

MAS: Betriebssysteme

Architekturansätze
(im Fokus: Universalbetriebssysteme)

T. Pospíšek

Gesamtüberblick

1. Einführung in Computersysteme
2. Entwicklung von Betriebssystemen
- 3. Architekturansätze**
4. Interruptverarbeitung in Betriebssystemen
5. Prozesse und Threads
6. CPU-Scheduling
7. Synchronisation und Kommunikation
8. Speicherverwaltung
9. Geräte- und Dateiverwaltung
10. Betriebssystemvirtualisierung

Zielsetzung

- Die verschiedenen Architekturen von Betriebssystemen kennenlernen
- Aspekte der Verteilung von Betriebssystemen und Applikationen kennenlernen
- Sinn und Möglichkeiten der Virtualisierung von Betriebssystemen und von Cloud Computing kennenlernen

Überblick

1. Zugriffsschutz in Betriebssystemen

2. Lokale Architekturen
3. Verteilte Verarbeitung
4. Terminalserver-Betrieb
5. Virtualisierung von Betriebs- und Laufzeitsystemen
6. Cloud Computing

Usermodus und Kernelmodus

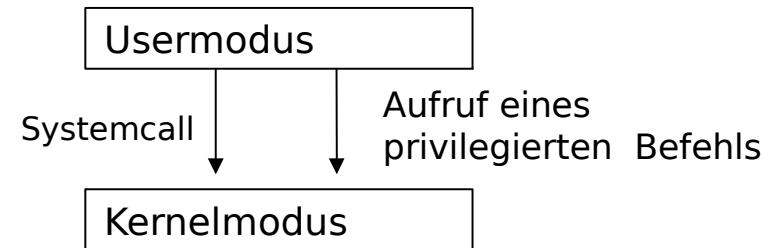
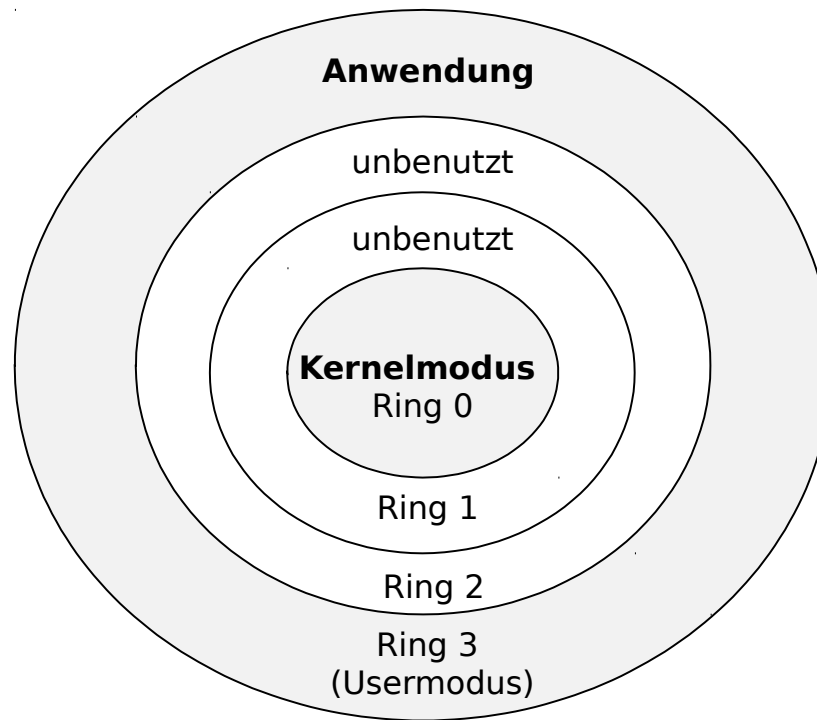
- **Usermodus (Benutzermodus)**
 - Ablaufmodus für Anwendungsprogramme
 - Kein Zugriff auf Kernel-spezifische Code- und Datenbereiche
 - Ausführung von privilegierten Maschinensprache Befehlen nicht erlaubt
- **Kernelmodus**
 - Privilegierter Modus
 - Dient der Ausführung der Programmteile des Kernels
 - Schutz von Datenstrukturen des Kernels
- **Umschaltung** über spezielle Maschinenbefehle
- Aktueller Modus steht in einem **Statusregister**

Hardware-Grundlagen am Beispiel der Intel-Architektur (1)

- Beispiel: x86-Architektur
 - Schutzkonzept über vier Privilegierungsstufen (Ring 0 - 3)
 - Prozess läuft zu einer Zeit in einem Ring
 - Meist werden aus Kompatibilitätsgründen zu anderen CPUs nur zwei Ringe unterstützt:
 - Ring 0: Kernelmodus (privilegiert, Zugriff auf Hardware möglich)
 - Ring 3: Usermodus (nicht privilegiert)
 - Übergang von Ring 3 nach Ring 0 über privilegierte Operation (int-Befehl)
 - Anmerkung: Ab x64/IA64 werden nur noch zwei Ringe unterstützt

Hardware-Grundlagen am Beispiel der Intel-Architektur (2)

