5. Netze, Verteilung

5.1 Verteilte Systeme - Warum

- ☆ Realisierung autonomer, kooperierender Systeme
- ☆ Anpassung an geographisch verteilte Datenhaltungs- und verarbeitungsbedürfnisse
- ☆ Leistungssteigerung durch parallele Bearbeitung von Aufgaben
- ☆ Erhöhung der Zuverlässigkeit eines Systems

5.2 Kommunikationskonzepte

Kanal/Mailbox

- ❍ Aufbau eines realen oder virtuellen Kanals zwischen Sender und Empfänger
- ❍ Asynchrone, gepufferte Übermittlung nicht adressierter Nachrichten (Mailbox-Paradigma)
- ❍ Kommunikationssystem stellt Dienste bereit für
  - ❍ Kanaleröffnung / -schließung
  - ❍ Versand
  - ❍ Benachrichrichtigung
  - ❍ Empfang

Globaler Versand (Broadcast)

- Alle Knoten sind über einen Bus oder Ring verbunden
- Senderknoten setzt Nachricht ab
- Nachricht enthält Ziel- und/oder Quellenadresse
- Kommunikationssystem
  - verteilt Nachricht an alle
  - verwaltet das Medium und verhindert Kollisionen oder löst sie auf
- Jeder Knoten betreibt einen Mithörprozess, der
  - die Adressen jeder Nachricht mithört
  - bei Zieladressierung die für den Knoten bestimmten Nachrichten empfängt
  - bei Quellenadressierung aufgrund einer Adressentabelle die für ihn interessanten Nachrichten empfängt

Fernaufruf (Remote Procedure Call RPC / Remote Method Invocation RMI)

- Analogon zu lokalem Prozedur- /Methodenaufruf
- Softwaretechnisch sehr einfach und klar (verbirgt die Verteilung gegenüber den benutzenden Komponenten weitgehend)
- Kommunikationssystem übernimmt Benachrichtigung des Empfängers, Übertragung der Parameter und der Ergebnisse
- Sender wartet auf Ergebnis (synchrone Kommunikation)
- Kommunikationssystem stellt Dienste bereit zum Aufrufen, sich Aufrufen lassen und einen Namensdienst (wer ist aufrufbar)
- Adressat eines RPC/RMI muss ständig verfügbar sein – Problem des Ausfalls muss gelöst werden

WWW

Drei Kommunikationstypen:

