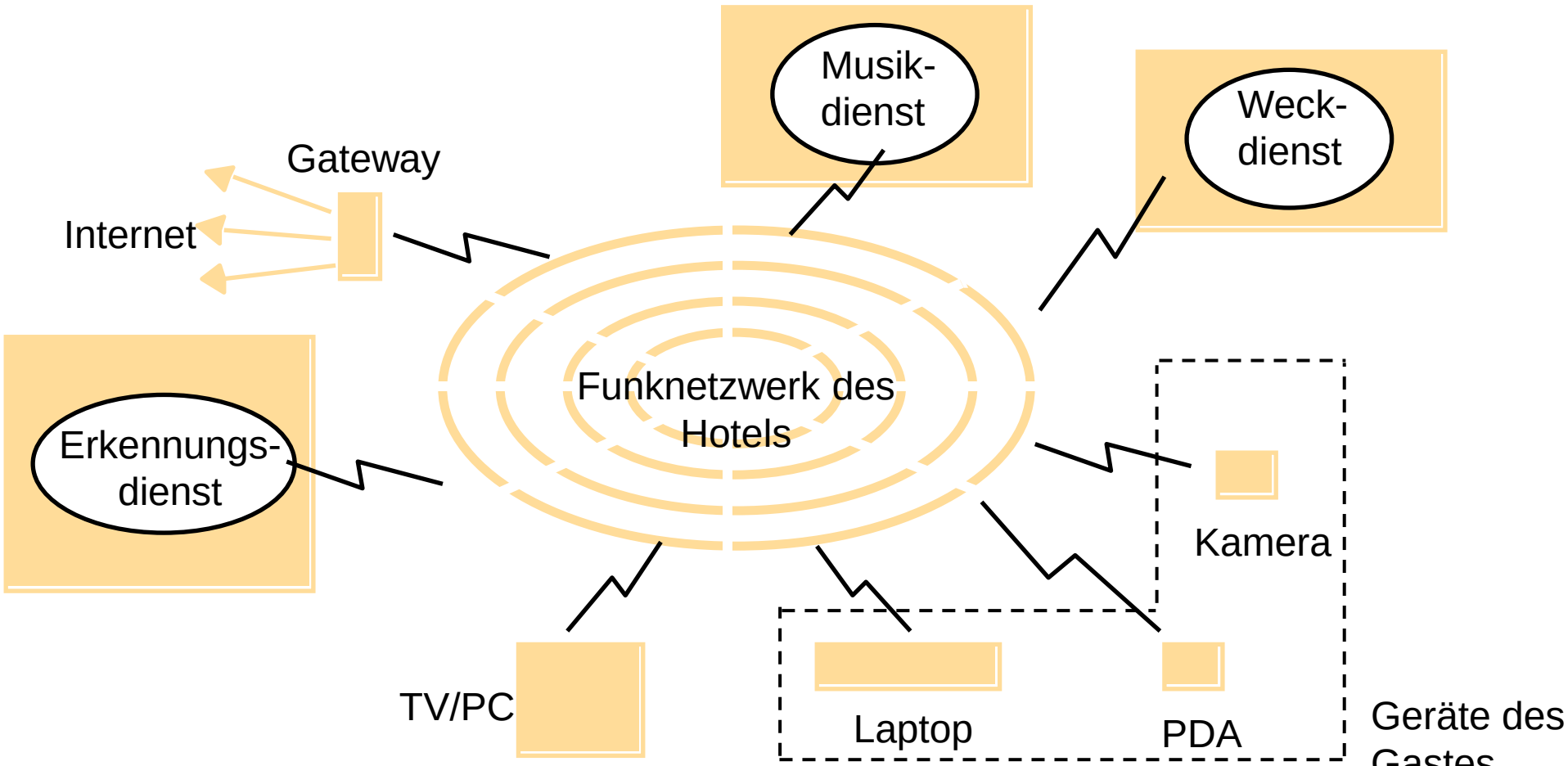
Legende:
Prozess:
Computer:

