

Plattformunabhängige Fernaufrufe

Motivation

Extensible Markup Language (XML)

Hypertext Transfer Protocol (HTTP)

XML-basierte Fernaufrufe (XML-RPC)

Zusammenfassung



Fernaufwurf (*Remote Procedure Call, RPC*)

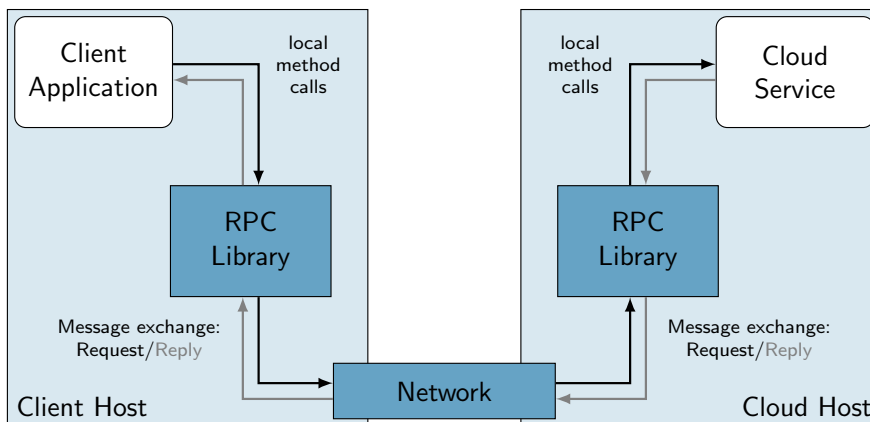
[Bei Interesse: Weiterführende Informationen in der Veranstaltung *Verteilte Systeme* im Sommersemester.]

- Kurzbezeichnung für Fern**methodenaufwurf**
- Unterschiede zu lokalem Methodenaufwurf
 - Aufruf und Ausführung einer Methode erfolgen auf verschiedenen Rechnern
 - Nachrichtenaustausch übers Netzwerk
 - Komplexeres Fehlermodell
- Einsatzszenarios (Beispiele)
 - Zugriff auf per Web-Service angebotene Cloud-Dienste
 - Auslagerung von aufwendigen Berechnungen in die Cloud
- Herausforderungen im Kontext von Cloud Computing
 - Wie lässt sich ein Fernaufruf soweit wie möglich *transparent* realisieren?
 - Aufrufer soll den Unterschied zu einem lokalen Methodenaufwurf nicht merken
 - Cloud-Zugriff soll verschattet werden
 - Wie kann ein plattformunabhängiges Fernaufrufprotokoll aussehen?



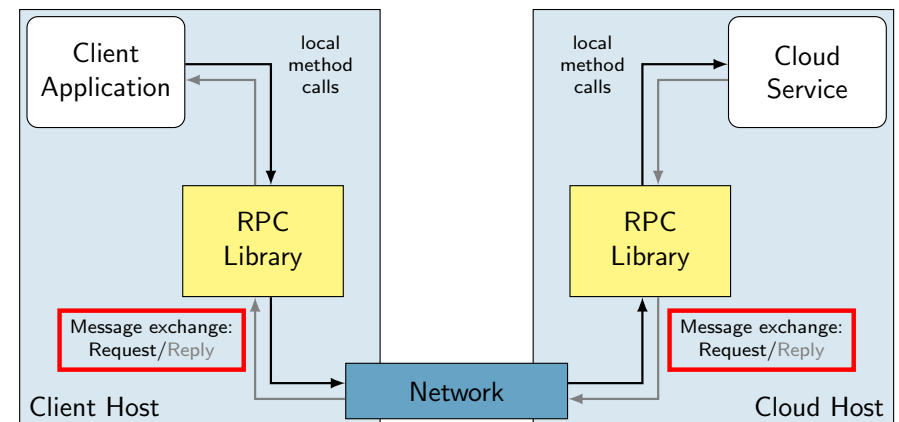
Fernaufwurf: Interaktion zwischen Client und Dienst

- Umwandlung des lokalen Aufrufs in Nachrichtenaustausch
 - Client sendet Anfragenachricht (*Request*)
 - Dienst sendet Antwortnachricht (*Reply*)




Plattformunabhängiges Fernaufrufprotokoll

- Anforderungen
 - Systemunabhängige Nachrichtenformate und Übertragungsprotokolle
 - Einsatz weit verbreiteter Standards