- Wechsel von Ring 3 nach Ring 0 (Trap, Unterbrechung)
→ mehr dazu in Kapitel 3



- Im Ring 3 ist nicht der komplette Befehlssatz vorhanden

Überblick

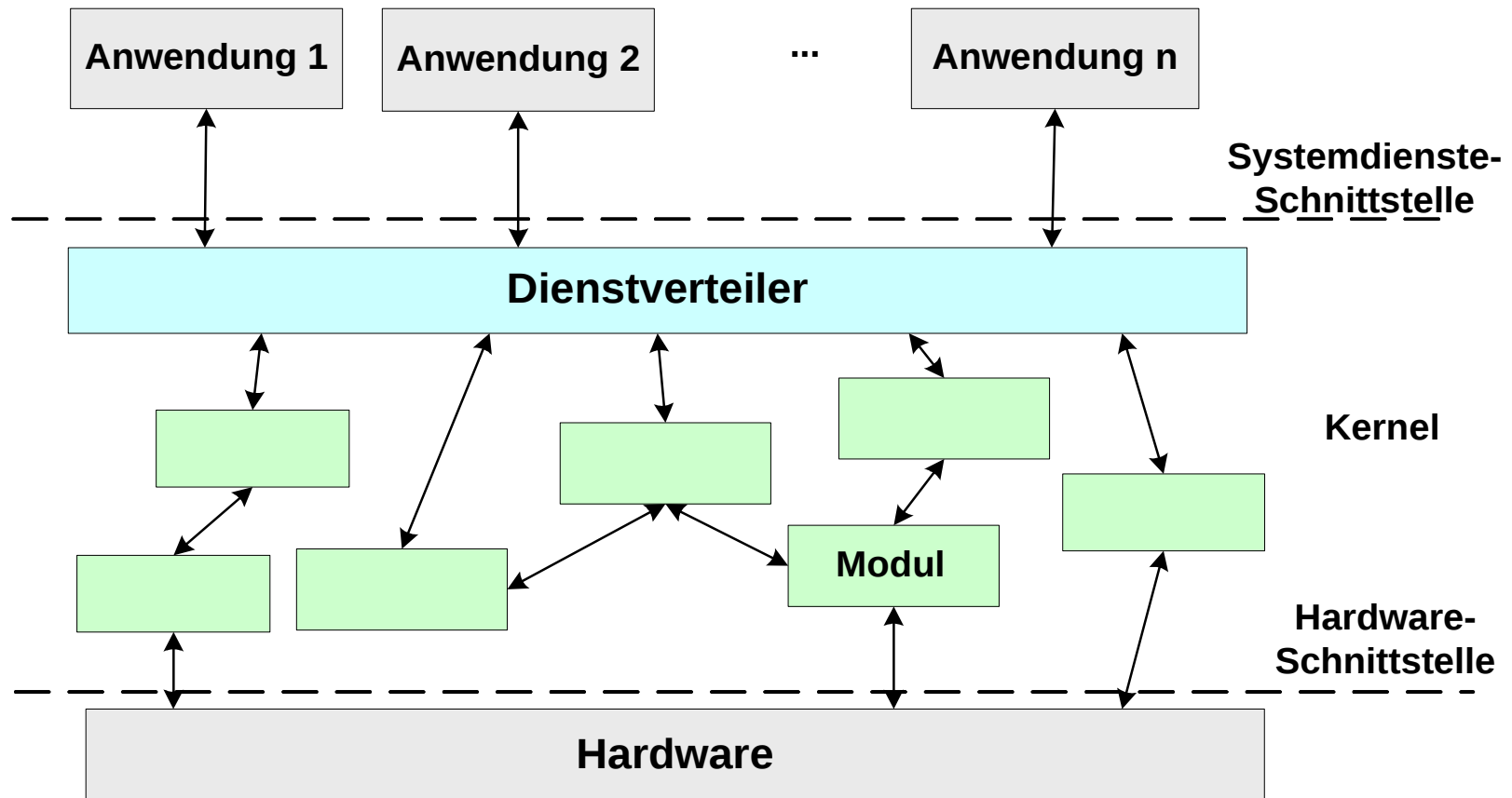
1. Zugriffsschutz in Betriebssystemen
- 2. Lokale Architekturen**
3. Verteilte Verarbeitung
4. Terminalserver-Betrieb
5. Virtualisierung von Betriebs- und Laufzeitsystemen
6. Cloud Computing

Der Betriebssystemkern

- Der Betriebssystemkern (Kernel, Kern) umfasst wesentliche Dienste des Betriebssystems, die möglichst immer im Hauptspeicher geladen sein sollen
- Hierzu gehört u.a.:
 - Prozess- und Prozessorverwaltung
 - Speicherverwaltung
 - Dateiverwaltung
 - Geräteverwaltung (Treibersoftware)
 - Netzwerkverwaltung

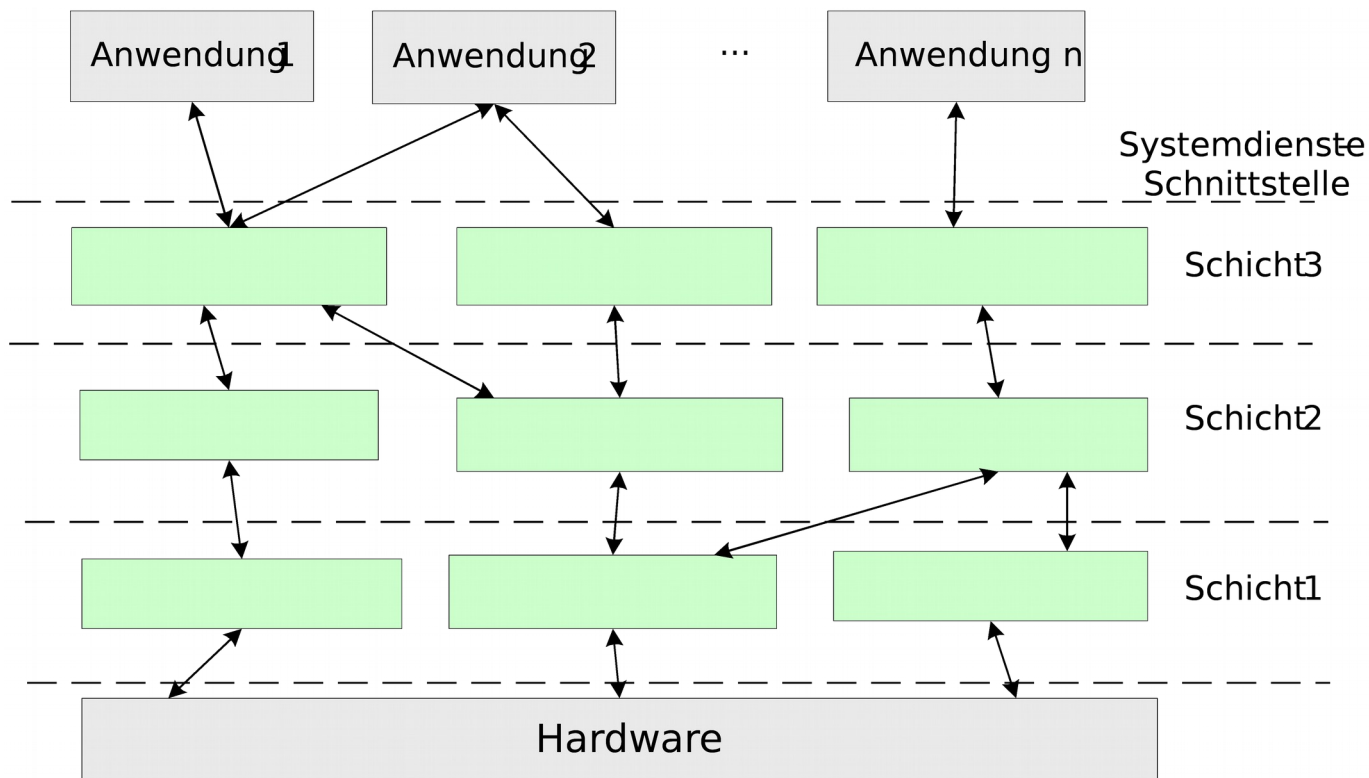
Monolithischer Kern

- Alle Module haben Zugriff auf einen Adressraum und teilen sich diesen (in einem Prozess, Definition später)



