Westfälische Hochschule
Gelsenkirchen Bocholt Recklinghausen
University of Applied Sciences

Verteilte Systeme
→ Einführung

Prof. Dr. (TU NN)
Norbert Pohlmann

Institut für Internet-Sicherheit – if(is)
Westfälische Hochschule, Gelsenkirchen
http://www.internet-sicherheit.de

Inhalt

- Definition
- Motivation
- Ziele von Verteilten Systemen
- Softwarekonzepte
- Beispiele von Verteilten Systemen
- Zusammenfassung

Inhalt

- **Definition**
- Motivation
- Ziele von Verteilten Systemen
- Softwarekonzepte
- Beispiele von Verteilten Systemen
- Zusammenfassung

Verteilte Systeme
→ Definition

- „Ein **Verteiltes System** ist eine Menge voneinander unabhängiger Rechnersysteme, die dem Benutzer wie ein **einzelnes System** erscheint.“

- Dies beinhaltet zwei Aspekte:
  - Die Rechnersysteme sind autonom (Hardware)
  - Die Benutzer haben den Eindruck, sie hätten es mit einem einzigen System zu tun (Software - virtuelle Maschine).

- Ein Verteiltes System baut auf ein Rechnernetz auf.

Verteilte Systeme
→ weitere „Definitionen“

- Selbständige Rechnersysteme bilden ein Verteiltes System, wenn sie in der Lage sind, im Verbund ein gemeinsames Problem zu lösen.

- Ein **Verteiltes System ist ein System**, in dem sich Hardware- oder Software-Komponenten auf vernetzten Rechnersystemen befinden und nur über den Austausch von Nachrichten kommunizieren und ihre Aktionen koordinieren.

- Ein Verteiltes System ist eines, bei dem der Ausfall eines Rechnersystems, von dem man vorher noch nie gehört hat, zum Ausfall des Gesamtsystem führt.

  → Das soll es nicht sein!

Inhalt

- Definition
- **Motivation**
- Ziele von Verteilten Systemen
- Softwarekonzepte
- Beispiele von Verteilten Systemen
- Zusammenfassung

Verteilte Systeme

→ Motivation: Nutzung global verfügbarer Ressourcen

- World Wide Web (WWW)-Zugriffe auf weltweite Wissensbestände
  - ist ein riesiges Verteiltes System, das aus Millionen von Clients und Servern besteht, die auf verknüpfte Dokumente zugreifen
- Nutzung von Hochleistungsrechnern oder Spezialsystemen
  - Supercomputer
  - Software-Bibliotheken
- „Aufsammeln“ ungenutzter Rechnerleitung im Internet
  - Faktorisierung großer Zahler in verteilter Weise
  - Brechen einer symmetrischer Verschlüsselung (z.B. DES)
- 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 nationaler Niederlassungen
  - internationale Spezialistenteams (Medizin, Forschung)

Verteilte Systeme
→ Vorteile (1/2)

- **Lastverteilung**
  - Die Dienstanfragen können gleichzeitig von verschiedenen Dienstanbietern befriedigt werden.
  - Dadurch kann ein Flaschenhals prinzipiell vermieden werden.
  - Ein großes Verteiltes System hat mehr aggregierte Rechnerleistung, als ein Großrechner.

- **Skalierbarkeit**
  - Gut programmierte verteilte Anwendungen können ohne Probleme schrittweise erweitert werden.

- **Ausfallsicherheit / Zuverlässigkeit**
  - Die räumliche Verteilung redundanter Komponenten erhöht die Ausfallsicherheit, indem bei Ausfall einer Komponente eine „identische Komponente“ an einem anderen Ort deren Aufgabe übernimmt.

Verteilte Systeme
→ Vorteile (2/2)

- **Wirtschaftlichkeit**
  - Eine monolithische Anwendung kann in der Regel nur auf einem Großrechner laufen.
  - Viele kleine durch das Verteilte System koordinierte PCs haben die gleiche Leistung oder sogar mehr, sind aber in der Anschaffung wesentlich preisgünstiger.
  - Das Preis-/Leistungsverhältnis ist bei Verteilten Systemen größer als bei der Verwendung von Großrechnern.

- **Flexibilität**
  - Eine funktionale Verbesserung eines Dienstangebots kommt unmittelbar allen Dienstnutzern zugute.
  - Da die Prozessorlast über ein gesamtes System verteilt werden kann, ist der einzelne Rechner entlastbar. Dies ist bei PDAs oder SmartPhones von besonderer Bedeutung!

- **Neue Anwendungen**
  - Durch die Möglichkeit zur Kommunikation unter den Rechnersystemen können neue Anwendungen und Dienste wie Videokonferenzen oder das Web eingesetzt werden.

Verteilte Systeme
→ Nachteile (1/2)

