

Verteilte Systeme

Middleware

Sommersemester 2025

Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Systemsoftware)



Lehrstuhl für Informatik 4
Systemsoftware



**FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG**

TECHNISCHE FAKULTÄT

Middleware

Motivation

Heterogenität

Kommunikation

Common Object Request Broker Architecture (CORBA)

Web Services

■ Problem: **Heterogenität auf verschiedenen Ebenen**

Bereich	Beispiele
Netzwerke	Medium, Technik, Topographie
Rechner-Hardware	Peripherie, Speicher, CPU
Prozessoren	Informationsdarstellung, Endianness
Betriebssysteme	Ausführungsumgebung, API
Programmiersprachen	Semantik, Pragmatik

■ Mögliche Lösung: **Einsatz einer Middleware**

- Softwareschicht zur Abstraktion von den jeweiligen Systemeigenheiten
- Varianten
 - Sprachenabhängig (z. B. Java RMI)
 - Sprachenunabhängig (z. B. CORBA, Web Services)
- Grundlegende Bausteine
 - Prozesse
 - Kommunikation per Nachrichtenaustausch

Middleware

Motivation

Heterogenität

Kommunikation

Common Object Request Broker Architecture (CORBA)

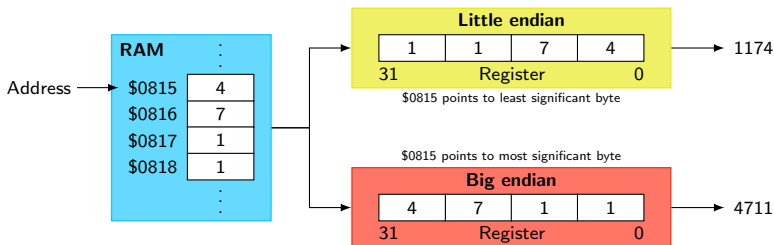
Web Services

Heterogene Datenrepräsentationen

■ Uneinheitliche Speicherung / Darstellung (elementarer) Datentypen

- Zeichensätze: ISO-8859-Familie (ASCII), BCD, EBCDIC, Unicode
- Natürliche/ganze Zahlen: vorzeichenbehaftet, {1,2}er-Komplement
- Fließkommazahlen: Basis, Mantisse, Exponent
- Speicherreihenfolge („Byte Sex“): *Big Endian*, *Little Endian*

■ Beispiel: Endianness



⇒ **Konvertierung von Daten erforderlich**

■ Beidseitig

- Datenübertragung in einer **kanonischen Darstellung** (= Standardformat)
- Bei Bedarf jeweils Konvertierung beim Sender und/oder Empfänger
- Problemszenario
 - Sender und Empfänger nutzen intern dieselbe Datenrepräsentation
 - Standard weicht von der Datenrepräsentation der Kommunikationspartner ab
 - Nutzloser Mehraufwand durch eigentlich unnötige Konvertierungen

■ Sendeseitig („Sender makes it right“)

- Sender übermittelt Daten in der **Repräsentation des Empfängers**
- Probleme
 - Sender benötigt Wissen über die lokale Datenrepräsentation des Empfängers
 - Kommunikation mit mehreren heterogenen Empfängern (z. B. Multicast)

■ Empfangsseitig („Receiver makes it right“)

- Sender übermittelt Daten in seiner **eigenen Repräsentation**
- Metadaten enthalten (kanonische) Bezeichnung des Datenformats
- Empfänger konvertiert Daten bei Bedarf in eigene Repräsentation

External Data Representation (XDR)

- Sprachbasierter Standard zur Beschreibung und Kodierung von Daten

- **Implizite Typung**

- Übermittlung der Variablenwerte
 - Variablentypen sind nicht Bestandteil von Nachrichten

- Annahme: Bytes bzw. Oktete (d. h. Einheiten von 8 Bits) sind portabel

- **Repräsentation von Daten**

- Blockgröße: 4 Bytes
 - Bei Bedarf: Auffüllen mit Null-Bytes

- Speicherreihenfolge: *Big Endian*

- **Bewertung**

- Bevorzugung von Big-Endian-Architekturen (z. B. Motorola 68k, SPARC)
 - Benachteiligung von Little-Endian-Architekturen (z. B. Intel x86)

- **Literatur**



Sun Microsystems

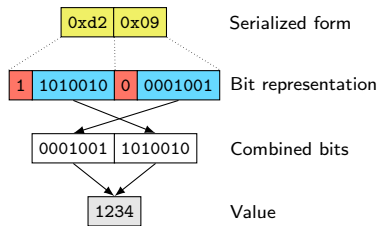
XDR: External Data Representation Standard, RFC 1014, 1987.