Schichtenmodell

- Verbesserung des monolithischen Kernels: Flexibler und übersichtlicher

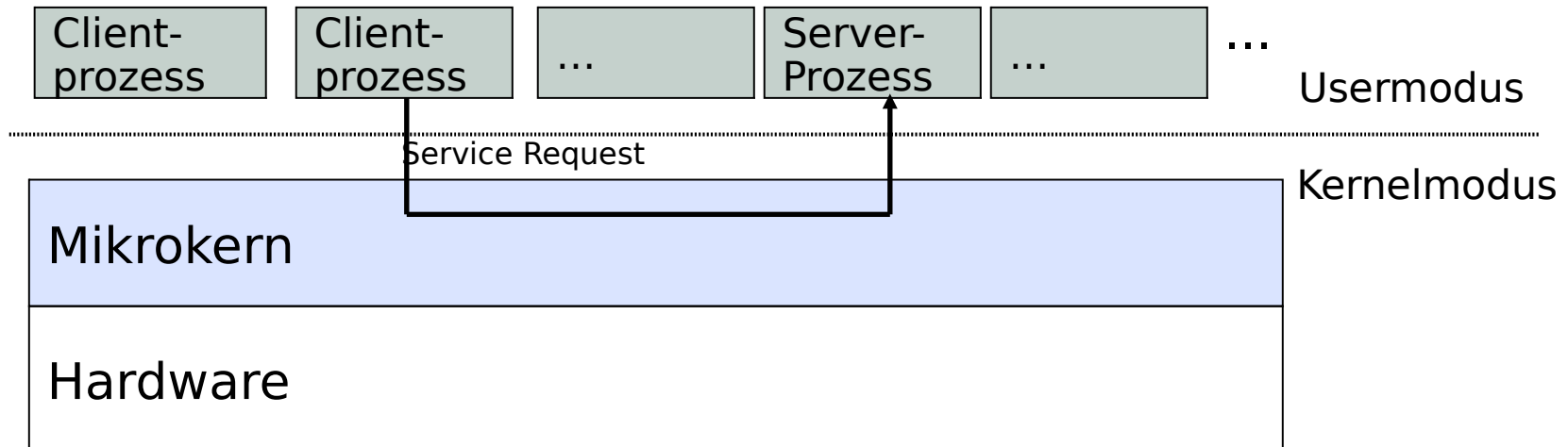


Mikrokern (1)

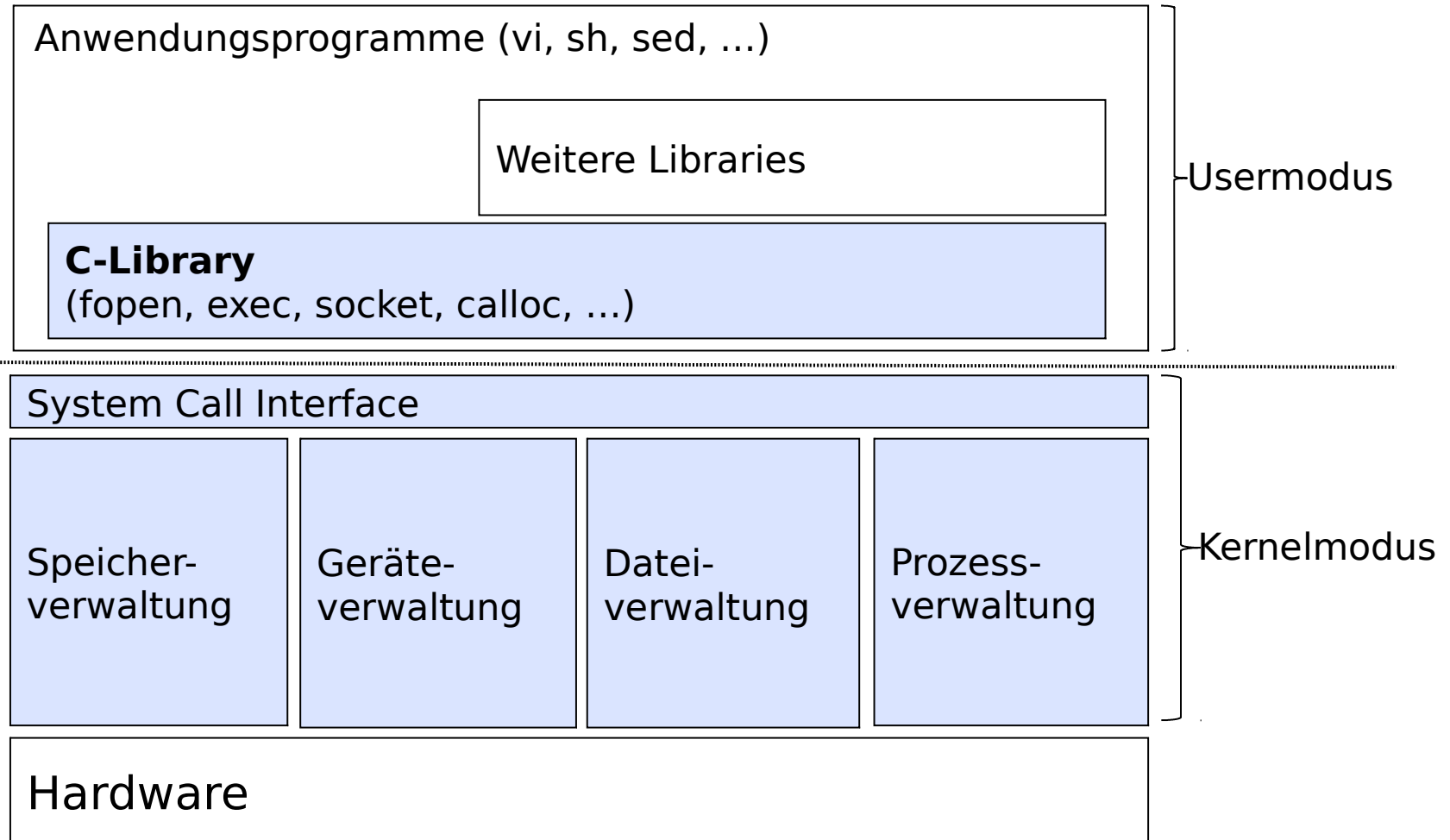
- Ein moderner Trend in der Betriebssystementwicklung:
 - Kernel „leichter“ machen
 - Entlastung durch Übernahme von Funktionalität in Anwendungsprozesse (Serverprozesse), eigene Adressräume
 - Serverprozesse können sein:
 - Fileserver
 - Memory Server (?)
 - ...
 - Was bleibt ist ein minimaler **Mikrokern**
 - Der Mikrokern übernimmt die Abwicklung der Kommunikation zwischen Anwendungsprozessen (Clientprozessen) und Serverprozessen
 - Auch „Message Passing OS“ genannt

Mikrokern (2)

- Clientprozesse fordern Dienste über Nachrichten an die Serverprozesse an
- Mikrokern-Konzept wurde u.a. bei den Betriebssystemen **Amoeba** (Tanenbaum), **Mach** und **Chorus** entwickelt
- Moderne Microkernel Familie: L7
- Elementarer Kern ermöglicht verschiedene, darauf aufsetzende Betriebssysteme
- Performance-Verschlechterung