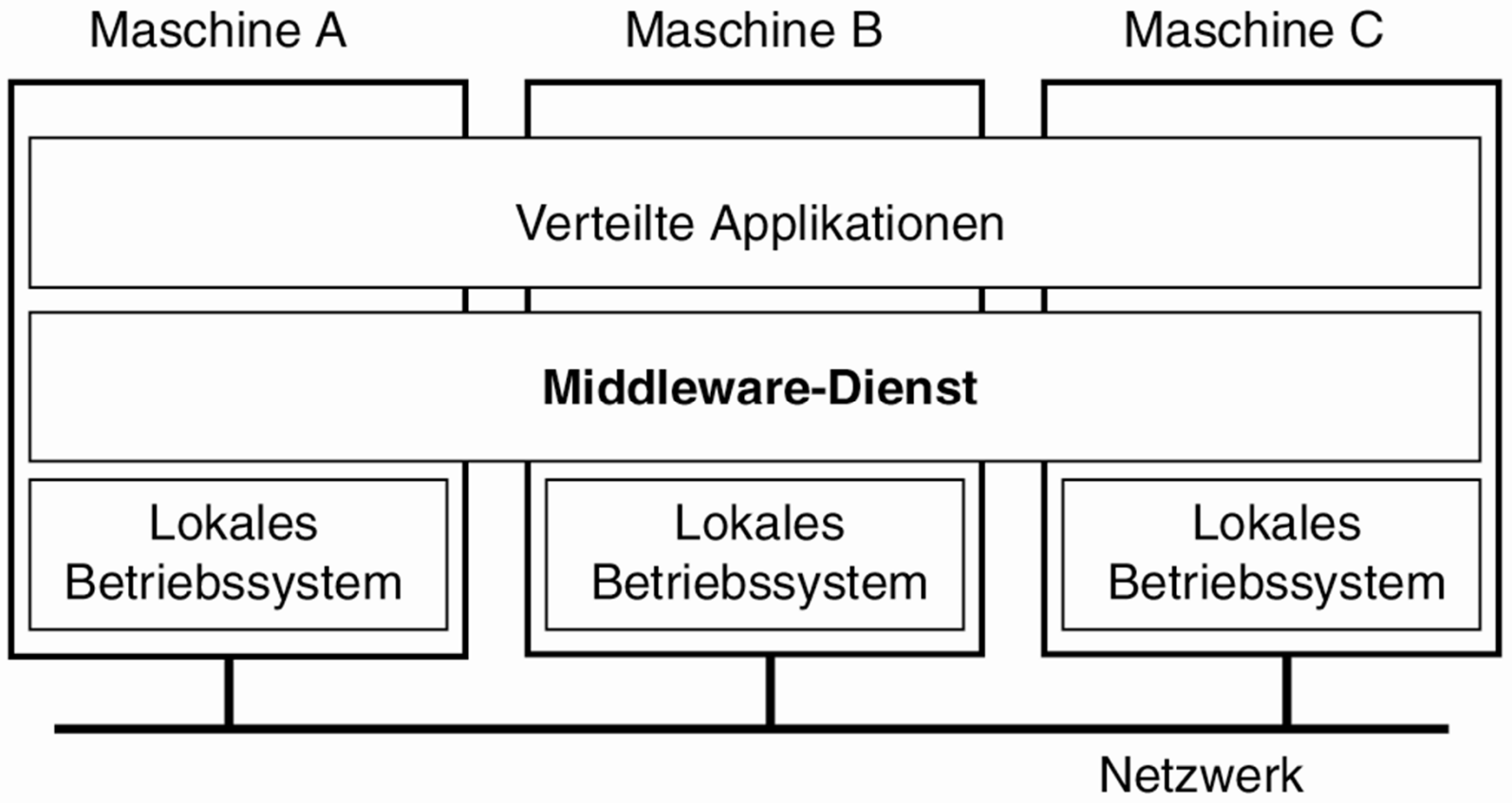
- **Keine globale Sicht**
  - Nur für wenige Eigenschaften eines Verteilten Systems ist eine globale Sicht möglich. Die Nutzung einer globalen Uhr zur Koordination ist begrenzt.
- **Nichtdeterminismus**
  - Die Verteilung kann zu Nichtdeterminismus (unvorhersagbarem Verhalten) führen, insbesondere bei Überlastung.
- **Verklemmung (deadlock)**
  - Eine Verklemmung entsteht, wenn mehrerer Komponenten gemeinsame Ressourcen beanspruchen, aber jeder auf die Freigabe einer Ressource von einer anderen Komponente angewiesen ist und wartet.
- **Verhungern (livelock)**
  - Das Verhungern entsteht, wenn eine Komponente eine Ressource beansprucht, sie aber nie zugewiesen bekommt, da jeweils andere Komponenten permanent bevorzugt werden.

Verteilte Systeme
→ Nachteile (2/2)

- **Verfügbarkeit**
  - Lange Kommunikationszeiten können die Verfügbarkeit einschränken, der Ausfall einzelner Komponenten kann den Ausfall des Gesamtsystems nach sich ziehen.
- **Programmierung**
  - Die Programmierung, insbesondere das Testen und die Fehlersuche sind wesentlich aufwändiger als bei Einzelanwendungen.
- **Komplexität**
  - Verteilte Systeme sind schwierig zu entwickeln, betreiben, beherrschen
  - Abstraktion als Mittel zur Beherrschung von Komplexität wichtig: (Schichtung (Kapselung, virtuelle Maschinen, ...), Modularisierung (Service, Mikrokerne, ...), „Transparenz“-Prinzip
- **Sicherheit**
  - Vertraulichkeit, Authentizität, Integrität, Verbindlichkeit, Verfügbarkeit,
  - notwendiger als in klassischen Systemen, da mehr Gefahren
  - aber schwieriger zu gewährleisten, da verteilte und unterschiedliche Hardware- und Software-Umgebung

Verteilte Systeme
→ Middleware

- Um heterogene Rechnersysteme und Netzwerke zu unterstützen, und gleichzeitig den Eindruck eines einzigen Systems zu präsentieren, werden verteilte Systeme häufig mithilfe einer Softwareschicht organisiert.
- Diese **Softwareschicht (Middleware)** liegt logisch zwischen einer Ebene des Benutzers und der Applikation und der Ebene des Betriebssystems.



Inhalt

- Definition
- Motivation
- **Ziele von Verteilten Systemen**
- Softwarekonzepte
- Beispiele von Verteilten Systemen
- Zusammenfassung

Verteilte Systeme
→ Ziele: Übersicht

- Damit ein **Verteiltes System** sinnvoll ist, sollen folgende Ziele beachtet werden:
  - Benutzer und Ressourcen verbinden
  - Transparenz
  - Offenheit
  - Skalierbarkeit

Verteilte Systeme
→ Ziele: Benutzer und Ressourcen verbinden

- Hauptziel eines Verteilten Systems ist die Möglichkeit, sowohl Hardwarekomponenten wie Drucker, Platten und Prozessoren, als auch **Daten und Dienste gemeinsam nutzen zu können**.