Proxy-Server und Cache

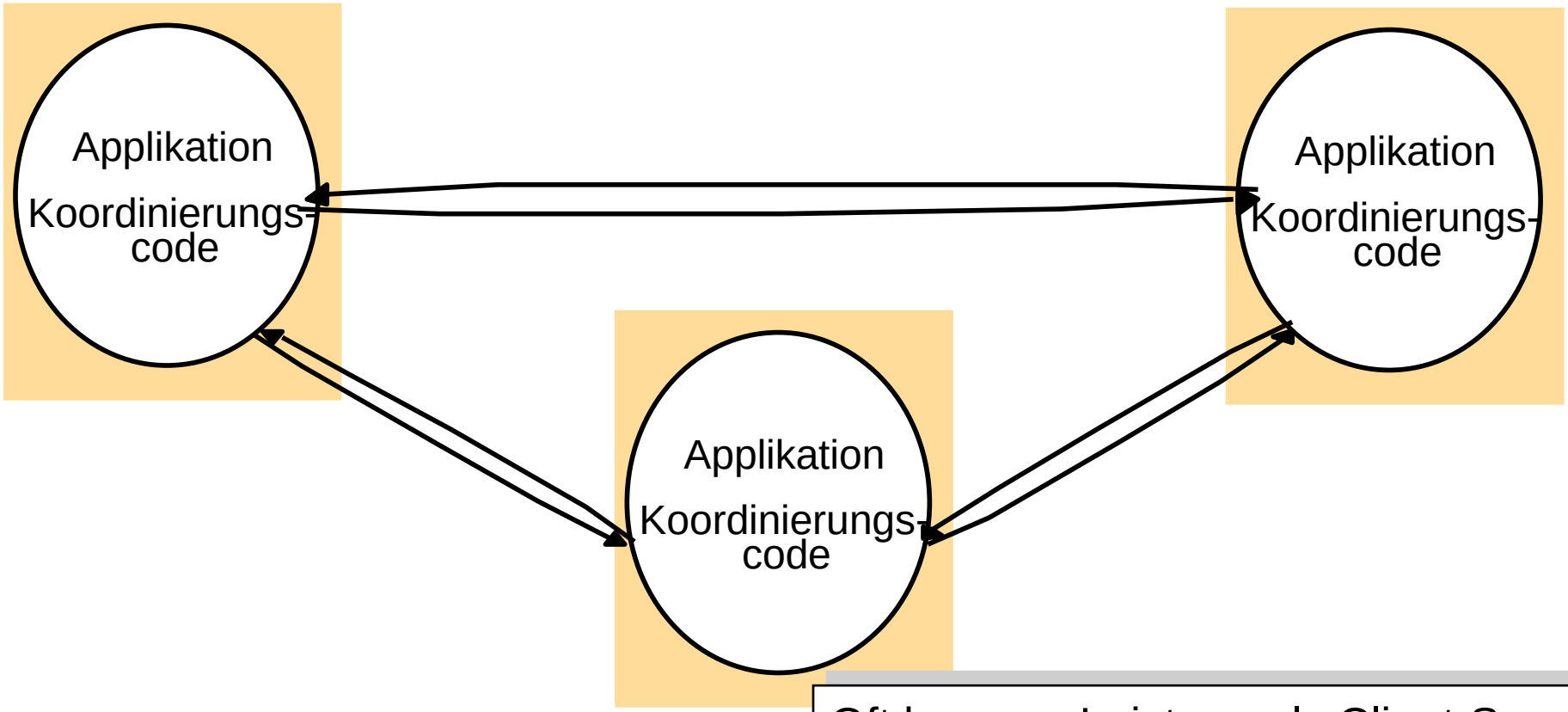


Proxy: Stellvertreter einer tatsächlichen Instanz
Zweck eines Proxies: Erhöhung der Leistung, Zugänglichkeit oder Verfügbarkeit

