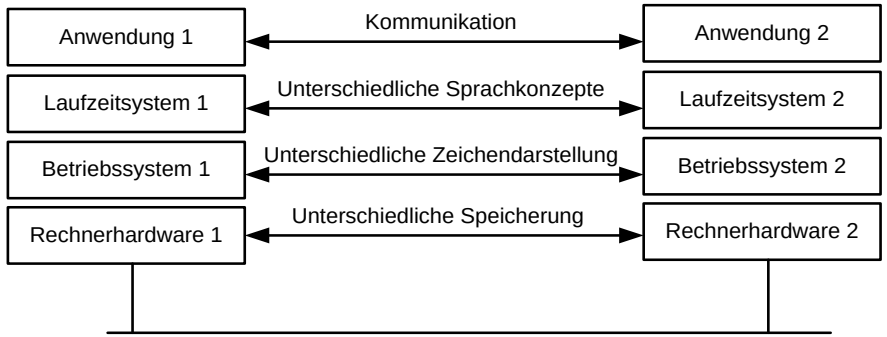
Ausgewählte Implementierungsaspekte von Client-Server-Systemen

- Wir betrachten im Weiteren einige ausgewählte Aspekte:
  - Heterogenität
  - Serverarchitektur
  - Nebenläufigkeit im Server (Parallelität)
  - Serverseitige Service- bzw. Dienstschnittstellen
  - Fehlersituationen, Fehlerklassierung
  - Parameterübergabe zwischen Client und Server
  - Marshalling/Unmarshalling
  - Kommunikation
  - Zustandsverwaltung
  - Garbage Collection
  - Lastverteilung, Verfügbarkeit, Skalierbarkeit

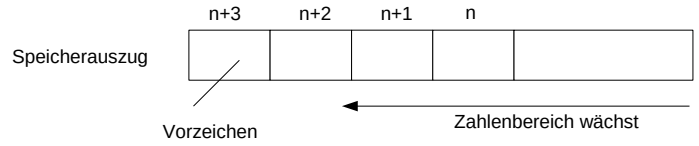
Heterogenität

- Mehrere Ebenen der Heterogenität
- Standardformate notwendig!

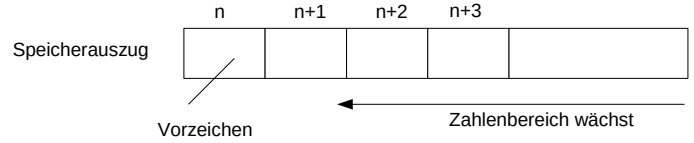


- **Rechnerhardware und Betriebssysteme**
  - Unterschiede bei der Speicherung der Daten
    - «Little Endian» versus «Big Endian»
  - Unterschiedliche Zeichensätze
    - ASCII - EBCDIC - Unicode

Darstellung: "little endian"

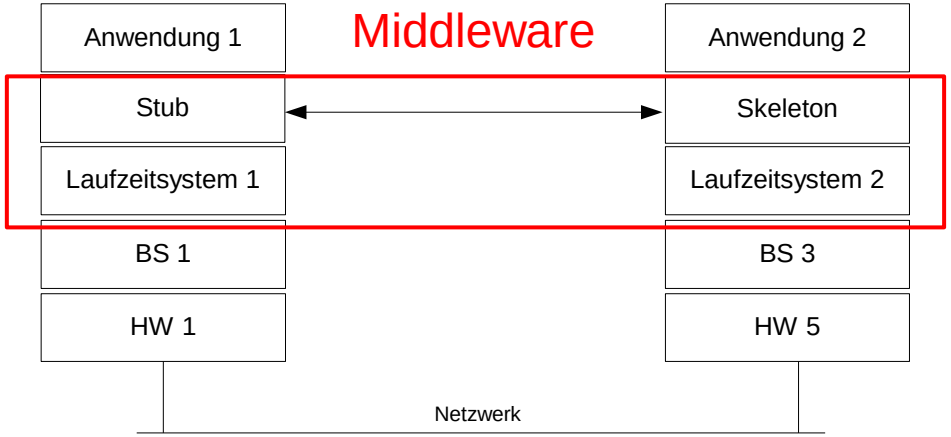


Darstellung: "big endian"



Überlegungen zur Überwindung von Heterogenität

- Was wir brauchen!
  - Einheitliche Transportsyntax (ASN.1, XDR, HTML, XML, JSON …) → Schicht 6 (ISO/OSI-Modell)
  - Middleware-Technologien bieten meist ähnliche Ansätze
  - Marshalling (Serialisierung) und Unmarshalling (Deserialisierung) der Nachrichten über generierten Code (Stubs und Skeletons)