- Damit können Ressourcen kosteneffektiver und flexibel eingesetzt werden.

Verteilte Systeme
→ Ziele: Transparenz (1/4)

- In Abwandlung von der umgangssprachlichen Bedeutung des Wortes „Transparenz“ spricht man im Fall von Verteilten Systemen auch von **Verteilungstransparenz**.
- Für Verteilte Systeme sollte es im Idealfall nicht unterscheidbar sein, ob die Anwendung zentral und nur auf einem Rechnersystem, oder ob sie verteilt ist.
- Das Verteilte System stellt sich wie ein einzelnes Rechnersystem dar (virtuelle Maschine).
- Der Anwender bemerkt nicht, welches Rechnersystem tatsächlich einen (Teil-) Anwendungsdienst erbringt.
- Die Verteilungstransparenz kann in die folgenden Aspekte unterteilt werden:

Verteilte Systeme
→ Ziele: Transparenz (2/4)

Transparenzeigenschaften, welche die Verteilung verbergen sind (1/2):

- **Zugriffstransparenz**
  - Die Art und Weise, wie auf lokale und entfernte Komponenten zugegriffen wird, ist identisch.
- **Ortstransparenz (Positionstransparenz)**
  - Der Ort, an dem sich die Daten befinden oder an dem ein Programm ausgeführt wird, ist unsichtbar.
  - Die Ortstransparenz ermöglicht den Zugriff auf eine Ressource ohne Wissen um ihre physikalische Lokation.
- **Concurrency-Transparenz (Nebenläufigkeitstransparenz)**
  - Mehrere Benutzer/Prozesse können gemeinsame Objekte (z.B. Dateien) benutzen, ohne dass es zu Inkonsistenzen kommt.
  - Dem Anwender bleibt verborgen, dass er sich die Ressourcen mit anderen Anwendern teilt.

Verteilte Systeme
→ Ziele: Transparenz (3/4)

Transparenzeigenschaften, welche die Verteilung verbergen sind (2/2):

- **Skalierbarkeitstransparenz**
  - Das Verteilte System reagiert flexibel auf Erweiterungen sowie auf Änderungen und Modifikation der Hardware und Softwarebasis.
- **Parallelitätstransparenz**
  - Aktivitäten können parallel ablaufen, ohne sich zu stören.
- **Technologietransparenz**
  - Unterschiedliche Technologien, wie beispielsweise Hardware, Programmier-sprachen oder Betriebssysteme, werden vor dem Anwender verborgen.

Verteilte Systeme
→ Ziele: Transparenz (4/4)

Transparenzeigenschaften, welche die Verteilung des Systems zur Steigerung der Leistung und Fehlertoleranz ausnutzen sind:

- **Migrationstransparenz (Mobilitätstransparenz)**
  - Ressourcen können im System physikalisch bewegt werden, ohne ihren Namen und Zugriffsmechanismen zu ändern (z.B. Web-Seite, URL).
- **Leistungs- oder Performancetransparenz**
  - Kein (spürbarer) Leistungsunterschied zwischen lokaler und entfernter Bearbeitung.
- **Replikationstransparenz**
  - Es bleibt verborgen, ob ein Dienst durch das Original oder durch eines ihrer Relikate erbracht wird.
  - Unsichtbar, wie viele Replikate eines Objektes (z.B. Datei) existieren.
- **Fehler- oder Ausfalltransparenz**
  - Das Gesamtsystem wird nicht von einem Ausfall von Teilkomponenten beeinflußt. Die Daten bleiben konsistent.

Verteilte Systeme
→ Ziele: Offenheit

- In einem Verteilten System kann keine homogene Hardwareumgebung und einheitliche Software vorausgesetzt werden.

- Damit jedoch eine Zusammenarbeit stattfinden kann, müssen einige Bedingungen erfüllt sein.
  - Die Schlüsselschnittstellen der Hardware- und Softwarekomponenten müssen offengelegt sein.
  - Die Interprozesskommunikation auf Anwendungsebene muss einheitlich und offengelegt sein, damit verteilte Anwendungen miteinander Informationen austauschen können.

Verteilte Systeme
→ Ziele: Skalierbarkeit

- Soll ein Verteiltes System beliebig erweiterbar sein, so müssen einige wichtige Punkte beachtet werden.
  - Die Skalierbarkeit sollt für die Software transparent sein, damit nicht bei jeder Erweiterung Veränderungen an den Programmen notwendig sind.
  - Das Verteilte System darf „keine“ zentralen Komponenten benötigen, sonst bildet sich schnell ein Flaschenhals, da alle Rechnersysteme auf dieselbe Zentralkomponente zugreifen müssen.
  - Im Verteilten System dürfen auf Grund der gleichen Problematik „keine“ zentralen Daten und Tabellen gehalten werden.
  - Algorithmen dürfen nicht zentralisiert ausgelegt werden, d.h. die Ausführung sollte nicht an einen bestimmten Ort gebunden sein.

Inhalt

- Definition
- Motivation
- Ziele von Verteilten Systemen
- **Softwarekonzepte**
- Beispiele von Verteilten Systemen
- Zusammenfassung

Software-Konzepte
→ Überblick (1/2)

- Verteilte Systeme, wie herkömmliche Betriebssysteme, agieren als Ressourcen-Manager für die zugrunde liegende Hardware, sodass mehrere Benutzer und Applikationen Ressourcen (CPU, Speicher, Peripherie, Netzwerk, Daten, ...) gemeinsam nutzen können.

- Verteilte Systeme versuchen die Konflikte sowie die heterogene Natur der zu Grunde liegenden Hardware zu verbergen, indem sie eine **virtuelle Maschine** bereitstellen, auf der Applikationen ausgeführt werden können.