Unix-Schichtenmodell (klassisch monolithisch)

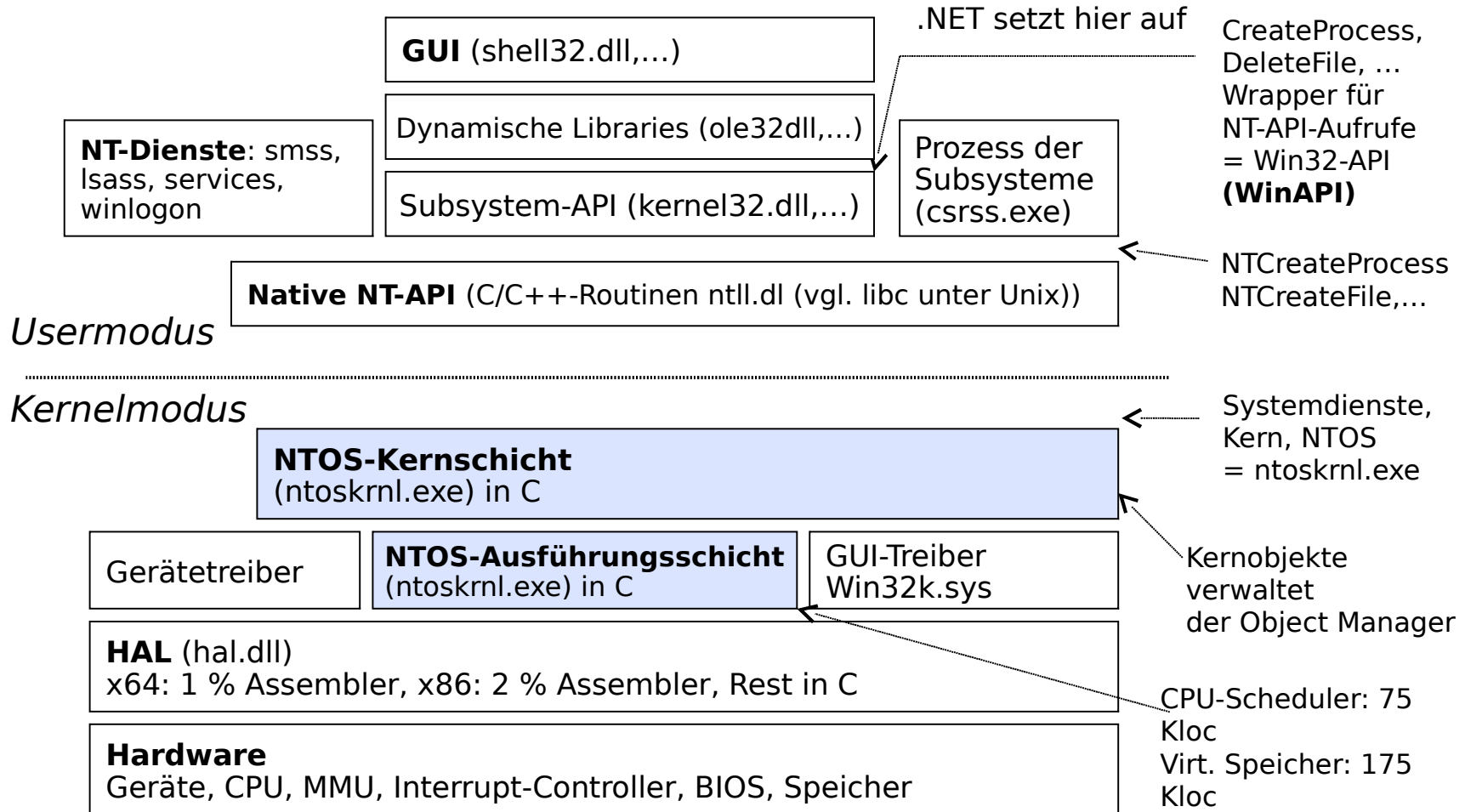


Linux-Architektur (monolithisch)



- siehe auch <http://lxr.free-electrons.com> Linux Cross Reference

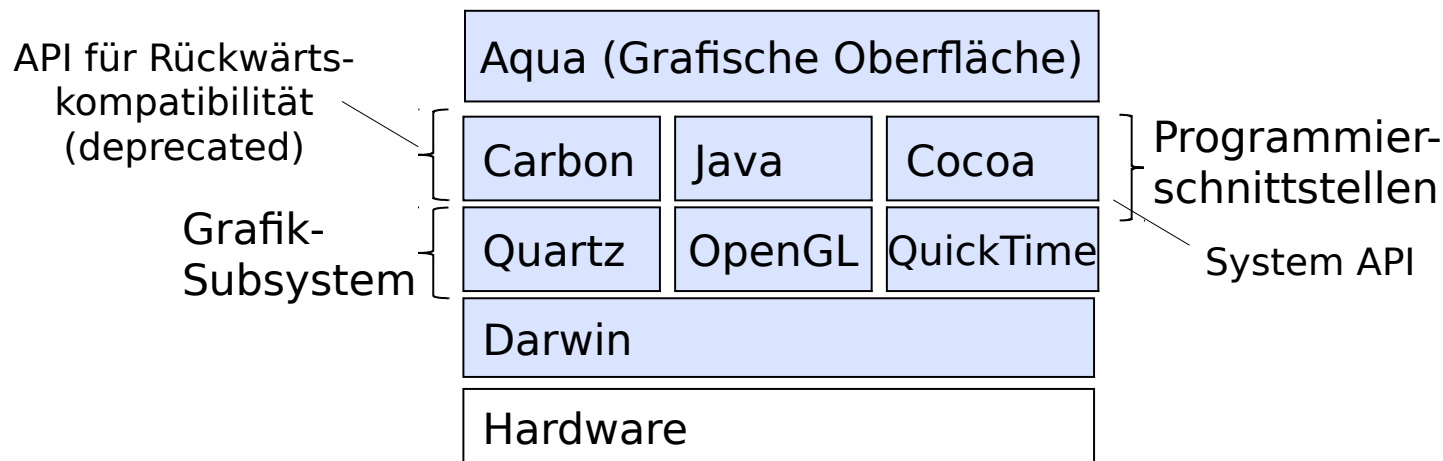
Windows-Schichtenarchitektur



Quelle: Tanenbaum, A. S.: Moderne Betriebssysteme., 3. aktualisierte Auflage, Pearson Studium, 2009

Architektur von Apple Mac OS X

- Mac OS X hat einen Hybrid-Kernel
 - Mischung aus MACH und FreeBSD
 - MACH ist ein Microkernel und NeXTStep nutzte MACH-Kernel
- Entwicklung:
 - Unix → BSD → NeXTStep → Darwin → Mac OS X
- OS X Kernel heißt **XNU** (X is not Unix)
- Abgewandelt auch in iOS für Apple iPhone, iPad, ...