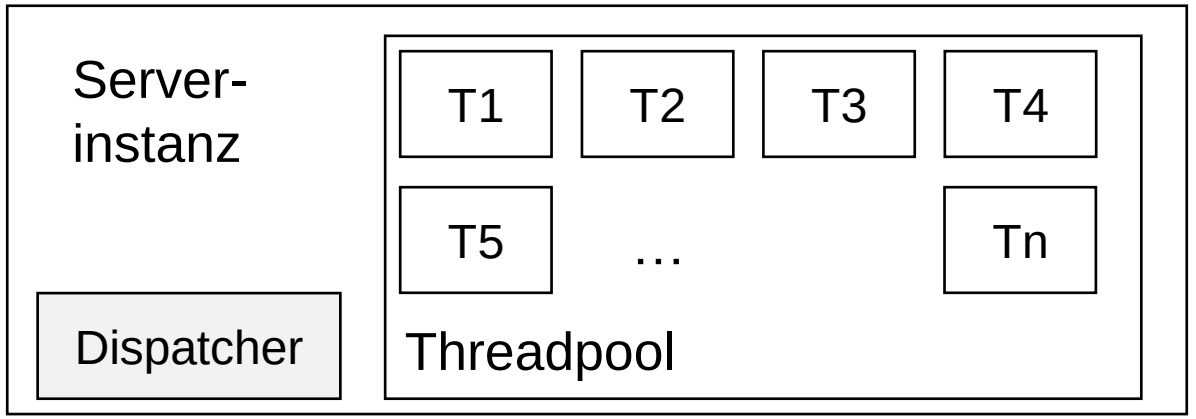


Nebenläufigkeit (Parallelität)

- Iterative (sequentielle) oder parallele Serverbausteine
- Threadpooling, Multithreading für die Bedienung mehrerer Clients gleichzeitig
- Ein Dispatcher ist ein Softwarebaustein im Server, der alle Requests der Clients entgegennimmt und sie auf Threads verteilt
- Einfaches sequentielles Programmiermodell für die Programmierer-Sicht
- Im JDK gibt es verschiedene Klassen für Thread-Pooling (s. java.util.concurrent)

Innenleben eines Servers

*Allg.: **Pooling** von Ressourcen = Vorbereiten zur schnelleren Nutzung*

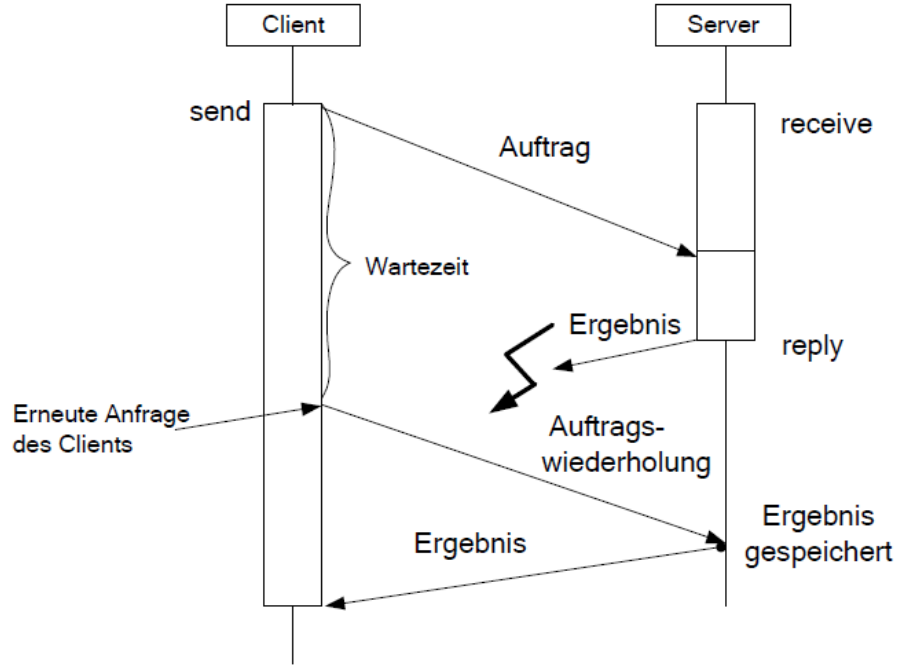


Dienst- bzw. Serviceschnittstellen

- Wie wird die Schnittstelle (Parameter- und Rückgabewertetypen) eines Serverbausteins beschrieben?
  - Neutrale Schnittstellenbeschreibungssprache oder eingebettet in Hostsprache (sprachabhängig)
  - Exception-Behandlung nicht immer gleich
- Diskussionsfrage:
  - Wie gut muss ein Server, der einen Service bereitstellt, prüfen, ob die empfangenen Parameter korrekt sind?