Software-Konzepte
→ Überblick (2/2)

Betriebssysteme für verteile Rechnersysteme und Middleware.

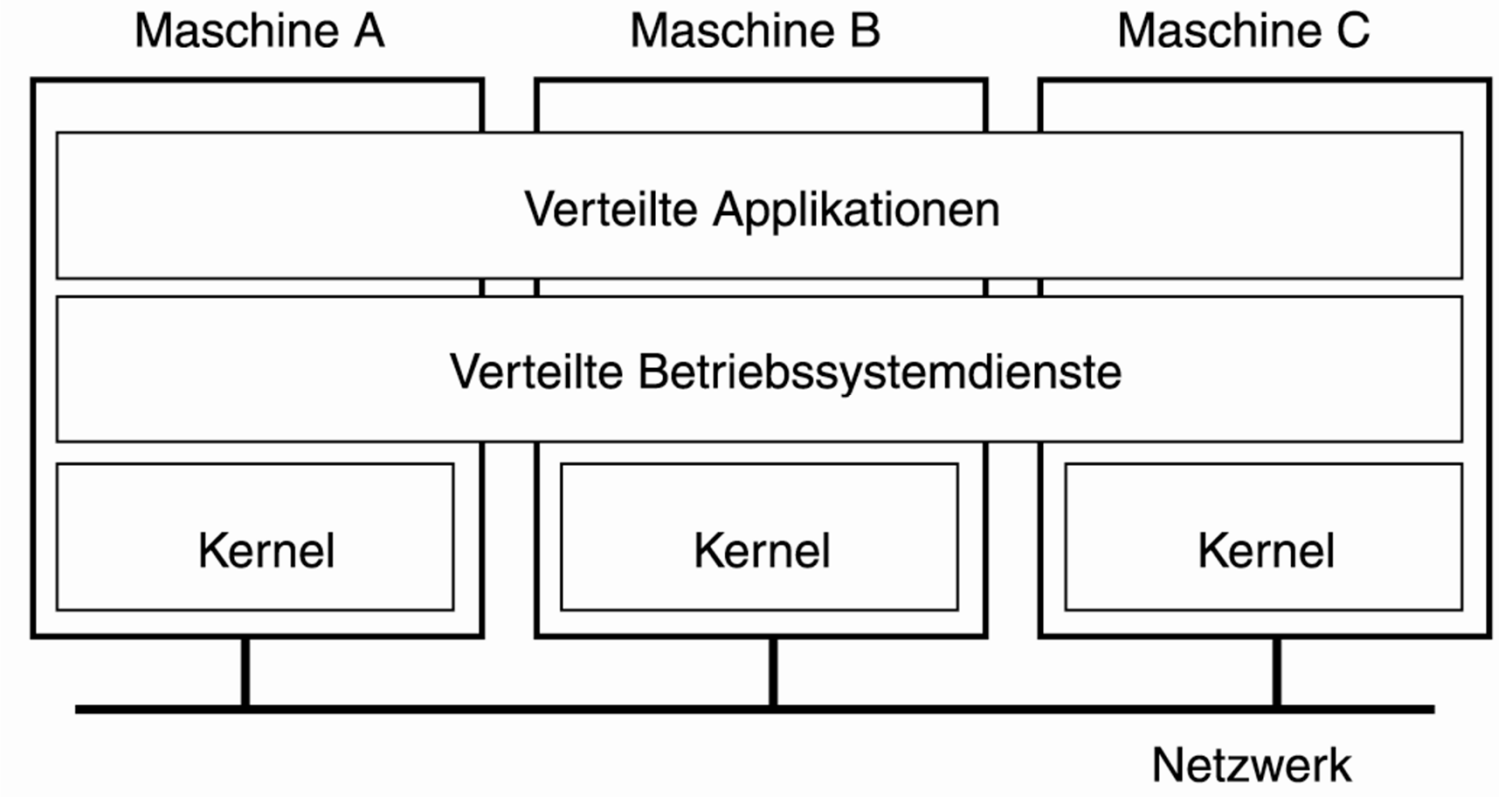
System	Beschreibung	Wichtigstes Ziel
DOS	Streng gekoppeltes Betriebssystem für Multiprozessoren und homogene Multicomputer	Hardware-Ressourcen verbergen und verwalten
NOS	Locker gekoppeltes Betriebssystem für heterogene Multicomputer (LAN und WAN)	Anbieten lokaler Dienste für entfernte Clients
Middleware	Zusätzliche Schicht über dem NOS, die allgemeine Dienste implementiert	Verteilungstransparenz erzielen

DOS: Distributed Operation System; **NOS:** Network Operation System

Software-Konzepte
→ Verteilte Betriebssysteme: Übersicht

- Es gibt zwei Gruppen von verteilten Betriebssystemen.
- Ein **Multiprozessor-Betriebssystem** verwaltet die Ressourcen eines Multiprozessors.
- Ein **Multicomputer-Betriebssystem** ist ein Betriebssystem, das für homogene Multicomputer entwickelt wurde.
- Die Funktionalität verteilter Betriebssysteme ist im Wesentlichen dieselbe wie die traditioneller Betriebssysteme für Einprozessorsysteme, außer dass sie mehrere CPUs verwalten.
- Betriebssysteme für Multicomputer haben eine völlig andere Struktur und Komplexität als Multiprozessor-Betriebssysteme.
- Die Datenstrukturen für die systemweite Ressourcenverwaltung kann nicht durch einen gemeinsam genutzten physikalischen Speicher geschehen, sondern muss über eine Kommunikationsverbindung durch einen Nachrichtenaustausch bewerkstelligt werden.