Überblick

1. Zugriffsschutz in Betriebssystemen
2. Lokale Architekturen
- 3. Verteilte Verarbeitung**
4. Terminalserver-Betrieb
5. Virtualisierung von Betriebs- und Laufzeitsystemen
6. Cloud Computing

Verteilte Systeme

■ **Verteilte Betriebssysteme:**

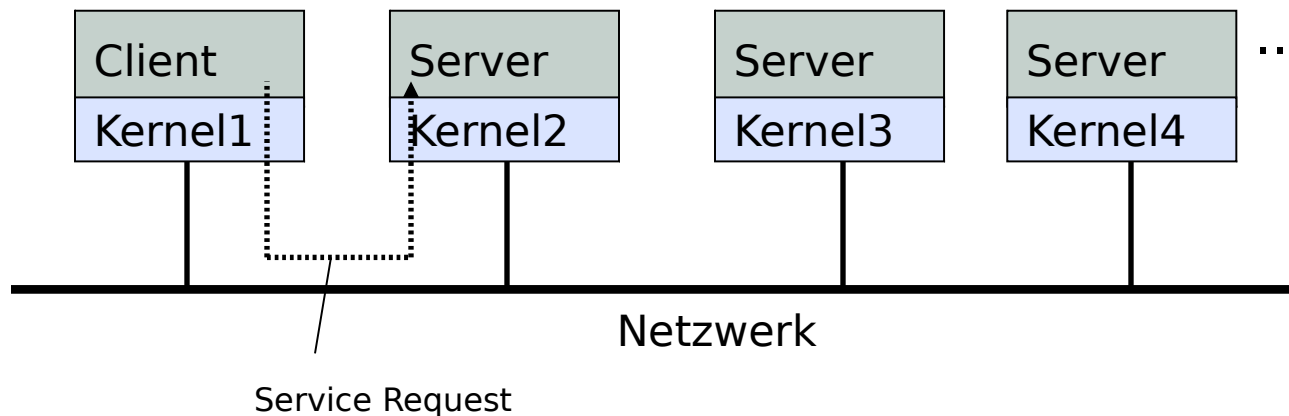
- Die Mikrokern-Architektur vereinfacht auch eine Verteilung der Serverprozesse auf mehrere Rechner in einem Netzwerk
- Ein **echt verteiltes Betriebssystem** macht die Verteilung der Services im Netz für den Clientprozess transparent
- Heute keine Praxisrelevanz

■ **Kommunikations-Middleware:**

- Heutige verteilte Systeme basieren in der Regel auf einer Middleware (Zwischenschicht), die im Usermodus abläuft
- Meist Nutzung des Client-/Server-Prinzips, Publish-Subscribe, aber auch Peer-to-Peer immer mehr verbreitet

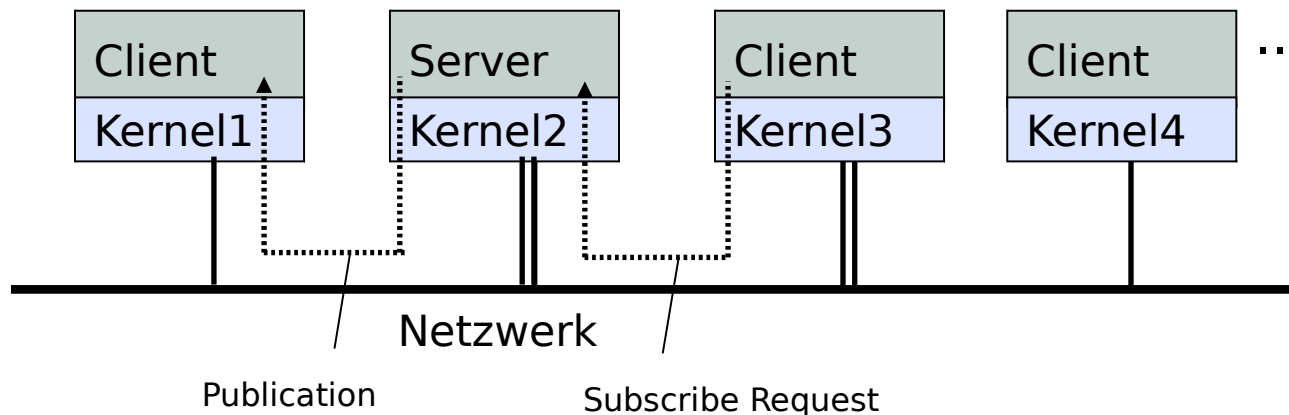
Client-/Server-Modell

- Dedizierte Rollen: Client und Server
- Client und Serverkomponenten sind üblicherweise über ein Netzwerk verteilt
- Client und Server kommunizieren über Requests
- Softwarekonzept!



Publish-/Subscribe-Modell

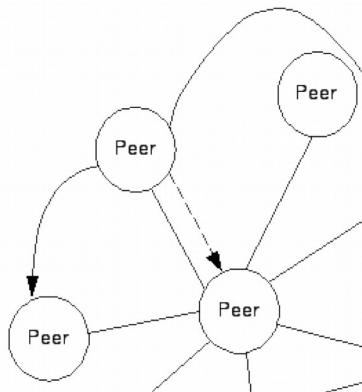
- Dedizierte Rollen: Publisher und Subscriber
- Publisher und Subscriberkomponenten sind üblicherweise über ein Netzwerk verteilt
- Client und Server kommunizieren über (transparente) Messages
- Softwarekonzept!



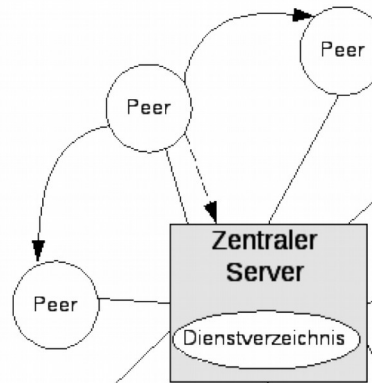
Peer-to-Peer-Modell

- Eigenes Betriebssystem auf jedem Peer
- Keine Rollenaufteilung wie im Client-/Server-Modell
- Varianten: pur, hybrid, Superpeer (Weiterentwicklung von hybrid)