Fehlersituationen

- Es kann u.a. passieren, dass
  - ein Auftrag (engl. request) verloren geht,
  - das Ergebnis (engl. reply) des Servers verloren geht,
  - der Server während der Ausführung des Auftrags abstürzt,
  - der Server für die Bearbeitung des Auftrags zu lange braucht oder
  - der Client vor Ankunft des Ergebnisses abstürzt.



Parameterübergabe

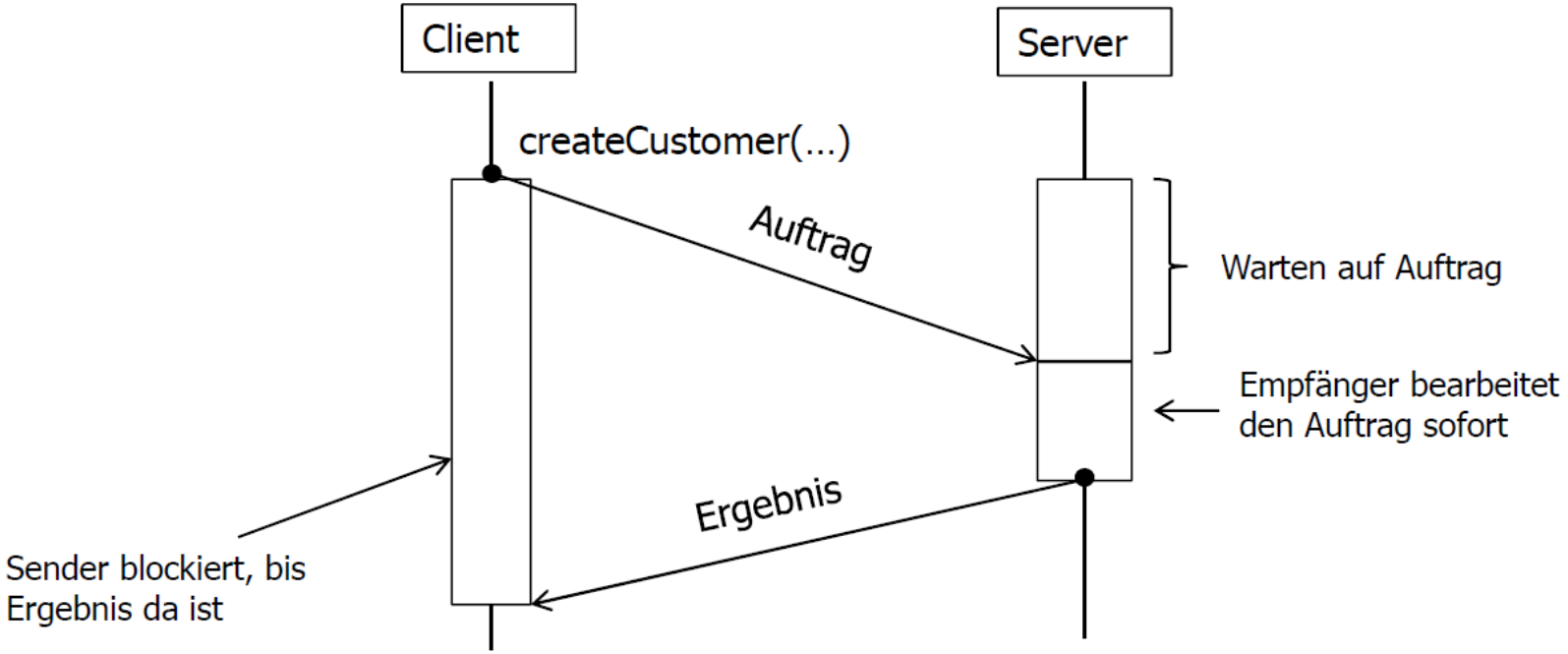
- Methodenaufruf und Parameterübergabe
  - ist lokal in demselben Prozess einfacher als bei entferntem (remote) Aufruf.
  - Entfernte Methodenaufrufe müssen für die Datenübertragung zwischen Rechnerknoten serialisiert (Marshalling) und deserialisiert (Unmarshalling) werden.
- Varianten für den entfernten Aufruf:
  - **Call-by-value:** Wert wird übergeben
    - Synonym: Call-by-copy
  - **Call-by-reference**: Verweis auf Variable wird übergeben
  - **Call-by-copy/copy-back**: Aufrufer arbeitet mit Kopie
    - Synonym: Call-by-restore = Call-by-value-result