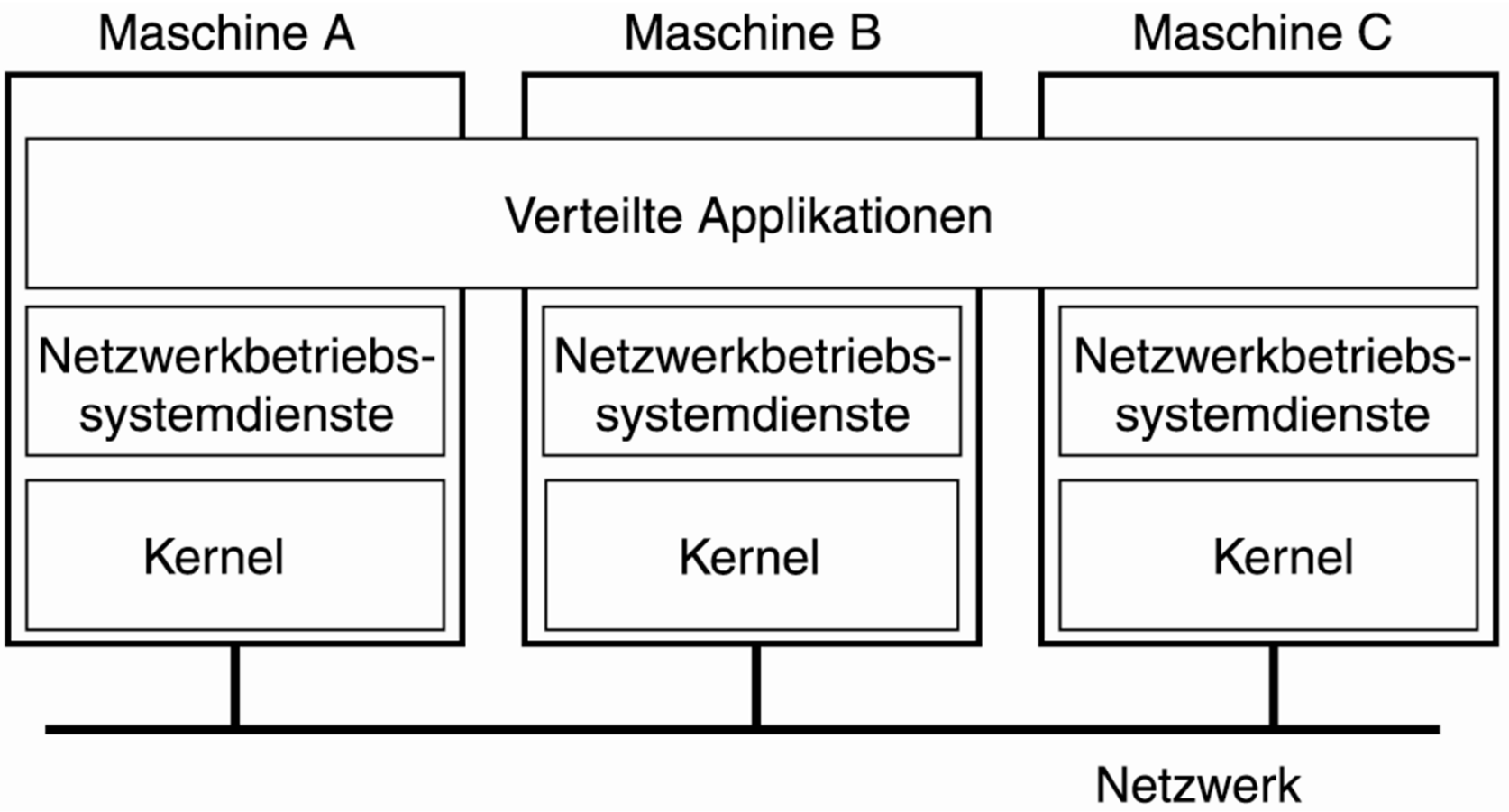
Software-Konzepte
→ Verteilte Betriebssysteme: Multicomputer BS



- Jeder Knoten hat einen eigenen Kernel, der Module für die Verwaltung lokaler Ressourcen enthält.
- Außerdem besitzt jeder Knoten ein separates Modul für die Verarbeitung der Kommunikation zwischen den Prozessoren.
- Oberhalb jedes lokalen Kernels befindet sich eine gemeinsame Software-Schicht, die das Betriebssystem als virtuelle Maschine implementiert, die die parallele und nebenläufige Ausführung verschiedener Aufgaben unterstützt.

Software-Konzepte
→ Netzwerkbetriebssysteme (1/2)

- Im Gegensatz zu verteilten Betriebssystemen setzen Netzwerkbetriebssysteme nicht voraus, dass die zugrunde liegende Hardware homogen ist und dass sie so verwaltet werden sollte, als handelt es sich dabei um ein einzelnes System.
- Stattdessen werden sie im Allgemeinen aus mehreren Einzelprozessor-Systemen zusammengesetzt, die alle ein eigenes Betriebssystem besitzen.



Software-Konzepte
→ Netzwerkbetriebssysteme (2/2)