Extensible Markup Language (XML)

- Auszeichnungssprache für hierarchisch strukturierte Daten
- Überblick
 - Repräsentation in Textform
 - Intention: Lesbarkeit für Mensch und Maschine
 - Nachteile
 - * Mehraufwand für das Erzeugen und Einlesen von Nachrichten
 - * Erhöhtes Datenvolumen bei Übertragung über ein Netzwerk
 - Darstellung von Informationen
 - Ursprünglich: Format zur Repräsentation von Dokumenten
 - Häufig aber auch als allgemeines Datenformat im Einsatz (→ Web-Services)
- Literatur
 -  Tim Bray, Jean Paoli, Michael Sperberg-McQueen, Eve Maler et al. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**, 2008.



Aufbau eines XML-Dokuments

- Dokumentstruktur
 - XML-Deklaration

```
<?xml version="[Versionsnummer]" [...]?>
```
 - Wurzelement
- Bestandteile
 - Elemente

```
<[Elementname] [Optional: Attribut 1, Attribut 2,...]>[Inhalt]</[Elementname]>
```

 - Beschreibung mittels eindeutigem *Start-* und *End-Tag*
 - Inhalt
 - * Nutzdaten
 - * Weitere Elemente (→ Elementhierarchie)
 - Spezialfall: *Empty-Element-Tag* für leeres Element

```
<[Elementname] [Optional: Attribut 1, Attribut 2,...]/>
```
 - Attribute: Bereitstellung als Schlüssel-Wert-Paare

```
[Attributname]="[Attributwert]"
```



Namensräume

- Problem
 - Bei der Kombination mehrerer XML-(Teil-)Dokumente zu einem neuen XML-Dokument sind Elementnamen eventuell nicht eindeutig
- Lösung: Verwendung von XML-Namensräumen
 - Identifizierung eines Namensraums per Universal Resource Identifier (URI)
 - [Hinweis: Ein Namensraum-URI hat nur bezeichnenden Charakter; unter der angegebenen Adresse muss nicht notwendigerweise eine Ressource mit Kontextinformationen zum referenzierten Namensraum abrufbar sein.]
 - Definition eines Default-Namensraums [xmlns: XML_namespace]

```
<[Elementname] xmlns="[URI des Namensraums]">
  [Gültigkeitsbereich der Namensraumreferenz]
</[Elementname]>
```
 - Einsatz von Präfixen für verschiedene Namensräume (Beispiel)

```
<[Elementname] xmlns:[Präfix A]="[URI des Namensraums A]"
  xmlns:[Präfix B]="[URI des Namensraums B]">
  <[Präfix A]:[Subelementname]>[...]</[Präfix A]:[Subelementname]>
  <[Präfix B]:[Subelementname]>[...]</[Präfix B]:[Subelementname]>
</[Elementname]>
```



Überprüfung von XML-Dokumenten

- Wohlgeformtheit (*Well-formedness*)
 - Dokument entspricht der XML-Spezifikation
 - Überprüfung der Syntax
 - Kriterien (Beispiele)
 - Einzelnes Wurzelement
 - Zu jedem Start-Tag muss ein korrespondierender End-Tag existieren
 - Keine Überlappung von Elementen
 - Attributname muss innerhalb eines Start-/Empty-Element-Tag eindeutig sein
 - [Hinweis: Die ersten drei Kriterien garantieren eine Baumstruktur des XML-Dokuments.]
- Gültigkeit (*Validity*)
 - Dokument entspricht der XML-Spezifikation sowie weiteren Kriterien
 - Zusätzliche Überprüfung des Dokuments auf Konformität zu einem im XML-Dokument referenzierten, zumeist separat bereitgestellten Schema
 - Festlegung von Regeln, z. B. mittels *XML Schema Definition (XSD)*
 - Definition der erlaubten Elemente und Attribute



■ Überblick

- Strukturbeschreibung von XML-Dokument-Klassen
 - Datentypen
 - Elemente
 - Attribute
- XML-Schema-Definitionen sind selbst XML-Dokumente


■ Einsatzbereiche

- Validierung von XML-Dokumenten
- Beschreibung der in Web-Service-Schnittstellen verwendeten Datentypen

■ Einbindung einer XML-Schema-Definition in ein XML-Element

```
<[Elementname] xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
    xsd:schemaLocation="[URI auf XML-Schema-Definition]"/>
```

■ Literatur

 David C. Fallside and Priscilla Walmsley
XML Schema Part 0: Primer Second Edition, 2004.