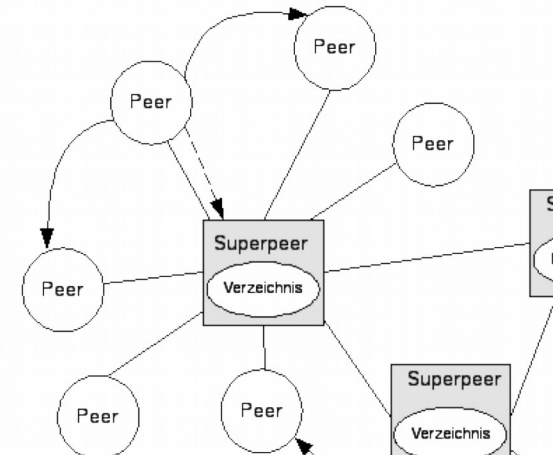
Pur P2P
Beispiel: Gnutella



Hybrid P2P
Beispiele: Napster



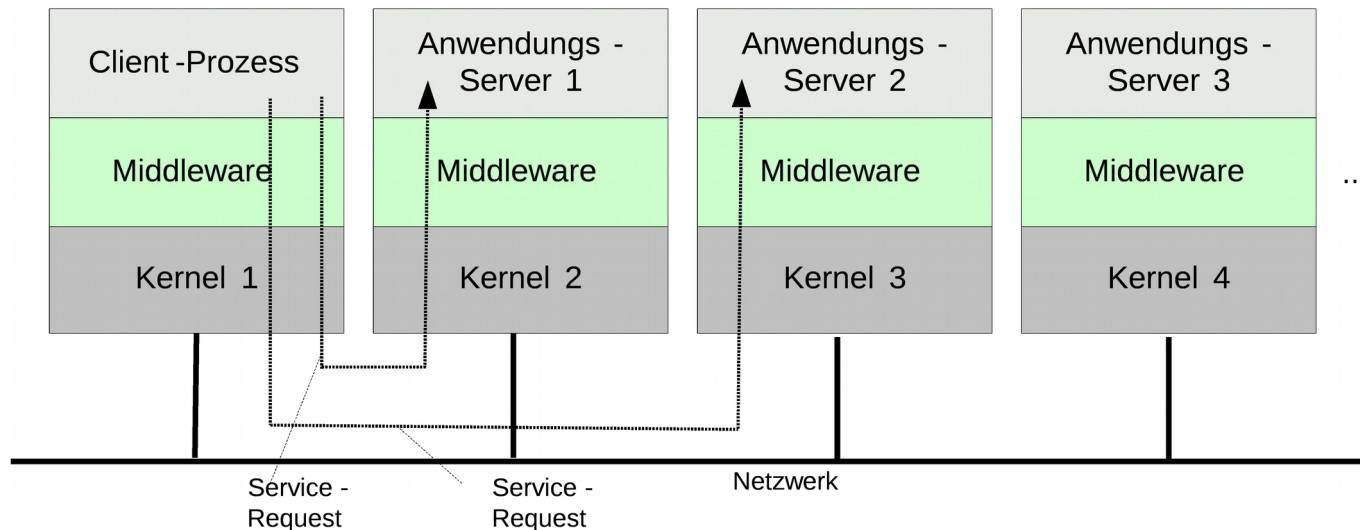
Superpeer:
BitTorrent, Skype



—————> Peer nutzt Dienst eines anderen Peers
-----> Dienst im Dienstverzeichnis anfragen

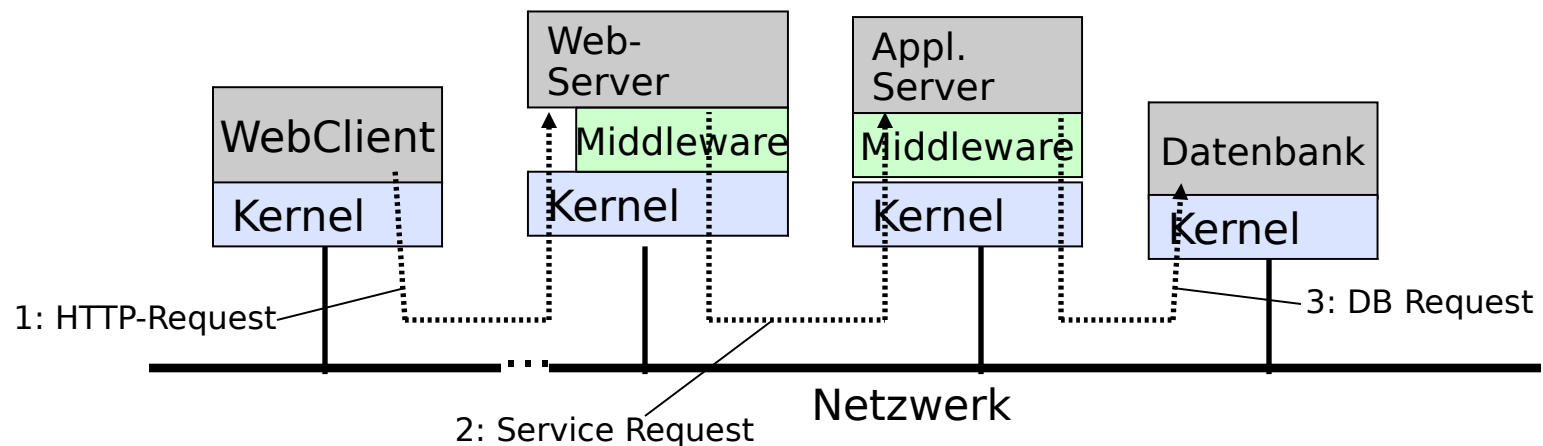
Middleware

- Die Kommunikations-Middleware übernimmt die Aufgaben der Client-/Server-Kommunikation (**verteilte Anwendung**)
- Verteilt werden Anwendungs-Clients und -Server
- Jeder Rechnerknoten verfügt über ein komplettes Betriebssystem