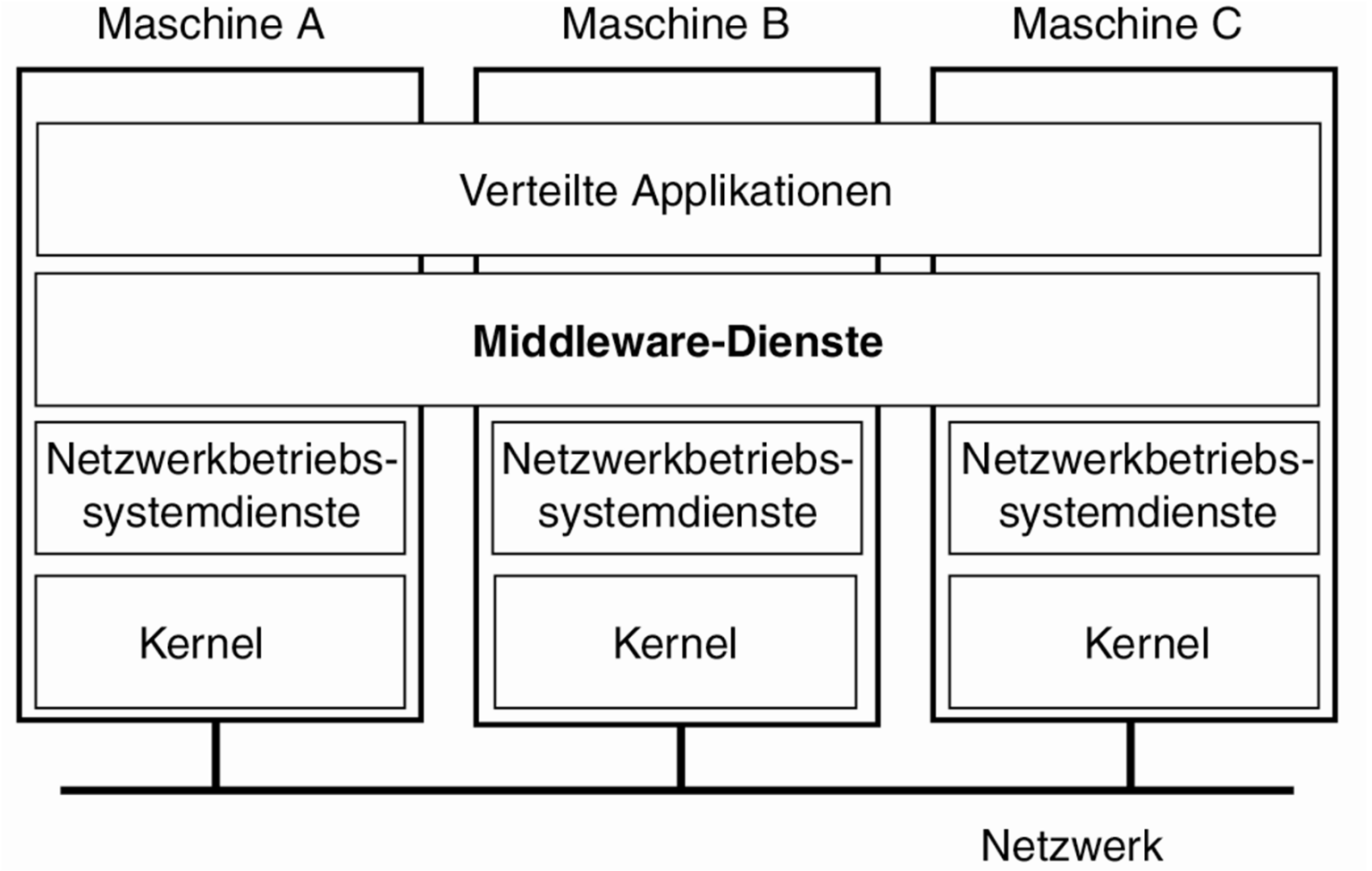
- Die Maschinen und ihre Betriebssysteme können sich unterscheiden, aber sie sind alle in einem Rechnernetzwerk miteinander verbunden.

- Darüber hinaus unterstützen Netzwerkbetriebssysteme Funktionen, die es den Benutzern erlauben, die auf einer bestimmten Maschine verfügbaren Dienste zu nutzen.

- Beispiele solcher Dienst sind:
  - Remote Login
  - Remote Copy
  - Globales Dateisystem (z.B. NFS)
  - usw.

Software-Konzepte
→ Middleware

- Verteilte Systeme arbeiteten mit einer zusätzlichen Software-Schicht, der Middleware, um die Heterogenität der vielen verschiedenen zugrunde liegenden Plattformen mehr oder weniger zu verbergen, aber auch, um sich um Verteilungsaspekte zu kümmern.