■ Ansatz zur Serialisierung bzw. Deserialisierung **strukturierter Daten**

- Entwicklung von Google
- Ziel: Reduzierung der zu übertragenden Datenmenge

■ Einsatz von **Integern variabler Länge** (*Varints*)

- Kodierung eines Werts als **Byte-Sequenz**
- Strukturierung eines Bytes
 - **Höchstwertiges Bit**
 - * 1, falls weitere Bytes folgen
 - * 0, falls letztes Byte der Sequenz
 - Restliche 7 Bits: **Nutzdaten**
- Beginn der Sequenz: Byte mit den niedrigstwertigen Nutzdaten



■ Literatur



Protocol Buffers Developer Guide – Encoding
<https://protobuf.dev/programming-guides/encoding/>

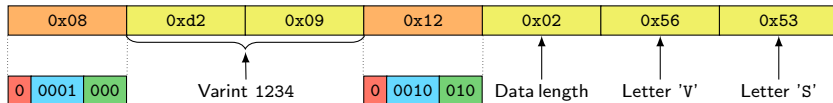
■ Datenbeschreibungssprache zur **Spezifizierung von Nachrichten**

- Einsatz vordefinierter Datentypen
- Vergabe von Feldindizes

```
message VSMMessage {  
  int32  key    = 1;    // Field #1  
  string value = 2;    // Field #2  
}
```

■ Kodierung einer VSMMessage mit key = 1234 und value = „VS“

- **Steuer-Varints** mit **Feldindex** und **Serialisierungstyp** (*Wire Type*)
- Beispiele für Serialisierungstypen
 - 000: Varint
 - 010: Längenbegrenzte Daten



Middleware

Motivation

Heterogenität

Kommunikation

Common Object Request Broker Architecture (CORBA)

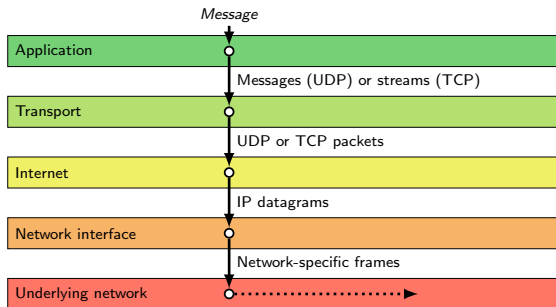
Web Services

- Kommunikation per **Nachrichtenaustausch**
 - Übertragung von Daten in Paketen begrenzter Länge
 - Einzelne Nachricht ist in manchen Fällen Teil eines Datenstroms
- Unterscheidung
 - Asynchrone Netze
 - **Keine Garantien** über die Verzögerungszeiten von Nachrichten
 - Regelfall in den meisten verteilten Systemen
 - Synchrone Netze
 - **Netzwerk garantiert Obergrenzen** für Verzögerungszeiten
 - Einsatz von Spezial-Hardware ermöglicht präzisere Fehlererkennung
- **Häufige Annahmen** bei der Entwicklung von verteilten Systemen
 - Asynchrones Netz
 - Unzuverlässige Zustellung von Nachrichten
 - Vermittlungsknoten (z. B. Router) sind für Kommunikationspartner transparent

- Regeln und Formate für den Austausch von Nachrichten

- **Aufteilung in Schichten**

- Trennung verschiedener Aufgaben, die bei einer Übertragung anfallen
- Schichtenspezifische Paketgrößen möglich
- Unterschiedliche Adressierung: Prozesse, Rechner, Vermittlungsknoten,...
- Eventuell **Verschmelzung von Schichten in der Implementierung**



■ *No-Wait Send*

- Sender wartet, bis die Nachricht **in seinem Kommunikationssystem** ist
- Keine Bestätigung, ob die Nachricht den Empfänger erreicht hat

■ *Synchronization Send*

- Sender wartet, bis die Nachricht **beim Empfänger** angekommen ist
- Synchronisation zwischen Sende- und Empfangsprozess

■ *Remote-Invocation Send*

- Sender wartet, bis die Nachricht **vom Empfänger bearbeitet** wurde
- Sender deblockiert erst, sobald er vom Empfänger eine Antwort erhält

■ Literatur



Barbara Liskov

Primitives for distributed computing

Proc. of the 7th Symposium on Operating Systems Principles (SOSP '79), S. 33–42, 1979.