Marshalling/Unmarshalling

- Marshalling/Unmarshalling ist das Umwandeln (Serialisierung/Deserialisierung) von strukturierten oder elementaren Daten für die Übermittlung an andere Prozesse.
- **Tag-basierte** Transfersyntax
  - Siehe ASN.1 mit BER (Basic Encoding Rules)
  - TLV-Kodierung (Type, Length, Value)
- **Tag-freie** Transfersyntax
  - Siehe Sun ONC XDR, CORBA CDR
  - Beschreibung der Daten aufgrund der Stellung in der Nachricht
  - Aufbau der Datenstrukturen ist dem Sender und dem Empfänger bekannt
- Meist automatische Erzeugung von Marshalling- und Unmarshalling-Routinen durch Compiler/Präcompiler
- Heute werden oft auch sprachunabhängige Notationen verwendet:
  - XML (Markup-Sprache), Tag-basiert
  - JSON (JavaScript Object Notation), Tag-basiert, sprachunabhängig?

Kommunikationsmodelle: Synchrone Kommunikation

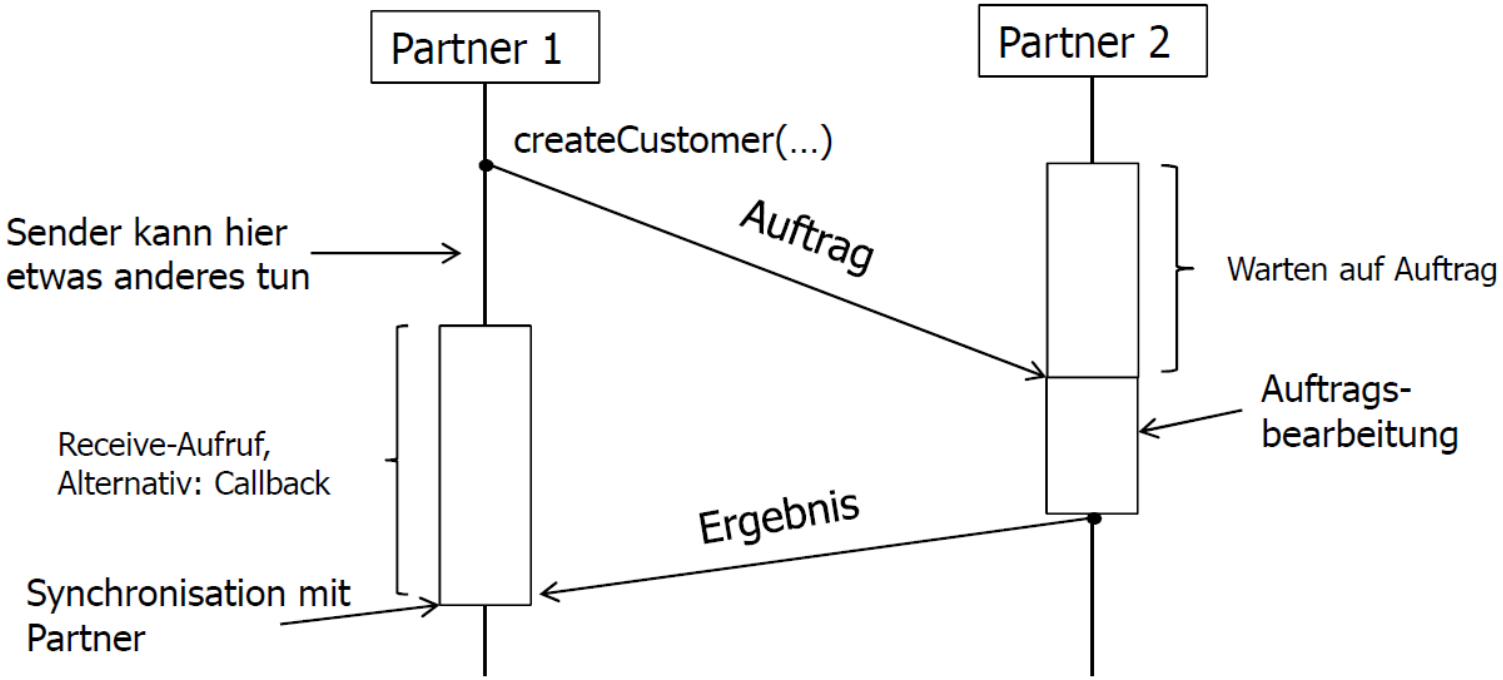
- Synchroner entfernter Dienstaufruf → blockierend
- Der Sender wartet, bis eine Methode send mit einem Ergebnis zurückkehrt