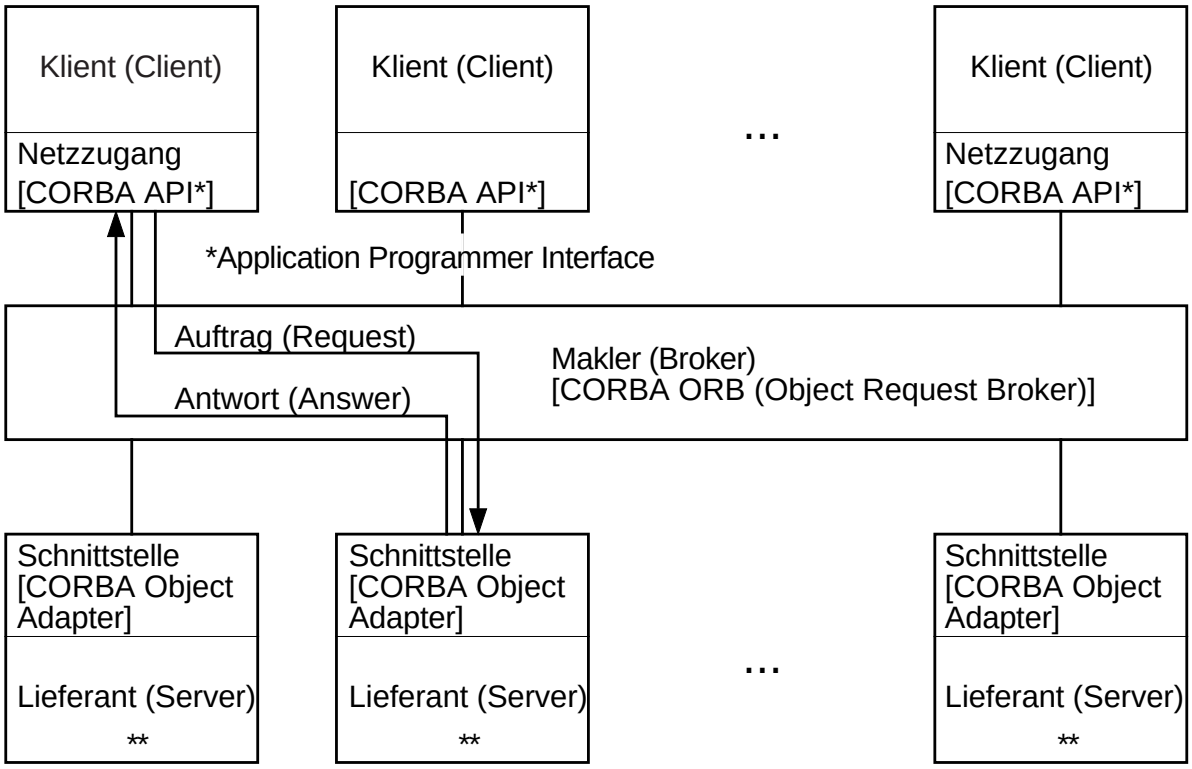
- Synchroner Dateitransfer :Klassisches Laden einer Seite von einem über eine URL adressierten Server
- Synchroner Dienstaufruf (analog RPC)
  - Bearbeitung einer Anfrage (Suchmaschinen, Auskunftsdienste, ...)
  - Versand der Anfrage zum Server, Bearbeitung auf Server, Rückversand von Ergebnissen
  - In der Regel mit CGI-BIN
- Dynamisches Laden von verteiltem Programmcode
  - Dynamisches Laden eines Programms von einem Server
  - Lokales Ausführen dieses Programms in geschützter Umgebung
  - Das Programm kann (unsichtbar für Klient) über das Netz kommunizieren
  - zum Beispiel Java-Applets

5.3 Verteilte Architekturen

Verteilte Objekte – Client/Server

- Geographisch verteilte Komponenten kommunizieren über Makler miteinander
- Komponenten werden über Schnittstelle angesprochen
- Schnittstelle ist programmiersprachen- und implementierungsunabhängig definiert
- Komponenten sind stark entkoppelt: kennen weder die Implementierung noch die geographische Lokalisierung der Partnerkomponenten.
- Beispiel: CORBA
- Realisiert häufig eine Client/Server-Architektur oder Middleware-Architektur