Software-Konzepte
→ Vergleich zwischen den Systemen

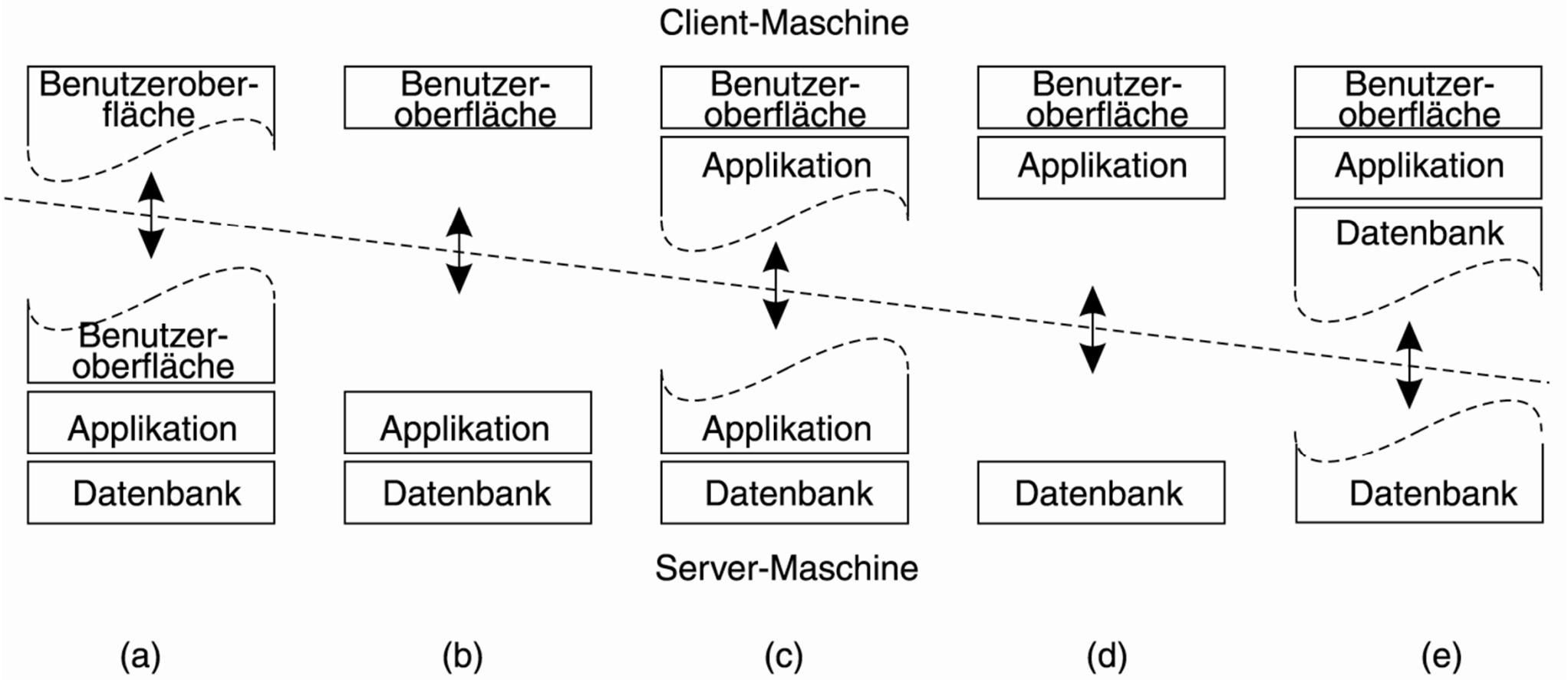
Aspekt	Verteilte Betriebs-systeme		Netzwerk-betriebs-system	Middleware-basiertes Betriebssystem
	Multi-prozessoren	Multi-computer		
Transparenzgrad	Sehr hoch	Hoch	Gering	Hoch
Dasselbe Betriebssystem auf allen Knoten?	Ja	Ja	Nein	Nein
Anzahl der Kopien des Betriebssystems	1	N	N	N
Kommunikationsbasis	Gemeinsam genutzter Speicher	Nach-richten	Dateien	Modell-spezifisch
Ressourcenverwaltung	Global, zentral	Global, verteilt	Pro Knoten	Pro Knoten
Skalierbarkeit	Nein	Moderat	Ja	Variiert
Offenheit	Geschlossen	Geschlos-sen	Offen	Offen

Das Client-Server-Modell
→ Ziel

- Es gibt viele Probleme und Ideen dazu, Verteilte Systeme zu realisieren.

- Bei einem Problem sind sich viele Forscher und Praktiker einig:
  - Wenn Clients die Dienste von Servern anfordern, dann hilft das sehr, die Komplexität Verteilter Systeme zu verstehen und zu verwalten.
  - Von daher sollen Verteilte Systeme auf dem Client-Server-Modell aufbauen.
  - Alle wichtigen Internet-Dienste bauen z.T. auf dem Client-Server-Modell auf (HTTP, FTP, Telnet, E-Mail-Protokolle (SMTP, POP3, IMAP), DNS, ...).

Das Client-Server-Architekturen
→ Mehrschichtige Architekturen (Multitier)

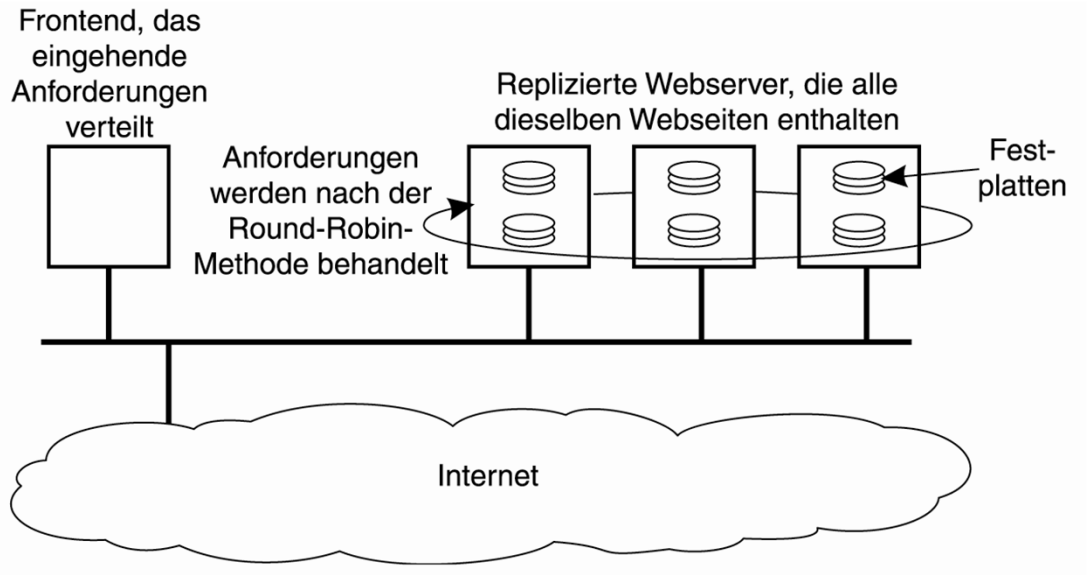


Moderne Architekturen
→ Übersicht (1/2)

- Mehrschichtige Client-Server-Architekturen sind eine direkte Folge der Unterteilung von Anwendungen in eine Benutzeroberfläche, Verarbeitungskomponenten und einer Datenebene.
- Die unterschiedlichen Schichten entsprechen direkt den logischen Anordnungen der Applikationen.
- In vielen Geschäftsumgebungen ist die verteilte Verarbeitung äquivalent zur Anordnung einer Client-Server-Applikation als mehrschichtige Architektur.
- Dies Art der Verteilung wird auch **vertikale Verteilung** genannt.
- Das charakteristische Merkmal der vertikalen Verteilung ist, dass sie erzielt wird, indem logisch unterschiedliche Komponenten auf unterschiedlichen Maschinen angeordnet werden.
- Der Begriff bezieht sich auf das Konzept der vertikalen Fragmentierung, wie es in verteilten relationalen Datenbanken verwendet wird, wo es ausdrückt, dass auf Tabellen nach Spalten unterteilt und dann auf verschiedenen Maschinen verteilt werden.