***Synchronisation** = **Synchronisierung (griech:** sýn = zusammen, chrónos = Zeit)**:** Aufeinander-Abstimmen von Vorgängen (zeitlich). Engere Bedeutung je nach Wissensgebiet: siehe Film, Informatik,…*

Kommunikationsmodelle: Asynchrone Kommunikation

- Asynchroner entfernter Serviceaufruf → Nicht blockierend, der Sender kann weiter machen



In der Datenkommunikation: asynchron = *Senden und Empfangen von Daten zeitlich versetzt und ohne Blockieren des Prozesses*

Ausgewählte Implementierungsaspekte von Client-Server-Systemen

- Wir betrachten im Weiteren einige ausgewählte Aspekte:
  - Heterogenität
  - Serverarchitektur
  - Nebenläufigkeit im Server (Parallelität)
  - Serverseitige Service- bzw. Dienstschnittstellen
  - Fehlersituationen, Fehlerklassierung
  - Parameterübergabe zwischen Client und Server
  - Marshalling/Unmarshalling
  - **Kommunikation**
  - Zustandsverwaltung
  - Garbage Collection
  - Lastverteilung, Verfügbarkeit, Skalierbarkeit

Kommunikation

- Namensauflösung und Adressierung auf der Anwendungsebene (entferntes Objekt oder Prozedur)
  - Naming- und Directory-Services notwendig
- Binding-Vorgang: Aufbau eines Verbindungskontextes zwischen Client und Server
  - Statisch zur Übersetzungszeit
  - Dynamisch zur Laufzeit
- Kommunikationsprotokoll für die Client-Server-Kommunikation
  - Nachrichtentypen (meist Request-Response-Protokolle)
  - Unterstützte Fehlersemantik
  - Unterstützung für verteiltes Garbage Collection

Zustandsverwaltung

- Server können zustandsinvariante und zustandsändernde Dienste bzw. Services anbieten
  - Zustandsändernde Dienste führen bei der Bearbeitung zu einer Änderung von Daten (z.B. in Datenbanken)
  - Zustandsinvariante Dienste verändern nichts
- Weiterer Aspekt: Server muss sich das Wissen über die Zustandsänderung über einen Aufruf hinweg merken
  - stateful und stateless Server
  - Stateless Server verwalten den aktuellen Zustand der Kommunikationsbeziehung zwischen Client und Server nicht
  - Wenn möglich: stateless!
- Zustandslose Kommunikationsprotokolle im Web: HTTP und REST für Webservices

Garbage Collection (GC)

- Verteiltes Reference-Counting
  - Server verwaltet eine Liste aller Clients (Proxies), die entfernte Referenzen nutzen
  - Server verwaltet Referenzzähler für alle benutzten Objekte
  - Client sendet spezielle Nachrichten an den Server, wenn Referenz benutzt bzw. gelöscht wird
- Leases
  - Referenz wird nur eine begrenzte Zeit für den Client freigegeben
  - Nach definierter Zeit löscht der Server die Referenz, wenn sich der Client nicht meldet
  - Ein Client kann sich somit problemlos beenden
- Zusammenarbeit mit lokalen GC-Mechanismen
  - Heap-Bereinigung

Lastverteilung, Hochverfügbarkeit, Skalierbarkeit

- Load Balancing (Lastverteilung)
  - Lastverteiler verteilen die Last auf mehrere Serverinstanzen
  - Dispatching z.B. über DNS-basiertes Request-Routing
- Hochverfügbarkeit
  - Server-Cluster, Beispiel: JBoss Cluster, Oracle Real Application Cluster
  - Failover
  - Session-Replikation
- Skalierbarkeit
  - Horizontal: Steigerung der Leistung durch Hinzunahme von Rechnern
  - Vertikal: Steigerung der Leistung durch Hinzufügen von Ressourcen zu einem Rechner (CPU, Speicher, …)