■ Primitive Datentypen

Datentyp	Beschreibung
boolean	Mögliche Werte: <i>wahr</i> (true bzw. 1), <i>falsch</i> (false bzw. 0)
decimal	Zahl der Form $i * 10^{-n}$ mit $i, n \in \mathbb{N}_0$
string	Zeichenkette
dateTime	Kombination aus Datum und Uhrzeit
anyURI	Absolute oder relative URI
[14 weitere...]	

■ Abgeleitete Datentypen

Datentyp	Abgeleitet von	Beschreibung
integer	decimal	Ganze Zahl
normalizedString	string	Zeichenkette ohne \r, \n oder \t
[23 weitere...]		

■ anyType: Abstrakter Basistyp für alle anderen Datentypen



■ Definition eigener einfacher Datentypen (simpleType)

```
<xsd:simpleType name="[Name des Datentyps]">
    [Beschreibung des Datentyps]
</xsd:simpleType>
```

■ Liste

```
<xsd:list itemType="[Datentyp der Listenelemente]"/>
```

■ Union: Datentyp mit kombiniertem Wertebereich

```
<xsd:union memberTypes="[Aufzählung erlaubter Datentypen]"/>
```

■ Ableitung eines Basisdatentyps mit Einschränkung des Wertebereichs

```
<xsd:restriction base="[Basisdatentyp]">
    [Einschränkung des Basisdatentyps]
</xsd:restriction>
```

- Aufzählung: Festlegung bestimmter zulässiger Werte (enumeration)
- Minimal-/Maximalwerte für Integer (minInclusive, maxInclusive)
- Reguläre Ausdrücke für Zeichenketten (pattern)
- ...



■ Definition eigener komplexer Datentypen (complexType)

```
<xsd:complexType name="[Name des Datentyps]">
    [Beschreibung des Datentyps]
</xsd:complexType>
```

■ Festlegung der Reihenfolge von Subelementen

```
<xsd:sequence>[Subelemente]</xsd:sequence>
```

■ Referenzierung von existierenden Datentypen

```
<xsd:element ref="[Name des Datentyps]"/>
```

■ Spezifizierung eigener Elementnamen und Zuordnung zu Datentypen

```
<xsd:element name="[Elementname]" type="[Name des Datentyps]"/>
```

■ Definition eigener Attribute (nur einfache Datentypen erlaubt)

```
<xsd:attribute name="[Attributname]" type="[Name des Datentyps]"/>
```



■ Attribute zur Einschränkung der Anzahl von Elementen

- Festlegung einer Mindest- bzw. Maximalanzahl (minOccurs, maxOccurs)
- Für optionale Elemente: minOccurs auf 0 setzen



- Protokoll für Zugriff auf *Ressourcen* über ein Netzwerk
 - Lesender Zugriff (z. B. Abruf einer Web-Seite)
 - Schreibender Zugriff (z. B. Hochladen einer Datei)
- Überblick
 - Zumeist TCP/IP als zuverlässiges Transportprotokoll
 - Eigentlich Anwendungsprotokoll, kommt aber oftmals auch selbst als Transportprotokoll zum Einsatz (siehe XML-RPC)
 - Textbasierter Nachrichtenaustausch
 - Zustandsloses Protokoll

Literatur

-  Tim Berners-Lee, Roy Fielding, and Henrik Frystyk
Hypertext Transfer Protocol – HTTP/1.0, RFC 1945, 1996.
-  Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter et al.
Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, 1999.



Header

```
[Methode] [Ressourcen-ID] HTTP/[Versionsnummer]\r\n[Feld 1: Name]: [Feld 1: Wert]\r\n[Weitere Felder...]\r\n
```

- Startzeile
 - Methode: Operation, die auf der adressierte Ressource ausgeführt werden soll
 - Ressourcen-ID: Adresse der Ressource, auf die zugegriffen werden soll
- Nachrichtenkopf
 - Liste von Schlüssel-Wert-Paaren (*Feldern*)
 - Beispiele

Feldname	Beschreibung
Host	Name des Rechners, der die adressierte Ressource verwaltet
User-Agent	Client-Anwendung, z. B. Web-Browser
If-Modified-Since	Zeitstempel für bedingte Ausführung
Content-Type	Art der Nutzdaten, z. B. text/html (HTML-Seite)
Content-Length	Länge der Nutzdaten