Moderne Architekturen
→ Übersicht (2/2)

- Die vertikale Verteilung stellt jedoch nur eine Möglichkeit dar, Client-Server-Applikationen anzuordnen, und ist in vielen Fällen auch die am wenigsten interessante Lösung.
- In modernen Architekturen ist es häufig die Verteilung der Clients und Server, die zählt, auch als **horizontale Verteilung** bezeichnet.
- Bei dieser Art der Verteilung können ein Client oder ein Server physikalisch in logisch äquivalente Teile unterteilt werden, aber jeder Teil arbeitet mit einem eigenen Anteil der vollständigen Datenmenge, sodass die Gesamtlast ausgeglichen ist.
- Eine gebräuchliche horizontale Verteilung ist die eines Webservers, der über mehrere Rechnersysteme in einem lokalen Rechnernetzwerk repliziert ist.



Inhalt

- Definition
- Motivation
- Ziele von Verteilten Systemen
- Softwarekonzepte
- **Beispiele von Verteilten Systemen**
- Zusammenfassung

Verteilte Systeme
→ Beispiele

- **Verteilte objektbasierte Systeme**
  - CORBA (Common Object Request Broker Architecture)
  - Distributed COM (MS)
  - Globe (Global Object-Based Environment - Forschungsprojekt - NL), ...
- **Verteilte Dateisysteme**
  - Sun Network File System (NFS)
  - Das Dateisystem von Coda (z.B. in Linux), ...
- **Verteilte dokumentbasierte Systeme**
  - Das World Wide Web
  - Lotus Notes, ...
- **Verteilte koordinationsbasierte Systeme**
  - TIB/Rendezvous (TIBCO)
  - Jini (SUN - JaveSpaces), ...

Inhalt

- Definition
- Motivation
- Ziele von Verteilten Systemen
- Softwarekonzepte
- Beispiele von Verteilten Systemen
- **Zusammenfassung**

Einführung: Verteilte Systeme
→ Zusammenfassung (1/3)

- Verteilte System bestehen aus autonomen Rechnersystemen, die zusammenarbeiten, um das Erscheinungsbild eines einzigen, hohärenten Systems zu präsentieren.
- Ein wichtiger Vorteil ist, dass sie es vereinfachen, unterschiedliche Anwendungen zu integrieren, die auf unterschiedliche Rechnersysteme innerhalb eines einzigen System ausgeführt werden.
- Ein weiterer Vorteil ist, dass Verteilte Systeme, wenn sie korrekt entworfen sind, im Hinblick auf das zugrunde liegende Rechnernetzwerk gut skaliert werden können.
- Diese Vorteile gehen häufig auf Kosten einer **komplexeren Software**, einer **verschlechterten Leistung** und häufig auch einer **schwächeren Sicherheit**.
- Nichtsdestotrotz besteht weltweit ein hohes Interesse, Verteilte Systeme zu erstellen und zu installieren.

Einführung: Verteilte Systeme
→ Zusammenfassung (2/3)

- Moderne Verteilte Systeme werden häufig mithilfe einer zusätzlichen Software-Schicht aufgebaut, die auf einem Netzwerkbetriebssystem aufsetzt.
- Diese Schicht, auch als **Middleware** bezeichnet, soll die Heterogenität sowie die verteilte Natur der zugrunde liegenden Rechnersysteme verbergen.
- Auf Middleware basierende Verteilte Systemen übernehmen im Allgemeinen ein spezielles Modell, um Verteilung und Kommunikation auszudrücken.
- Gebräuchliche Modelle basieren auf entfernten Prozeduraufrufen, verteilten Objekten, Dokumenten und Dateien.
- Wichtig für jedes Verteilte System ist seine interne Anordnung.
- Ein allgemein gebräuchliches Modell ist, das Client-Prozesse Dienste bei Server-Prozessen anfordern.
- Ein Client sendet eine Nachricht an einen Server und wartet, bis dieser eine Antwort zurückgibt.

Einführung: Verteilte Systeme
→ Zusammenfassung (3/3)

- Eine weitere Verfeinerung wird häufig getroffen, indem zwischen einer Benutzeroberflächenebene, einer Verarbeitungsebene und einer Datenebene unterschieden wird.
- Der Server ist im Allgemeinen für die Datenebene verantwortlich, während die Ebene der Benutzeroberfläche auf der Client-Seite implementiert wird.
- Die Verarbeitungsebene kann auf dem Client, auf dem Server, oder unterteilt auf beiden implementiert werden.
- Für moderne Verteilte Systeme ist diese vertikale Anforderung von Client-Server-Applikationen nicht ausreichend, um große Systeme erstellen zu können.
- Es wird eine **horizontale Verteilung** gebraucht, bei der Clients und Server über mehrere Rechnersysteme verteilt und repliziert sind.
- Ein typisches Beispiel, in denen die horizontale Verteilung erfolgreich angewendet wurde, ist das World Wide Web.

Westfälische Hochschule
Gelsenkirchen Bocholt Recklinghausen
University of Applied Sciences

Verteilte Systeme
→ Einführung

Vielen Dank für Ihre Aufmerksamkeit
Fragen ?

Prof. Dr. (TU NN)
Norbert Pohlmann

Institut für Internet-Sicherheit – if(is)
Westfälische Hochschule, Gelsenkirchen
http://www.internet-sicherheit.de

if(is) internet-sicherheit.