■ Gruppenkommunikation

- Abstraktionsmechanismus für mehrere Nachrichtenempfänger
- Sender schickt **Nachricht an Gruppe**, nicht an die konkreten Empfänger

■ *Publish-Subscribe*-Mechanismen

- Sender (*Publisher*) verteilen Dateneinheiten (*Events*)
- Empfänger (*Subscriber*) registrieren sich anhand **bestimmter Kriterien**
- System übernimmt Übermittlung von Dateneinheiten an Empfänger

■ Tupelräume

- **Persistenter Speicherbereich** für strukturierte Dateneinheiten (*Tupel*)
- Kommunikation erfolgt per Erzeugen, Lesen und Löschen von Tupeln
- Sender und Empfänger müssen nicht zwingend gleichzeitig existieren

■ Verteilter **gemeinsamer Speicher**

- Sender und Empfänger greifen auf (scheinbar) lokale Datenstrukturen zu
- System sorgt für Synchronisation und Konsistenz

Middleware

Motivation

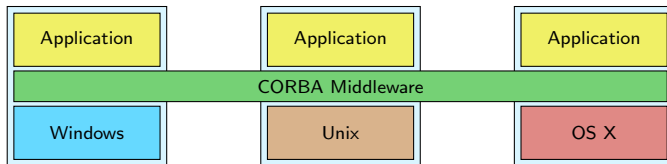
Heterogenität

Kommunikation

Common Object Request Broker Architecture (CORBA)

Web Services

- Common Object Request Broker Architecture (CORBA)
 - **Plattformunabhängige Middleware-Architektur** für verteilte Objekte
 - Erste umfangreiche Normierung von Middleware-Konzepten



- Interface Definition Language (IDL)
 - **Spezifizierung von Schnittstellen**
 - Automatische Generierung von Stubs und Skeletons für Fernaufrufe
- Literatur

[Nähere Details in der nächsten Vorlesung.]



Steve Vinoski

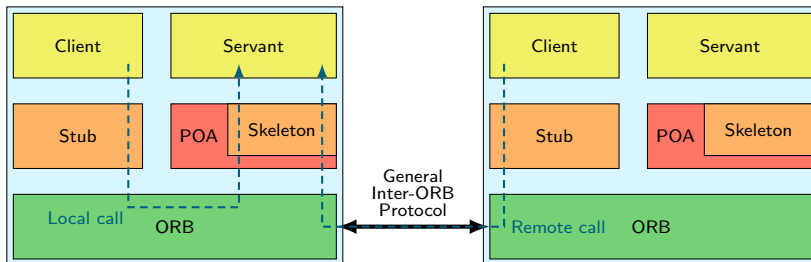
CORBA: Integrating diverse applications within distributed heterogeneous environments
IEEE Communications Magazine, 35(2):46–55, 1997.

■ Object Request Broker (ORB)

- **Vermittlung von Methodenaufrufen** zwischen Objekten
- Interaktion mit anderen ORBs

■ Portable Object Adapter (POA)

- Portabilität von Objektimplementierungen zwischen verschiedenen ORBs
- **Trennung von Objekt und Objektimplementierung** (*Servant*)
 - Objektimplementierung lässt sich bei Bedarf dynamisch aktivieren
 - Persistente Objekte können die Laufzeit eines Servants überdauern



Middleware

Motivation

Heterogenität

Kommunikation

Common Object Request Broker Architecture (CORBA)

Web Services

- Web Services Description Language (WSDL)
 - XML-basierte **Beschreibung von Diensten**
 - Bestandteile von WSDL-Beschreibungen
 - Datentypen und Schnittstellen
 - Kommunikationsendpunkte und Abbildung auf Kommunikationsprotokolle
- SOAP
 - **Kommunikationsprotokoll** für Web-Service-Nachrichten
 - Kontrollflussinformationen für Kommunikationspartner
 - Transport von Nachrichten mittels HTTP

„A **Web service** is a software system designed to support **interoperable machine-to-machine interaction over a network**. It has an interface described in a machine-processable format (specifically **WSDL**). Other systems interact with the Web service in a manner prescribed by its description using **SOAP messages**, typically conveyed **using HTTP** with an XML serialization in conjunction with other Web-related standards.“

[Web Services Architecture – W3C Working Group Note 11, <http://www.w3.org/TR/ws-arch/>]

Vergleich: CORBA vs. Web Services

■ CORBA

- Auslegung auf **einzelne Organisation**
- Einsatz eines eigenständigen Namensdiensts
- Installation einer eigenen Plattform erforderlich
- Nachrichten im Binärformat

■ Web Services

- Interaktion **unabhängiger, per Internet verbundener Knoten**
- Auflösung von Namen mittels DNS
- Gängige Technologien (z. B. HTTP, XML) als Basis
- Nachrichten im Textformat (XML)

■ Literatur



George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair
Distributed systems: Concepts and design
Addison-Wesley Publishing Company, S. 398–399, 2011.