Body (optional): Nutzdaten



Überblick über die wichtigsten HTTP-Methoden

Methode	Beschreibung
GET	Lesender Zugriff auf die adressierte Ressource
HEAD	Unterschied zu GET: Antwort enthält keinen Body
POST	Modifizierender Zugriff auf die adressierte Ressource (Beispiele) <ul style="list-style-type: none">– Annotation einer existierende Ressource– Senden einer Nachricht an eine Mailing-Liste– Übermittlung von Formular Daten– Anfügen eines Datensatzes an eine Datenbank
PUT	Registrieren von Daten unter der übergebenen Ressourcenadresse
DELETE	Löschen der adressierten Ressource

Weitere Methoden: OPTIONS, TRACE, CONNECT, ...

[RFC 2616]

Zentrale Methoden für die Browser-Server-Kommunikation

- GET
- POST



Universal Resource Identifier (URI)

- Zeichenkette zur global eindeutigen Identifikation einer Ressource
- Struktur

```
"http://" [Host] [":" [Port]] [[Absoluter Pfad]]
```

Beispiel

```
http://www4.cs.fau.de/Lehre/WS12/V_MW/index.ushtml
```

Absoluter Pfad

- Lokale Adresse einer Ressource auf dem Zielrechner
- Globale Eindeutigkeit ist eventuell nur in Kombination mit dem Host-Feld im HTTP-Header gegeben
- Beispiel-Header

```
GET /Lehre/WS12/V_MW/index.ushtml HTTP/1.1  
Host: www4.cs.fau.de  
[...]
```



Aufbau einer Antwortnachricht

■ Header

```
HTTP/[Versionsnummer] [Status-Code] [Statusnachricht]\r\n[Feld 1: Name]: [Feld 1: Wert]\r\n[Weitere Felder...]\r\n
```

■ Statuszeile

- Status-Code: Als Zahl repräsentierte Statusmeldung
- Statusnachricht: Textuelle Beschreibung des Status-Codes

Klasse	Kategorie	Beschreibung
1xx	Informell	Anfrage wurde empfangen, Bearbeitung erfolgt
2xx	Erfolg	Anfrage wurde empfangen, verstanden und akzeptiert
3xx	Weiterleitung	Weitere Aktionen notwendig
4xx	Client-Fehler	Anfrage war fehlerhaft
5xx	Server-Fehler	Anfrage war korrekt, aber im Server lag ein Fehler vor

[RFC 2616]

■ Nachrichtenkopf: Struktur analog zu Nachrichtenkopf in Anfragenachricht

■ Body (optional): Nutzdaten



Kommunikation über TCP/IP

■ Transmission Control Protocol (TCP)

- Zuverlässiges Transportprotokoll
- Verbindungsorientierte Kommunikation
- Mechanismus für Flusskontrolle
- Erhaltung der Nachrichtenreihenfolge

■ Einsatz von TCP/IP in unterschiedlichen HTTP-Versionen

■ HTTP/1.0

- Separate TCP/IP-Verbindung pro Anfrage-Antwort-Interaktion
- Nachteil: Zusätzliche Last für Rechner und Netzwerk

■ HTTP/1.1

- Persistente Verbindungen: Wiederverwendung der TCP/IP-Verbindung für weitere Anfragen an den selben Server
- Vorteile
 - * Minimaler Mehraufwand für Verbindungsaufbau
 - * *Pipelining* möglich: Parallele, asynchrone Anfragen eines Clients



Indirekte Kommunikation

■ Aufrufketten

- Paarweise Punkt-zu-Punkt-Verbindungen
- Anwendungsprozesse als Bindeglied zwischen Verbindungen

■ Beispiele für als Vermittler fungierende Anwendungsprozesse

■ Proxy

- Abfangen und Weiterleiten von Anfragen und Antworten
- Vom Client explizit ausgewählt, um mit dem Server zu kommunizieren
- Eventuell Modifikation der Nachrichten

■ Gateway

- Knoten an der Schwelle zwischen zwei heterogenen Netzwerken
- Client sieht Gateway als eigentlichen Server an
- Eventuell Modifikation der Nachrichten

■ Tunnel

- Transparenter Vermittler zwischen zwei Verbindungen
- Keine Modifikation der Nachrichten



XML-RPC

■ Plattformunabhängiges Fernaufrufprotokoll

- XML: Nachrichtenformat
- HTTP: Transportprotokoll

■ Umfang

■ Spezifikation

- Festlegung der verfügbaren Datentypen
- Aufbau von Anfrage- und Antwortnachrichten

■ Implementierungen für verschiedene Programmiersprachen

- Java
- C++
- Python
- ...