Beispiel: WWW-Anwendung

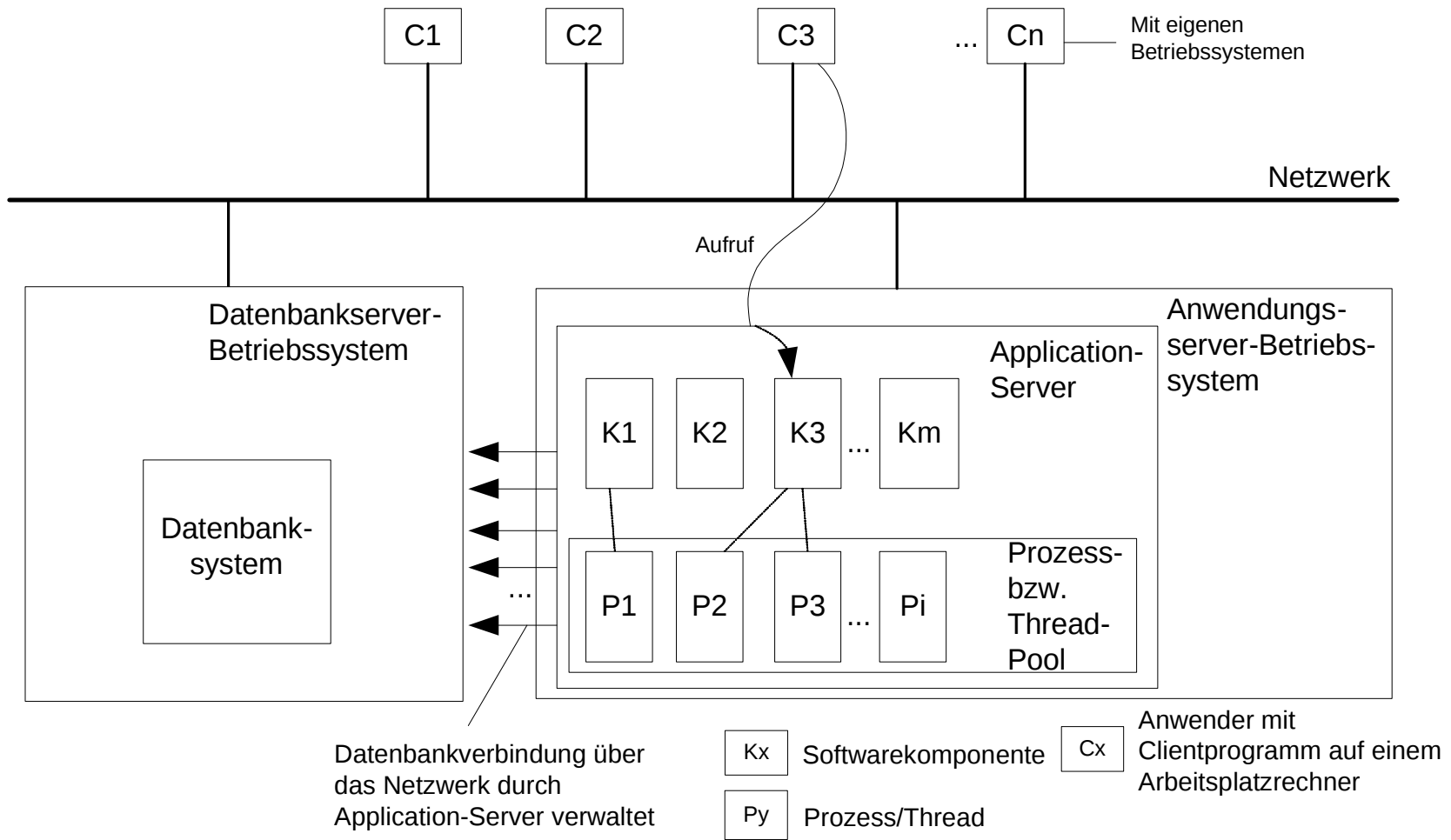
- Web-Anwendungen sind auch verteilte Anwendungen und benötigen mehrere Serversysteme
 - Web-Server
 - Application Server
 - Datenbanksystem
- Web-Client (Browser) erzeugt den Request über HTTP



Application-Server-Betrieb (1)

- In heutigen verteilten Systemen werden statt Transaktionsmonitoren oft auch sog. **Application-Server** verwendet (auch Middleware)
 - Derzeit aktuell: **EJB** Application Server (Enterprise Java Beans) und .NET
 - PHP-fpm, Rails (Passenger), node.js, ...
 - Stellen auch Plattformen für verteilte Komponentensysteme bereit
 - Beispiele: Oracle Weblogic AS, JBoss AS, IBM Websphere AS
- Verteilte Transaktionsverarbeitung ist aber wesentlich komplexer als lokale Transaktionsverarbeitung

Application-Server-Betrieb (2)



Überblick

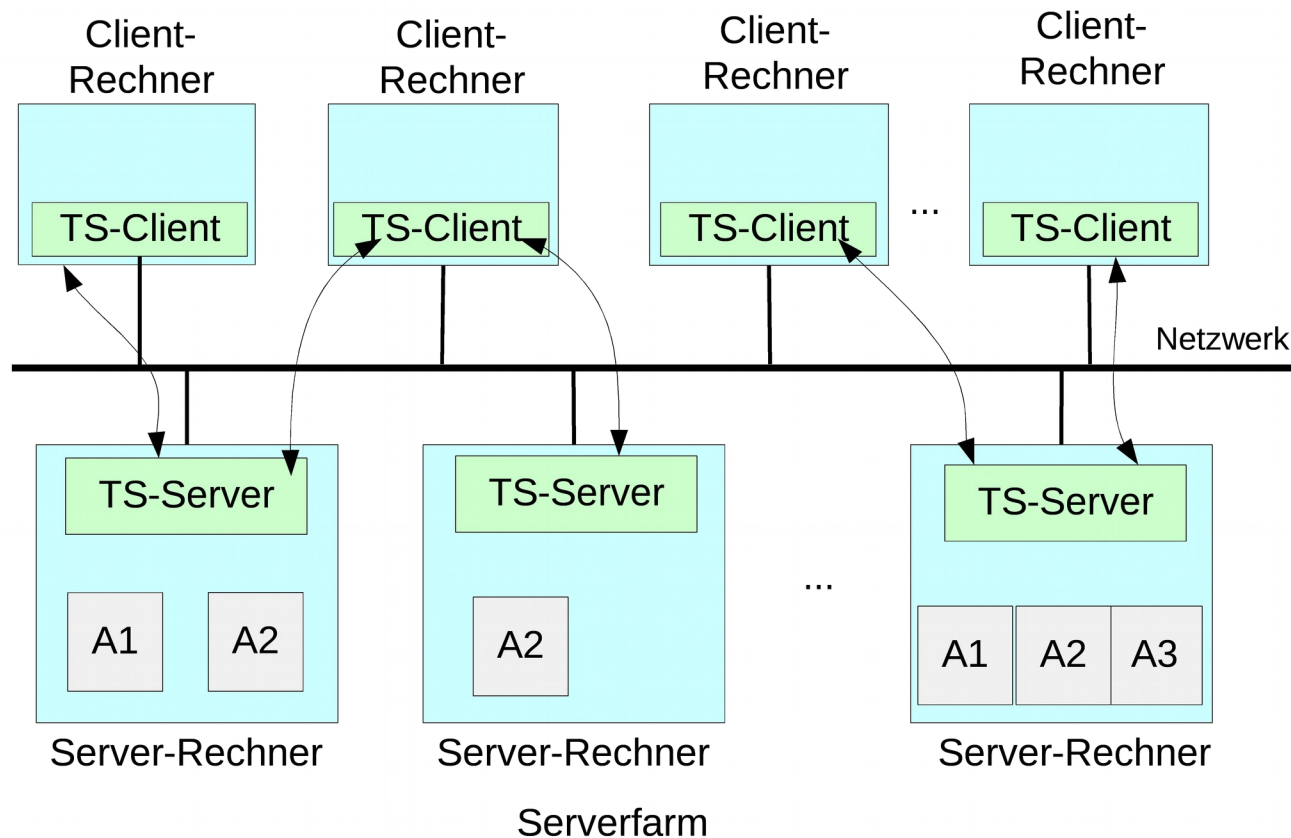
1. Zugriffsschutz in Betriebssystemen
2. Lokale Architekturen
3. Verteilte Verarbeitung
- 4. Terminalserver-Betrieb**
5. Virtualisierung von Betriebs- und Laufzeitsystemen
6. Cloud Computing

Terminalserver: Idee

- **Früher:** Anwendungen liefen in der Ablaufumgebung des Mainframes, Terminals waren „blockorientiert“ und dienten nur der Präsentation
 - „Dumme“ Terminals
- **Heute:** Clientrechner sind intelligent und enthalten clientseitige Anwendungskomponenten
- **Probleme:**
 - Verteilung von Programmänderungen auf Clientrechner ist teuer
 - Leichtere Angriffe auf dezentrale Systeme möglich (Sicherheitsaspekt)
 - ...
- Terminalserver dienen der **Reduzierung der Probleme**
- Bsp: RDP, VNC, X11-Clients, proprietär/custom, ...