Spontane Netzwerkverbindungen

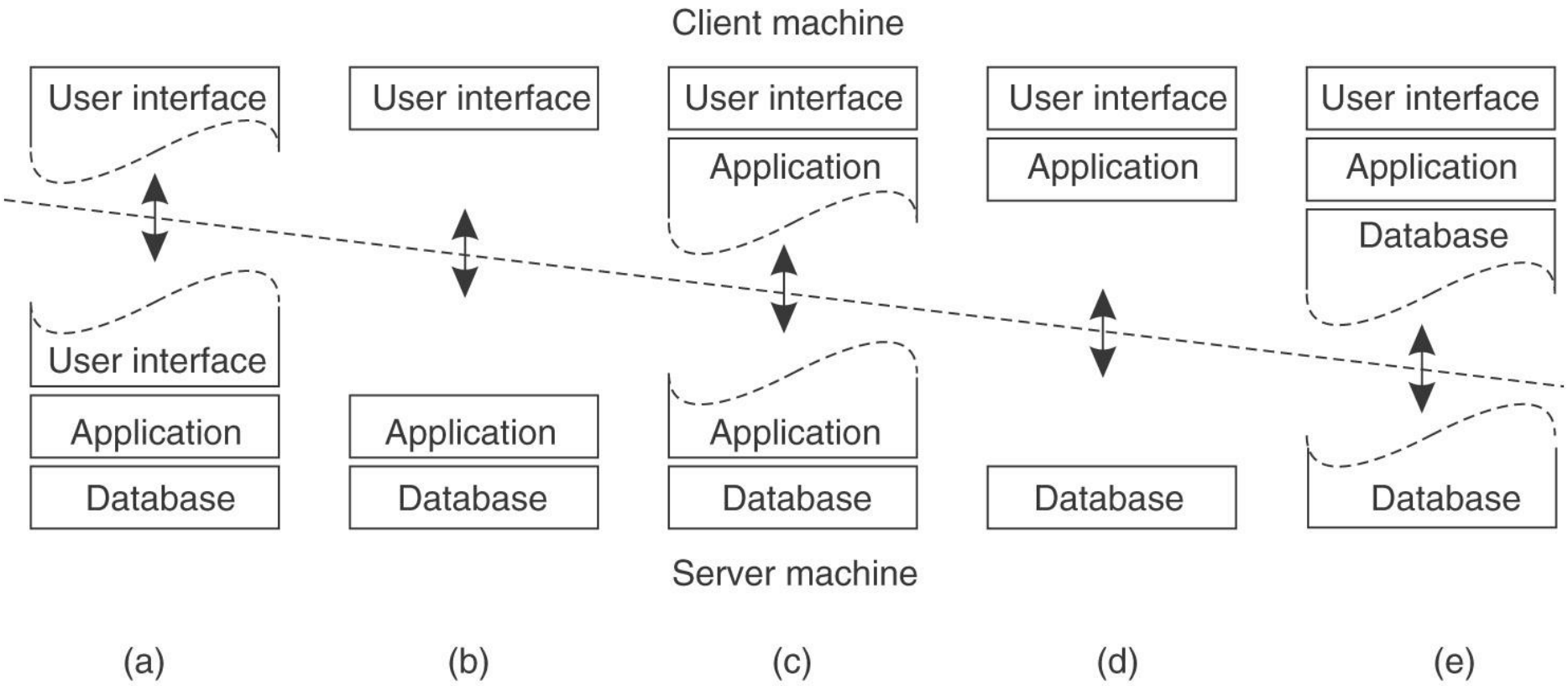


Gleichrangige Prozesse (Peer Processes)

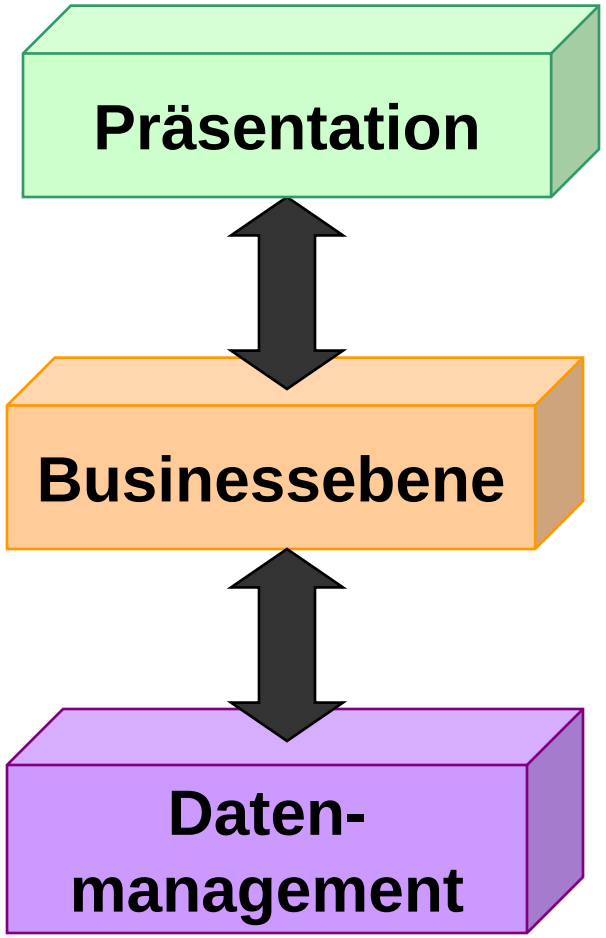


Oft bessere Leistung als Client-Server, durch Nutzung der Gesamtressourcen, die mit Teilnehmern wachsen.
Beispiel: Filesharing, Conferencing