■ Literatur



XML-RPC

<http://xmlrpc.scripting.com/>



- Primitive Datentypen (<value>)

XML-Tag	Wert
i4 bzw. int	32-Bit-Integer
double	Fließkommazahl mit doppelter Genauigkeit
boolean	0 (false), 1 (true)
string	Zeichenkette
base64	Base64-codierte Binärdaten
dateTime.iso8601	Datum und Uhrzeit

[XML-RPC Specification, <http://xmlrpc.scripting.com/spec>]

- Komplexe Datentypen

- Liste primitiver und/oder komplexer Datentypen (<array>)

```
<array><data>
  <value>[Wert]</value>
  [Weitere <value>-Elemente]
</data></array>
```

- Ungeordnete Menge aus Schlüssel-Wert-Paaren (<struct>)

```
<struct>
  <member><name>[Schlüssel]</name><value>[Wert]</value></member>
  [Weitere <member>-Elemente]
</struct>
```

- HTTP-Header

```
POST [URI] HTTP/1.1
Content-Type: text/xml
User-Agent: [Nutzeranwendung]
Content-Length: [Nachrichtenlänge]
Host: [Zielrechneradresse: z.B. Hostname und Port]
[...]
```

- POST und Content-Type
 - Identisch für alle Anfragen
 - Intention: Unterstützung einfacher Firewall-Regeln
- URI
 - Beliebiger Pfad (möglicherweise leer)
 - Ermöglicht die Request-Handler-Auswahl auf Server-Seite

- HTTP-Body

```
<?xml version="1.0"?>
<methodCall>
  <methodName>[Methodenname]</methodName>
  <params>
    <param>[Aufrufparameter]</param>
    [...]
  </params>
</methodCall>
```

- Kapselung der Aufrufinformationen in einzelner <methodCall>-Element
- Methodenname
 - Eindeutige Identifikation der aufzurufenden Methode
 - Beliebige Zeichenkette
 - Beispiel: Zusammengesetzter Name aus Objekt-ID und Methodenname
- Aufrufparameter: Primitive oder komplexe Datentypen

- HTTP-Header

```
HTTP/1.1 200 OK
Server: [Server-Anwendung]
Content-Type: text/xml
Content-Length: [Nachrichtenlänge]
[...]
```

- Identisch für beide Antworttypen
- Status-Code bezieht sich auf HTTP, nicht auf den Methodenaufruf

- Reguläre Antwort: HTTP-Body

```
<?xml version="1.0"?>
<methodResponse>
  <params><param>[Rückgabewert]</param></params>
</methodResponse>
```

- Kapselung des Rückgabewerts in einzelner <methodResponse>-Element
- Rückgabewert: Primitiver oder komplexer Datentyp in <param>-Element

■ Fehlermeldung: HTTP-Body

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value><struct>
      <member>
        <name>faultCode</name>
        <value><i4>[Fehler-Code]</i4></value>
      </member>
      <member>
        <name>faultString</name>
        <value><string>[Fehlerbeschreibung]</string></value>
      </member>
    </struct></value>
  </fault>
</methodResponse>
```

- Kapselung der Fehlermeldung in individuellem <fault>-Element
- Semantik des Fehler-Codes von der Anwendung frei wählbar



- Extensible Markup Language (XML)
 - Auszeichnungssprache für hierarchisch strukturierte Daten
 - Kriterien für Wohlgeformtheit und Gültigkeit von Dokumenten
 - Definition eigener Datentypen mit XML Schema
- Hypertext Transfer Protocol (HTTP)
 - Protokoll für Zugriff auf Ressourcen über ein Netzwerk
 - Oftmals TCP/IP als Transportprotokoll
 - Zustandslose Interaktion
- XML-basierte Fernaufrufe (XML-RPC)
 - Plattformunabhängiger Fernaufrufmechanismus
 - Austausch von XML-Dokumenten
 - HTTP als Transportprotokoll
- Ausblick: XML-RPC nicht flexibel genug für komplexe Web-Services
 - SOAP: Kommunikationsprotokoll für Web-Services
 - REST: Grundprinzipien für die Entwicklung von skalierbaren Web-Services