Terminalserver-Architektur

Üblicherweise PCs, sog. Thin-Clients



Ax: Anwendung

TS-Client: Clientsoftware des Terminalservers

TS-Server: Serversoftware des Terminalservers

Überblick

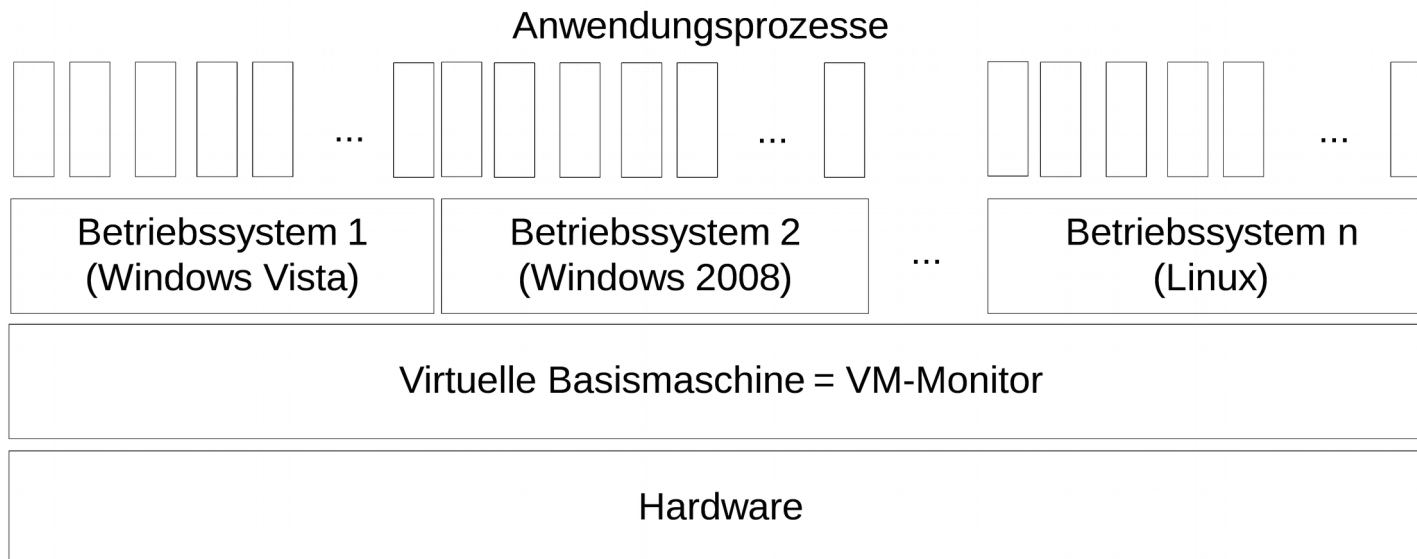
1. Zugriffsschutz in Betriebssystemen
2. Lokale Architekturen
3. Verteilte Verarbeitung
4. Terminalserver-Betrieb
- 5. Virtualisierung von Betriebs- und Laufzeitsystemen**
6. Cloud Computing

Was ist Virtualisierung?

- **Allgemeine Definition:**
 - Unter Virtualisierung versteht man Methoden zur Abstraktion von Ressourcen mit Hilfe von Software
- **Virtuelle Maschine verhält sich wie die reale Maschine**
- **Diverse Varianten:**
 - **Virtuelle Computer:**
Server- und Desktopvirtualisierung
(= Betriebssystem- bzw. Plattformvirtualisierung)
 - Storage Virtualisierung
 - Anwendungsvirtualisierung
 - Virtuelle Prozessumgebungen (Prozessmodell und virtueller Speicher)
 - Virtuelle Prozessoren: Java Virtual Machine (JVM)
 - Netzwerkvirtualisierung (vLAN)

Betriebssystemvirtualisierung

- Grobes Virtualisierungsmodell mit unterschiedlichen Ausprägungen (Hypervisor Typ 1, Typ2 ...)
- Mehr dazu später



Laufzeitsysteme virtualisieren

- Wird heute immer mehr eingesetzt!
- Beispiele:
 - Java Virtual Machine
 - .NET Common Language Runtime
- Bytecode wird meist interpretiert
- Vorteil
 - Programme werden unabhängig von der Systemplattform
 - Konsistenzgarantien

Überblick

1. Zugriffsschutz in Betriebssystemen
2. Lokale Architekturen
3. Verteilte Verarbeitung
4. Terminalserver-Betrieb
5. Virtualisierung von Betriebs- und Laufzeitsystemen
- 6. Cloud Computing**

Cloud Computing: Grundlegendes

- Cloud = Rechnerwolke
- Cloud-Computing-Ansatz: IT-Infrastruktur wird über ein Netzwerk (z.B. Internet) von einem Cloud-Anbieter zur Verfügung gestellt:
 - Rechenkapazität, Betriebssysteme, Datenspeicher, Software
- Man unterscheidet aus Betreibersicht:
 - Private Cloud
 - Public Cloud
 - Hybrid Cloud

Cloud Computing: Cloud-Dienstmodelle

- **IaaS = Infrastructure as a Service**
 - Cloud-Provider bietet Zugang zu virtualisierten Rechnern (inkl. BS) und Speichersystemen
→ Basis ist Betriebssystemvirtualisierung
 - Beispiele: Amazon Elastic Compute Cloud (EC2)
- **PaaS = Platform as a Service**
 - Cloud-Provider bietet Zugang zu Programmierungsumgebungen
 - Beispiele: Google App Engine, Windows Azure
- **SaaS = Software as a Service**
 - Cloud-Provider bietet Zugang zu Anwendungsprogrammen
 - Beispiele: Google Docs, Microsoft Cloud Services, Salesforce.com (CRM-System)

Überblick

- ✓ Einführung in Computersysteme
- ✓ Entwicklung von Betriebssystemen
- ✓ Architekturansätze
- 4. Interruptverarbeitung in Betriebssystemen
- 5. Prozesse und Threads
- 6. CPU-Scheduling
- 7. Synchronisation und Kommunikation
- 8. Speicherverwaltung
- 9. Geräte- und Dateiverwaltung
- 10. Betriebssystemvirtualisierung