Thin Client/Fat Server



Drei Ebenen Architektur: „*three-tier*“

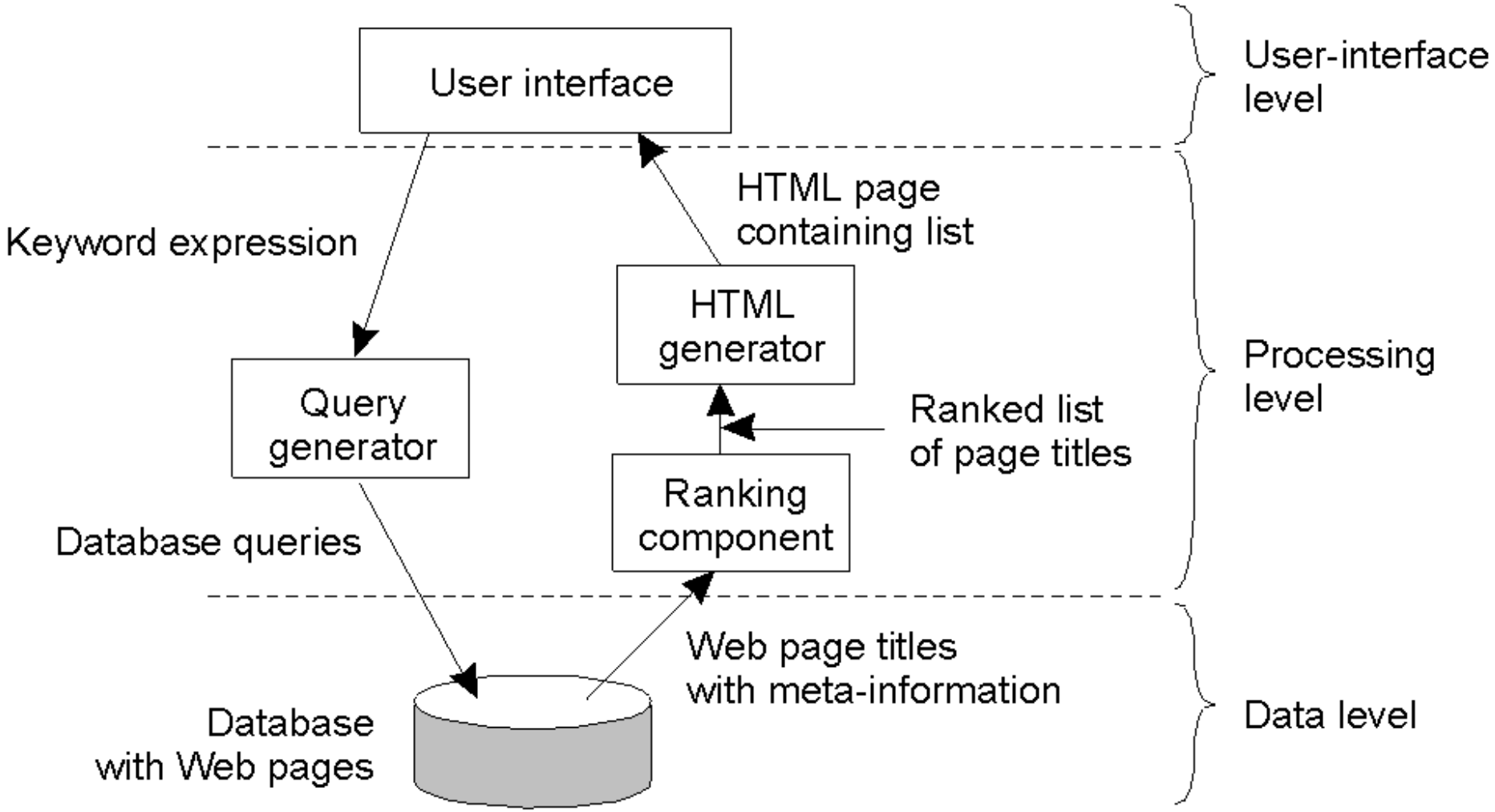


Kommuniziert mit dem Anwender

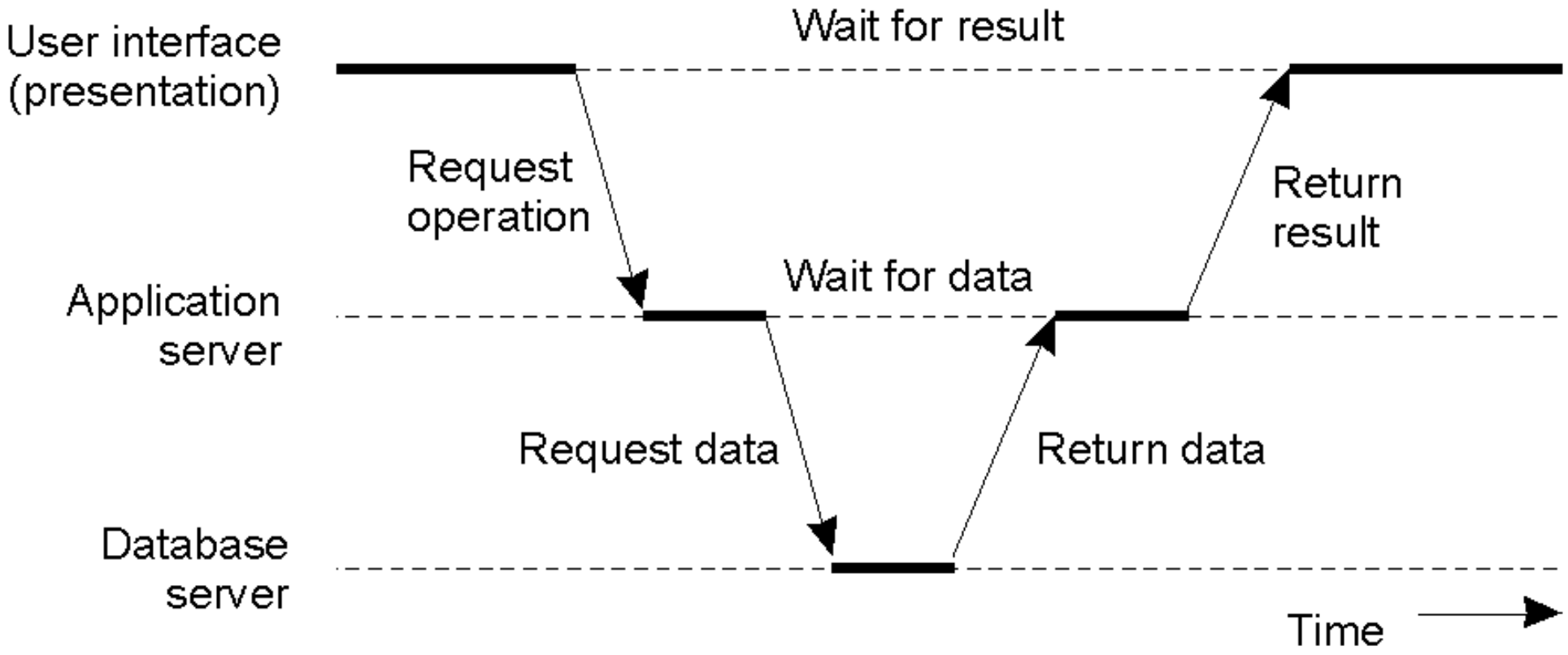
Führt die Geschäftsregeln aus,
verwaltet Prozessinformationen

Datenzugriff und Datenspeicherung

Beispiel: Suchmaschine



Beispiel: Suchmaschine



Anforderungen an die Auswahl eines Modells

- Welches sind die **Einflussfaktoren**
  - bei der **Auswahl** des Modells
  - bei der **Platzierung der Komponenten**
- **Wichtig** u.a.
  - Performance (Antwortverhalten, Durchsatz, Lastbalancierung, etc.)
  - Quality of Service (Zuverlässigkeit, Sicherheit, Echtzeit, etc.)
  - Inwieweit sollen Caching und Replikation eingesetzt werden (mit welchem Konsistenzmodell?)
  - Benötigter Grad der Verlässlichkeit des Systems (Korrektheit, Sicherheit, Fehlertoleranz, etc.)
  - Kosten (Anschaffung, Schulung, Lizenzen, etc.)