Client/Server-Architektur unter Verwendung von CORBA



**Typisch Datendienste wie Datenbanksystem(e), Dateisystem(e), Bildarchiv(e),...

Middleware-Architekturen

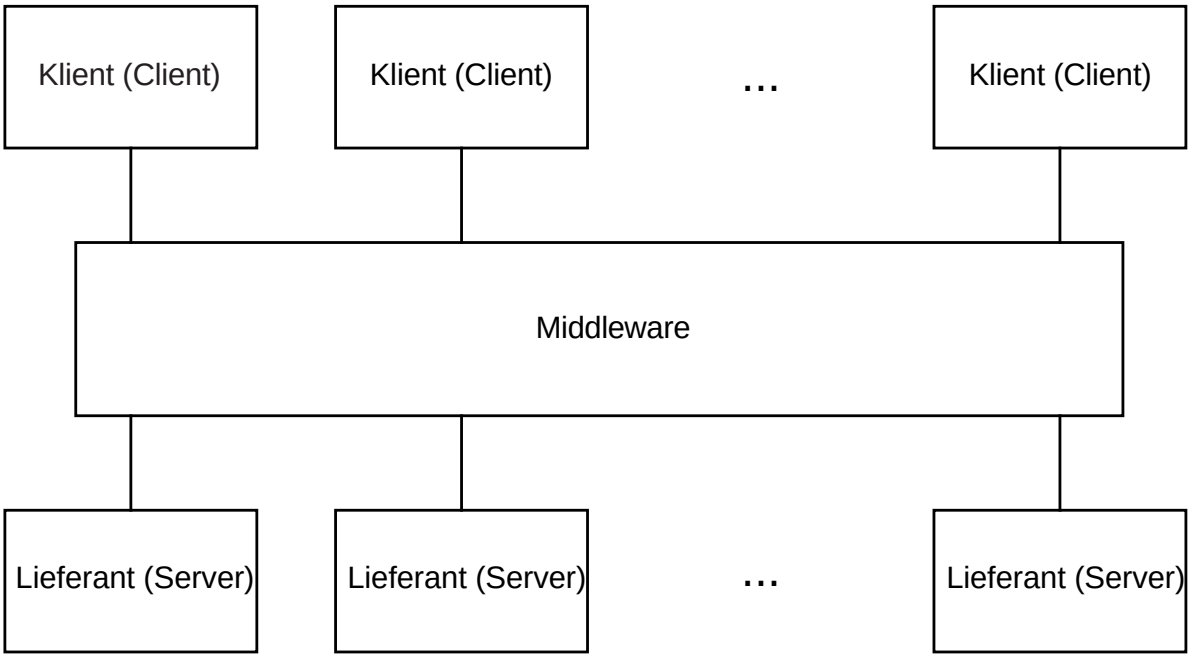
Drei-Schicht-Architekturen:

- ☆ Klienten (Clients)
- ☆ Dienste (Middleware)
- ☆ Lieferanten (Servers)

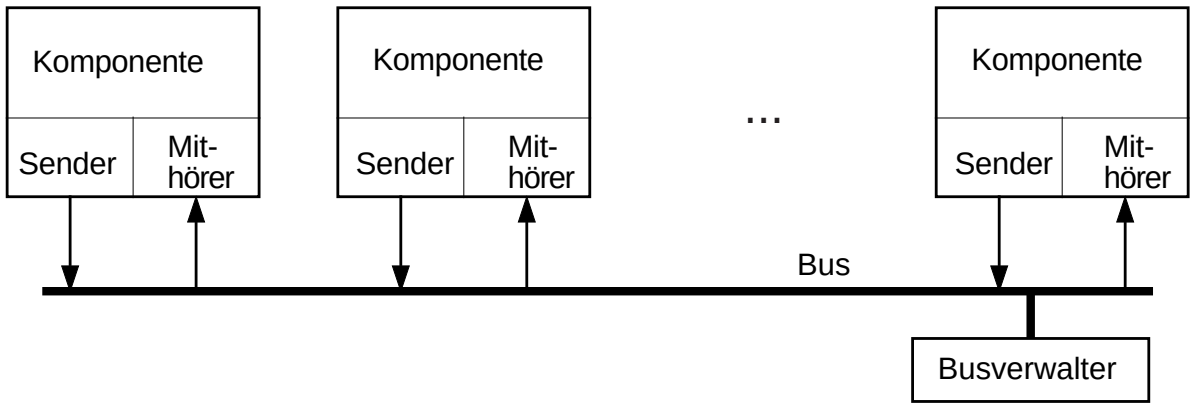
Middleware-Dienste:

- ❍ Kommunikation
- ❍ Schutz und Sicherheit
- ❍ Informationsdienste: Verwaltung, Aufbereitung, Koordination
- ❍ Ablaufsteuerung
- ❍ Präsentation
- ❍ Berechnungen

Prinzip einer Middleware-Architektur



Bus-Architektur



- Steckbare Komponenten auf Bus
- Globaler Nachrichtenversand über Bus
- Sehr flexibel, im laufenden Betrieb änderbar
- Einsatz u.a. in der industriellen Prozessleittechnik